

LAPORAN AKHIR

PROJECT-BASED LEARNING

MATA KULIAH ALGORITMA DAN PEMROGRAMAN LANJUT

TAHAP 1: RANCANGAN *SCRIPT* PEMROGRAMAN FUNGSIONAL

DAN/ATAU OOP PADA STUDI KASUS

NATURAL LANGUAGE PROCESSING

KELAS C



**“Automatic Text Summarization Menggunakan Metode
Ekstraktif untuk Artikel Berbahasa Indonesia”**

DISUSUN OLEH KELOMPOK “II” :

- | | |
|---------------------------|-------------------------|
| 1. KINANTHI PUTRI ARIYANI | (21083010047) - ANGGOTA |
| 2. FADLILA AGUSTINA | (21083010050) - ANGGOTA |
| 3. AISYAH KIRANA PUTRI I. | (21083010065) - ANGGOTA |
| 4. ANIYSAH FAUZIYYAH ALFA | (21083010083) - KETUA |

DOSEN PENGAMPU:

TRESNA MAULANA FAHRUDIN, S.S.T., MT (20219930501200)
SUGIARTO, S.KOM., M.KOM (198702142021211001)

PROGRAM STUDI SAINS DATA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”
JAWA TIMUR
2022

KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan hidayah-Nya serta karunia-Nya sehingga penyusun dapat menyelesaikan laporan yang berjudul “*Automatic Text Summarization Menggunakan Metode Ekstraktif untuk Artikel Berbahasa Indonesia*” ini dengan baik.

Terselesaikannya laporan ini tentu tidak lepas dari bantuan banyak pihak. Oleh karena itu, kami mengucapkan terima kasih yang setulusnya kepada :

1. Bapak Tresna Maulana Fahrudin, S.S.T., MT dan Bapak Sugiarto, S.Kom., M.Kom. selaku Dosen Pengampu Mata Kuliah Algoritma dan Pemrograman Lanjut.
2. Teman-teman kelompok yang kompak dalam menyusun laporan ini.

Laporan ini disusun untuk melengkapi tugas Mata Kuliah Algoritma dan Pemrograman Lanjut. Selain itu, kami berharap semoga laporan ini dapat bermanfaat bagi semua pihak dan menjadi referensi untuk menambah wawasan dan ilmu pengetahuan.

Oleh karena itu, kami mengharapkan segala kritik dan saran yang membangun dan dapat menjadikan laporan ini lebih baik. Kami mohon maaf atas kesalahan maupun kekurangan dalam penyusunan laporan ini.

Semoga dengan kami membuat laporan ini dapat bermanfaat dan memberikan motivasi bagi para pembaca, khususnya bagi kami dan para generasi muda yang akan datang.

Surabaya, 15 Maret 2022

Penyusun

DAFTAR ISI

KATA PENGANTAR	i
BAB I: PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Permasalahan	2
1.3 Tujuan	2
1.4 Manfaat	2
BAB II: TINJAUAN PUSTAKA	3
2.1 Teori Penunjang	3
2.1.1 Extraction-based Summarization	3
2.1.2 Abstractive Summarization	4
2.1.3 Algoritma TextRank	4
2.1.4 Bahasa Python	4
2.1.5 Anaconda	5
2.1.6 Cosine Similarity	5
2.2 Penelitian Terkait	6
2.2.1 Automatic Text Summarization Menggunakan Metode Graph dan Ant Colony Optimization	6
2.2.2 Implementasi Metode Recurrent Neural Network Pada Text Summarization Dengan Teknik Abstraktif	7
2.2.3 Latent Semantic Analysis (LSA) dan Automatic Text Summarization (ATS) dalam Optimasi Pencarian Artikel Covid-19	7
BAB III: METODOLOGI PENELITIAN	9
3.1 Desain Sistem	9
3.2 Arsitektur Sistem	10
3.2.1 Penjelasan Arsitektur Sistem	10
3.3 Proses Penelitian	11
3.3.1 Tahap preprocess	11

3.3.2 Main processing	12
BAB IV: HASIL DAN PEMBAHASAN	14
4.1 Hasil Proyek	14
4.2 Pembahasan	16
BAB V: KESIMPULAN	23
DAFTAR PUSTAKA	24
LAMPIRAN	25

DAFTAR GAMBAR

Gambar 2.1 Python	4
Gambar 2.2 Anaconda	5
Gambar 2.3 Rumus CosSim	6
Gambar 3.1 Arsitektur Sistem	9
Gambar 3.2 Diagram Alir (flowchart) Text Summarization	10
Gambar 3.3 Membaca Artikel	10
Gambar 3.4 Melakukan Pre-Processing	10
Gambar 4.1 Percobaan 1	14
Gambar 4.2 Percobaan 2	14
Gambar 4.3 Percobaan 3	14
Gambar 4.4 Percobaan 4	15
Gambar 4.5 Percobaan 5	15
Gambar 4.6 Percobaan 6	15
Gambar 4.7 Percobaan 7	15
Gambar 4.8 Percobaan 8	16
Gambar 4.9 Percobaan 9	16
Gambar 4.10 Tampilan text setelah dirangkum menggunakan UI	19

Gambar 4.11 Proses install auto-py-to-exe	20
Gambar 4.12 Proses membuka py installer	20
Gambar 4.13 Proses memasukkan library	21
Gambar 4.14 Proses convert to exe	21
Gambar 4.15 Hasil convert	22
Gambar 4.16 Tampilan file type	22
Gambar 4.17 Tampilan User Interface	22
Gambar a.1 Dokumentasi diskusi kelompok melalui google meet	25
Gambar a.2 Dokumentasi penggerjaan laporan di google docs	25
Gambar a.3 Dokumentasi referensi dari youtube	26
Gambar a.4 Dokumentasi script pada jupyterlab	26

DAFTAR TABEL

Tabel 1. Stopwords and Numbers	12
--------------------------------	----

BAB 1: PENDAHULUAN

1.1 Latar Belakang

Peringkas teks atau biasa disebut *text summarization* merupakan salah satu materi yang bertujuan untuk memberikan kemudahan kepada para pembaca agar lebih mudah mencari informasi secara cepat. Hal ini ditandai dengan semakin banyaknya orang yang melakukan *browsing* sebagai bentuk kebutuhan akan informasi yang lebih cepat dan menyingkat waktu, tetapi tidak mungkin setiap pengguna untuk mencari atau membaca keseluruhan dokumen. Hal inilah yang harus diperhatikan, oleh karena itu metode yang memungkinkan pengguna untuk mencari dan menelusuri informasi dengan cepat dalam koleksi dokumen. Peringkasan dokumen tunggal telah menjadi subjek fokus beberapa tahun ini dalam masalah kompresi, redundansi kecepatan, dan pemilihan bagian sangat penting dalam pembentukan ringkasan yang berguna. (Ridwan, 2018)

Untuk mempermudah dalam meringkas suatu artikel terdapat dua pendekatan pada peringkasan teks, yaitu ekstraksi (*shallow approaches*) dan abstraksi (*deeper approaches*). Pada teknik ekstraksi, sistem menyalin unit-unit teks yang dianggap paling penting atau paling informatif dari teks sumber menjadi ringkasan. Unit-unit teks yang disalin dapat berupa klausa utama, kalimat utama, atau paragraf utama. Sedangkan teknik abstraksi melibatkan parafrasa dari teks sumber. Pada umumnya, abstraksi dapat meringkas teks lebih kuat daripada ekstraksi, tetapi sistemnya lebih sulit dikembangkan karena mengaplikasikan teknologi *natural language generation* yang merupakan bahasan yang dikembangkan tersendiri.

Penelitian mengenai peringkasan teks otomatis (*automatic text summarization*) dengan menggunakan berbagai macam metode dan pendekatan, diawali sejak tahun 1958 oleh Luhn. Banyak teknik yang digunakan dalam *summarization* ini, seperti teknik pendekatan statistika yaitu teknik *word frequency* (Luhn, 1958), *position in text* (Baxendale, 1958), *cue words and heading* (Edmundson, 1969), *sentence position* (Lin dan Hoovy, 1997).

Berdasarkan jumlah sumbernya, sebuah ringkasan dapat dihasilkan dari satu sumber (*single-document*) atau dari banyak sumber (*multi-document*). Peringkasan *single-document* masukannya berupa sebuah teks dan keluarannya berupa sebuah teks baru yang lebih singkat. Pada peringkasan *multi-document* adalah memasukkan beberapa dokumen teks yang memiliki tema sama, biasanya sudah ada dalam satu klaster kemudian akan dihasilkan keluaran berupa sebuah teks yang lebih singkat yang merangkum informasi-informasi utama pada klaster masukan.

Suatu ringkasan dapat bersifat general, yaitu ringkasan yang berupaya mengambil sebanyak mungkin informasi penting yang mampu menggambarkan keseluruhan isi teks. Selain itu dapat juga informasi yang diambil untuk ringkasan berdasar pada query masukan yang didefinisikan pengguna sistem. *Query Oriented* atau *user-oriented summarization* mencoba mengambil informasi yang relevan dengan *query* pengguna dan menampilkannya dalam bentuk ringkasan.

Berdasarkan fungsinya, sebuah ringkasan dapat memiliki sifat *indicative*, *informative or evaluative*. Ringkasan *information* berfungsi menyajikan informasi utama atau yang paling penting dari teks sumber. Ringkasan *indicative* memberikan saran untuk pembacaan lebih lanjut mengenai hal-hal tertentu dalam isi teks. Sedangkan ringkasan *evaluative* memberi komentar atau evaluasi terhadap informasi utama pada teks sumber.

Dengan adanya permasalahan tersebut maka dibentuklah sistem perangkat lunak yaitu algoritma *automatic text summarization* dengan *compression rate* atau proses peringkasan nantinya akan menentukan panjang ringkasan yang dihasilkan. Biasanya diukur berdasarkan persentase dari teks sumber, misalnya ringkasan sepanjang 10%, 25%, atau 50% dari teks sumber. Selain itu dapat pula diukur berdasarkan jumlah kata, misalnya ditentukan ringkasan sepanjang 100 kata. Biasanya, panjang ringkasan tidak lebih dari setengah teks sumber. (Ridwan Atmala, 2013)

Dengan dibuatnya makalah yang membahas tentang *Text Summarization* ini diharapkan dapat membantu para pembaca dapat memahami mengenai konsep dasar *text summarization*, jenis ringkasan dari *text summarization* dan pendekatan pada *text summarization*.

1.2 Permasalahan

Berdasarkan uraian pada latar belakang, permasalahan yang dibahas pada penelitian ini sebagai berikut:

- Bagaimana cara membuat algoritma untuk meringkas suatu artikel secara otomatis menggunakan metode *Extraction-based summarization*?

1.3 Tujuan

Dari rumusan masalah yang tersaji, tujuan dari penelitian ini adalah untuk mengetahui sistem dalam meringkas suatu teks pada sebuah artikel, mengetahui algoritma yang digunakan dalam meringkas artikel, dan untuk mengambil suatu artikel yang panjang dan mengubahnya menjadi ringkasan yang dapat dengan mudah untuk dibaca dan dipahami tanpa kehilangan makna teks aslinya.

1.4 Manfaat

Manfaat yang kami dapatkan dari penelitian ini adalah dapat mengurangi waktu membaca bagi pengguna, selanjutnya pada saat meneliti ringkasan artikel, membuat proses pemilihan kalimat penting dalam suatu artikel tersebut menjadi lebih mudah dan akurat sehingga dapat membantu kita dalam mengingat isi dari artikel yang begitu panjang.

BAB II: TINJAUAN PUSTAKA

Beberapa landasan teori yang berkaitan dengan topik bahasan, digunakan untuk mendukung project ini :

2.1 Teori Penunjang

Text summarization (perangkum teks) adalah pendekatan yang bisa digunakan untuk meringkas atau memadatkan teks artikel yang panjang menjadi lebih pendek dan ringkas, sehingga hasil rangkuman teks yang relatif lebih pendek dapat mewakilkan teks yang panjang. Pada dasarnya menggunakan prinsip *Natural Language Processing* dan algoritma-algoritma untuk membuat sistem mengerti artikel-artikel dan menghasilkan rangkuman yang lebih pendek dan efisien. *Automatic summarization* (otomasi perangkuman) adalah proses mengurangi dokumen teks menggunakan program komputer dalam rangka membuat rangkuman yang menyimpan titik-titik terpenting dari dokumen asli. *Automatic data summarization* adalah wilayah yang sangat penting dalam *machine learning* dan *data mining*. Saat ini, teknologi *summarization* digunakan pada banyak sektor industri. Sebuah contoh adalah *search engine* seperti *Google*. Contoh lain yang termasuk yaitu *document summarization*, *image collection summarization* dan *video summarization*. Inti dari *summarization* adalah menemukan bagian yang representatif, yang juga mengandung informasi dari seluruh bagian. Macam-macam *summarization* dijelaskan pada bagian berikut ini.

2.1.1 Extraction-based summarization

Ekstraksi berarti memilih unit teks (kalimat, segmen-semen kalimat, paragraf atau passages), lalu dianggap berisi informasi penting dari dokumen dan menyusun unit-unit ini dengan cara yang benar. Menurut Radev et al., algoritma-algoritma berbasis ekstraksi terbagi menjadi tiga, yaitu :

1. Surface-level

Metode *sentence scoring* mengambil 8 nilai dari sebuah kalimat diantaranya adalah TF/IDF, huruf besar, kata benda, frasa isyarat, data numerik, panjang kalimat, posisi kalimat dan kesamaan dengan judul yang kemudian nilainya diproses sehingga menjadi skor dari sebuah kalimat. Skor tersebut menjadi hasil data latih dimana data tersebut digunakan sebagai acuan *decision tree* untuk menentukan sebuah kalimat termasuk dalam rangkuman atau tidak. (Sabuna, 2017)

2. Intermediate-level

Melakukan penelitian perangkum *ekstraktif* dengan menggunakan *lexical chain*. Barzilay menyatakan bahwa ekstraksi rangkuman dapat dilakukan dengan menggunakan *lexical chain* sebagai model topik pembahasan. (Barzilay, 1997)

3. Deep Parsing

Menggunakan metode *summarization* yang berdasarkan *Rhetorical Structure Theory* (RST). RST merupakan sebuah kerangka kerja yang

dirancang untuk perhitungan struktur teks pada tingkat klausa. Sistem mengutip (*extract*) struktur retorik dari teks dan menggabungkan relasi retorik antara kalimat-kalimat dan memotong bagian yang kurang penting. (Chengcheng, 2010)

2.1.2 *Abstractive summarization*

Teknik ekstraksi hanya menyalin informasi yang dianggap paling penting oleh sistem (*key clause*, kalimat atau paragraf-paragraf), teknik abstraksi melibatkan *paraphrasing section* dari dokumen asal. Pada umumnya, abstraksi dapat membentuk teks lebih kuat daripada ekstraksi, namun program yang mampu melakukannya lebih sulit untuk dikembangkan mengetahui syarat sistem tersebut harus menggunakan teknologi *natural language generation*, dimana teknologi tersebut masih merupakan bidang yang berkembang. (Maybury, 1999)

2.1.3 Algoritma *TextRank*

TextRank merupakan metode pemeringkatan berdasarkan graf (*graph-based ranking*) dimana graf merupakan hasil ekstraksi dari teks menggunakan *natural language text*. Graf pada *TextRank* yaitu graf berbobot tak berarah. Algoritma *graph-based ranking* merupakan sebuah cara menentukan simpul-simpul penting pada graf, berdasarkan pada informasi global yang digambarkan secara rekursif dari keseluruhan graf. (Mihalcea, 2004)

2.1.4 Bahasa Python



Gambar 2.1 Python

Python adalah bahasa pemrograman interpretatif yang dapat digunakan di berbagai platform dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode dan merupakan salah satu bahasa populer yang berkaitan dengan *Data Science*, *Machine Learning*, dan *Internet of Things* (IoT). (Sumber : dicoding.com)

2.1.5 Anaconda



Gambar 2.2 Anaconda.

Anaconda adalah distribusi bahasa pemrograman *Python* dan *R* untuk komputasi ilmiah, yang bertujuan untuk menyederhanakan manajemen dan penerapan paket. Distribusi ini mencakup paket ilmu data yang cocok untuk *Windows*, *Linux*, dan *macOS*. (Sumber: anaconda.com)

2.1.6 Cosine Similarity

Seperti namanya, ukuran ini menghitung nilai cosinus sudut antara dua vektor. Vektor yang digunakan dalam perhitungan ini adalah vektor merepresentasikan teks yang ada.

Karena perhitungan *cosine similarity* membutuhkan tipe data berbentuk vektor, maka perlu dilakukan transformasi data teks dengan vektor. Vektorisasi teks adalah teknik untuk mengubah teks menjadi bentuk yang mudah dipahami oleh mesin yaitu vektor, susunan dari angka-angka bilangan bulat. Setiap kalimat yang ada, diekstrak dan direpresentasikan dalam bentuk vektor. (K. Kavitha, 2016)

Cosine similarity berfungsi untuk membandingkan kemiripan antar dokumen, dalam hal ini yang dibandingkan adalah *query* dengan dokumen latih. Dalam menghitung *cosine similarity*, pertama yang dilakukan yaitu melakukan perkalian skalar antara *query* dengan dokumen kemudian dijumlahkan, setelah itu melakukan perkalian antara panjang dokumen dengan panjang *query* yang telah dikuadratkan, setelah itu dihitung akar pangkat dua. Selanjutnya hasil perkalian skalar tersebut dibagi dengan hasil perkalian panjang dokumen dan *query*. (Andi Y, 2020)

Rumus dapat dilihat sebagai berikut.

$$\text{cosSim}(d_j, q_k) = \frac{\sum_{i=1}^n (td_{ij} \times tq_{ik})}{\sqrt{\sum_{i=1}^n td_{ij}^2} \times \sqrt{\sum_{i=1}^n tq_{ik}^2}}$$

Keterangan:

$\text{cosSim}(d_j, q_k)$: tingkat kesamaan dokumen dengan query tertentu
td_{ij}	: term ke- i dalam vektor untuk dokumen ke- j
tq_{ik}	: term ke- i dalam vektor untuk query ke- k
n	: jumlah term yang unik dalam data set

Gambar 2.3 Rumus cosSim

2.2 Penelitian Terkait

Pada laporan ini penyusun banyak mendapatkan referensi dari penelitian-penelitian sebelumnya yang berkaitan dengan tugas laporan akhir *Project-Based Learning* mata kuliah Algoritma dan pemrograman lanjut :

2.2.1 “Automatic Text Summarization Menggunakan Metode Graph dan Ant Colony Optimization” (I Wayan Adi, Duman Care Krishne, dan I Made Arsa, 2018)

Pada penelitian ini penulis dapat melakukan analisa pada setiap titik-titik yang digambarkan oleh *Graph*. Semut akan melakukan rute perjalanan untuk mendapatkan hasil nilai terbaik. Semut terbaik nantinya akan membangun rute perjalanan hingga mendapatkan rute terbaik. *Graph* dibangun dengan menggunakan bobot *edge*. Untuk mencari bobot dari setiap simpul pada *Graph* digunakan empat fitur dokumen yaitu kemiripan antar-kalimat (f_1), kalimat yang menyerupai judul dokumen (f_2), TF-ISF (f_3) dan TF-IDF (f_4). Fitur-fitur ini telah banyak digunakan dalam penelitian sebelumnya dan sudah dibuktikan mampu mengambil fitur dari kalimat. Dengan penggabungan metode ini diharapkan mampu menghasilkan hasil ringkasan yang baik tanpa mengurangi informasi-informasi yang terdapat pada teks asli.

Berdasarkan hasil dan pembahasan yang dilakukan tentang *Automatic Text Summarization* Menggunakan Metode *Graph* dan Metode *Ant Colony Optimization* maka dapat disimpulkan beberapa hal sebagai berikut :

1. Rancangan sistem *automatic text summarization* bisa dibangun dengan pendekatan optimasi menggunakan metode *ant colony optimization*. Agar metode *ant colony optimization* bisa bekerja kalimat yang akan diringkas ditransformasi menjadi *graph* yang memiliki bobot pada sisi setiap simpulnya. Nilai bobot ini didapat dengan mengekstrak fitur kalimat dalam dokumen. Dalam penelitian ini fitur kalimat didapat menggunakan fitur f_1-f_4 .

2. Pengujian hasil ringkasan dengan mencari kesamaan hasil ringkasan sistem dengan hasil ringkasan secara manual menggunakan

cosine similarity memperoleh persentase kesamaan 76.3%. Pengujian dengan *autosummary tools* pada *Microsoft Word* memperoleh hasil ringkasan rata-rata memiliki kesamaan 68.15%. Hasil ringkasan sistem dengan hasil ringkasan ahli memiliki kesamaan 78.43%. Hal ini berarti lebih dari 75% informasi yang dianggap penting oleh manusia sudah dapat di temukan oleh sistem.

3. *Compression rate* ringkasan sistem mencapai 78.2%. Artinya informasi dalam dokumen berhasil diringkas. menyisakan 21.80% informasi yang telah dapat mewakili isi seluruh dokumen.

2.2.2 “Implementasi Metode Recurrent Neural Network Pada Text Summarization Dengan Teknik Abstraktif” (Kasyfi Ivanedra dan Metty Mustikasari, 2019)

Peneliti membuat peringkas teks otomatis dengan menggunakan teknik Abstraktif yang dapat meringkas teks lebih natural seperti ringkasan teks yang dibuat oleh manusia. Penelitian ini menguji performa sistem menggunakan *Precision*, *Recall*, dan *F-Measure* dan kemudian membandingkan hasil ringkasan yang dihasilkan oleh sistem dan ringkasan yang dibuat oleh manusia.

Dataset yang digunakan adalah data artikel berita dengan jumlah total artikel sebanyak 4515 buah artikel. Pengujian dibagi berdasarkan data dengan menggunakan *Stemming* dan dengan teknik *Non-stemming*. Nilai rata-rata *recall* artikel berita *non stemming* adalah sebesar 41%, *precision* sebesar 81%, dan *F-measure* sebesar 54,27%. Sedangkan nilai rata-rata *recall* artikel berita dengan teknik *stemming* sebesar 44%, *precision* sebesar 88%, dan *F-measure* sebesar 58,20 %. Perbedaan ini disebabkan oleh pengaruh jumlah dataset yang digunakan sebagai bahan training. Karena pembendaharaan kata yang lebih banyak tentunya menjadikan program lebih mengerti dan lebih akurat dalam menciptakan ringkasan.

2.2.3 “Latent Semantic Analysis (LSA) dan Automatic Text Summarization (ATS) dalam Optimasi Pencarian Artikel Covid-19” (Herlina, Ruth Damayanti, dan Juwairiah, 2020)

Pada penelitian ini membahas tentang proses untuk melakukan *information extraction* dari *frontier* terkait menggunakan metode *scrapping*. Proses ini bisa digambarkan seperti kegiatan menyalin dan menempel informasi, namun dilakukan secara otomatis. Teknik untuk melakukan *scraping* ada beberapa cara dengan cara *HTTP programming*, *DOM parsing*, dan *HTML parser* (Vargiu & Urru, 2012). Pada penelitian ini digunakan dengan proses *HTML parser*, dimana sistem akan mendeteksi setiap element HTML yang ada, dan akan mengambil element yang mengandung berita didalamnya. Setiap *website* akan

memiliki struktur tag HTML yang berbeda, sehingga saat *fetching page*, diperlukan untuk *filtering tag HTML* dan melakukan penyaringan untuk tag HTML yang dibutuhkan saja.

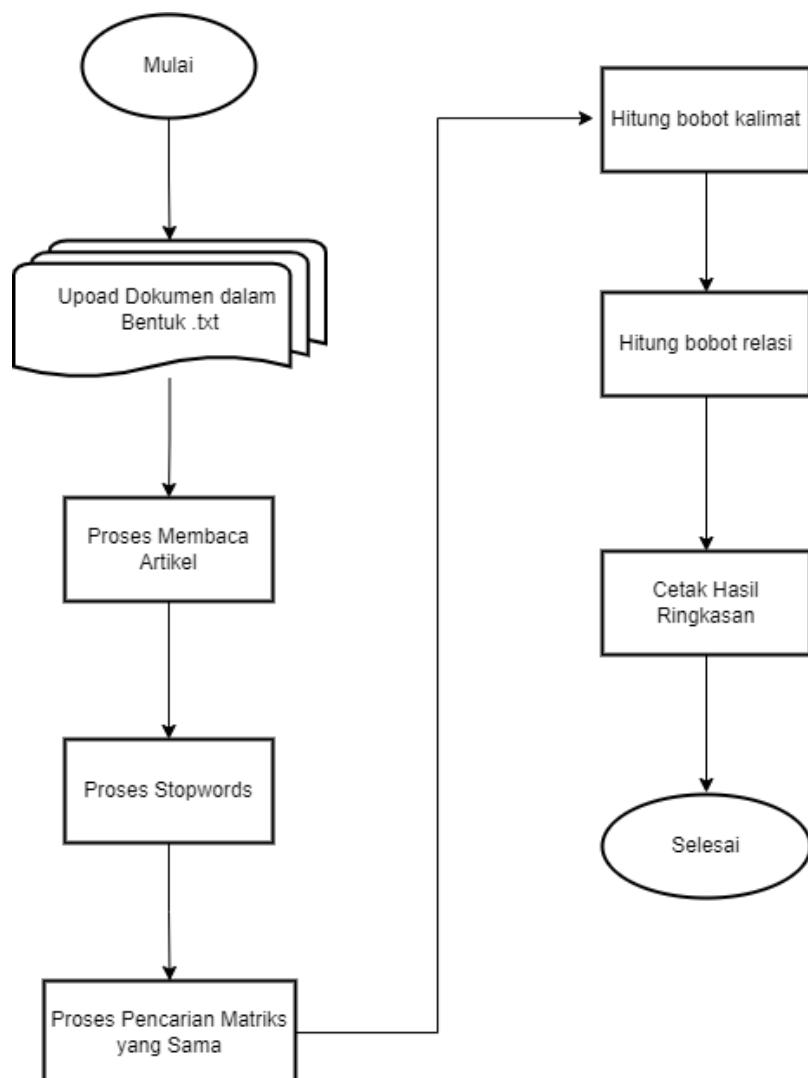
Adapun kesimpulan dari penelitian ini adalah Sistem yang dibuat mampu melakukan optimasi pencarian artikel *Covid-19* dari situs berita Kompas.com, Liputan6.com, Klikdokter.com, detikhealth.com dengan peringkasan teks otomatis menggunakan metode *latent semantic analysis*. Hasil pengujian menghasilkan nilai *f-measure* dan *recall* rata-rata sebesar 90.68% dan 85% dengan persentase data latih : data uji adalah 90 : 10. Pengumpulan data dilakukan selama bulan Februari-Juni 2020 diambil 200 dokumen latih, dan 20 dokumen uji. Pengujian dilakukan dengan *compression rate* sebesar 30%.

BAB III: METODOLOGI PENELITIAN

Pada bab ini akan menjelaskan tentang tahap atau metode yang digunakan dalam penelitian pembuatan peringkasan teks secara otomatis menggunakan metode ekstraktif dimana berfungsi untuk menemukan kesamaan di antara semua kalimat. Dengan menggunakan *Cosine Similarity* sebagai matriks kesamaan dan algoritma *TextRank* untuk menentukan peringkat kalimat berdasarkan kepentingannya.

3.1 Desain Sistem

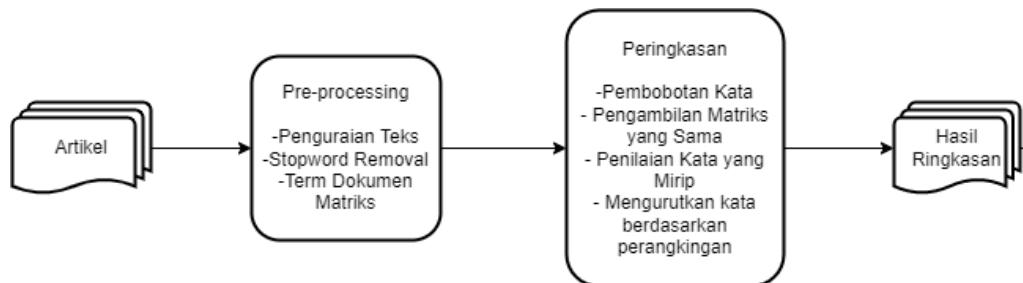
Pada subbab ini pembahasan tentang unsur-unsur yang berkaitan dengan penelitian peringkasan teks otomatis seperti *downloader*, ekstraksi, peringkasan serta semua yang diperlukan dalam proses membuat ringkasan yang sesuai dengan isi teks. Teknik ekstraksi hanya menyalin informasi yang dianggap paling penting oleh sistem untuk ringkasan (misalnya, klausa kunci, kalimat atau paragraf).



Gambar 3.1 Diagram Alir (flowchart) Text Summarization

3.2 Arsitektur Sistem

Arsitektur peringkasan artikel Bahasa Indonesia yang digunakan untuk menggambarkan sistem kerja pada penelitian ini terdiri dari tiga tahapan yaitu, *preprocessing*, peringkasan dan metode pengujian seperti pada Gambar dibawah ini :



Gambar 3.2 Arsitektur Sistem.

3.2.1 Penjelasan Arsitektur Sistem

Berikut penjelasan arsitektur sistem :

1. Pengambilan data pada penelitian kami yaitu menggunakan teks artikel yang diambil dari internet/web. Kemudian nanti akan diubah menjadi file .txt terlebih dahulu. Lalu dimasukkan ke dalam *script* yang sudah kami buat.

```
# Untuk membaca artikel
class Summarization:
    def read_article(file_name):
        file = open(file_name, "r")
        filedata = file.readlines() # Untuk membaca tiap barisnya
        article = filedata[0].split(".") # untuk membatasi tiap kalimat
        sentences = []
        for sentence in article:
            sentences.append(sentence.replace("[^a-zA-Z]", " ").split(
                " ")) # untuk menambahkan kalimat dan merubah spasi tiap antarkata menjadi batas pemisah kata
        sentences.pop()
        return sentences
```

Gambar 3.3 Membaca Article

2. Tahap *pre-processing* adalah tahap membaca teks artikel dengan proses menambahkan kalimat dan mengubah spasi tiap antar kata menjadi batas pemisah kata.

```
# Untuk membandingkan kata-kata yang sama
def sentence_similarity(senti, sent2, stopwords=None):
    if stopwords is None:
        stopwords = [] # menghapus kata-kata yg kurang bermakna
    sent1 = [w.lower() for w in sent1]
    sent2 = [w.lower() for w in sent2]
    all_words = list(set(sent1 + sent2))
    vector1 = [0] * len(all_words)
    vector2 = [0] * len(all_words)
    for w in sent1:
        if w in stopwords:
            continue
        vector2[all_words.index(w)] += 1
    return 1 - cosine_distance(vector1, vector2) # untuk membandingkan perbedaan antara vector 1 dan vector 2
```

Gambar 3.4 Melakukan pre-processing

3. Pada tahap peringkasan adalah tahap memproses teks yang telah dibaca oleh *script* yang sudah dibuat. Dengan melewati tahap pembobotan kata, pengambilan matriks yang sama, penilaian kata yang mirip dan mengurutkan kata berdasarkan perangkingan.
4. Hasil ringkasan yang dihasilkan berupa teks yang sudah diringkas otomatis oleh *script*.

3.3 Proses Penelitian

Pada bagian ini akan digambarkan alur secara rinci dari penelitian yang akan dilakukan penulis, yaitu dimulai dari pencarian teks artikel dari media *online*, proses dari metode yang digunakan hingga menghasilkan sebuah ringkasan.

3.3.1 Tahap *Preprocess* :

Proses peringkasan teks otomatis dokumen Bahasa Indonesia dimulai dengan membaca isi dokumen, dan rasio ringkasan. Masukan isi dokumen berupa teks. Sedangkan, masukan rasio ringkasan berupa angka persentase dari jumlah kalimat ringkasan yang diinginkan.

a. *Tokenizing and Filtering*

Tokenizing dan *Filtering* menghilangkan tanda baca yang tidak diperlukan dan dilakukan dengan memisahkan kalimat menjadi kata-kata. Kalimat dipisah menjadi kata-kata dengan pemisah karakter spasi. Setiap kata-kata di kalimat kemudian dibandingkan apakah tidak termasuk pada kumpulan tanda baca. Jika tidak termasuk pada kumpulan tanda baca, maka kata-kata masuk dalam hasil *preprocess*. Tanda baca diambil dari modul standar *library python string punctuation*. Berikutnya mengubah semua token ke bentuk huruf kecil (*lower case*).

b. *Stopwords and numbers*

Removal Stopwords merupakan proses penghilangan kata tidak penting pada deskripsi melalui pengecekan kata-kata hasil parsing deskripsi apakah termasuk di dalam daftar kata tidak penting (*stoplist*) atau tidak. Angka dalam bentuk kata tidak dimasukkan dalam perhitungan karena dapat menimbulkan multitafsir. Kamus stopword bahasa indonesia diperoleh dari modul python bernama *stop_words*.

Perlunya langkah-langkah diatas agar dapat menentukan kata-kata yang mengandung makna ambigu serta mempercepat proses perhitungan nanti. Lebih jelas akan ditunjukkan melalui tabel dibawah :

Tahapan	Input	Output	Penjelasan
Tokenizing	Pasokan minyak goreng di sejumlah daerah mulai membanjiri pasar setelah pemerintah	['Pasokan', 'minyak', 'goreng', 'di', 'sejumlah', 'daerah', 'mulai', 'membanjiri', 'pasar', 'setelah', 'pemerintah']	Memisahkan kata-kata sebagai struktur kalimat.
Remove stop words	Pasokan minyak goreng di sejumlah daerah mulai membanjiri pasar setelah pemerintah menaikkan harga eceran tertinggi (HET).	[Minyak goreng sejumlah daerah mulai membanjiri pasar setelah pemerintah menaikkan harga.]	Menghilangkan kata (di, HET, Pasokan).

Tabel 1. Stopwords and Numbers

3.3.2 Main Processing :

Tahap *Main Process* adalah representasi teks menjadi graf, mengidentifikasi hubungan antarkalimat, pembobotan tiap kalimat berdasarkan identifikasi hubungan antarkalimat dan perangkingan kalimat berdasarkan bobot yang telah dihitung. Representasi dilakukan sebagai berikut:

1. Proses peringkasan teks otomatis dokumen dimulai dengan membaca judul, isi dokumen, dan rasio ringkasan. Masukan judul dan isi dokumen berupa teks. Sedangkan, masukan rasio ringkasan berupa angka persentase dari jumlah kalimat ringkasan yang diinginkan. Setelah membaca masukan judul, isi dokumen dan rasio ringkasan, dilakukan proses *text preprocessing* yang *Text preprocessing* dimulai dengan *tokenization* kalimat dan kata dari judul dan isi dokumen menjadi kata-kata (token).
2. Identifikasi hubungan antarkalimat dengan membuat sisi antara simpul-simpul dengan menggunakan *cosine similarity*. *Cosine similarity* merupakan fungsi yang menerima dua buah objek atau lebih dan mengembalikan nilai kemiripan (*similarity*) antara kedua objek tersebut berupa bilangan riil. *Cosine Similarity* mengukur dua objek vektor yang masing-masing memiliki n dimensi dengan menemukan nilai *cosinus* dari sudut yang dibentuk dua vektor tersebut. Umumnya, nilai yang dihasilkan oleh fungsi *similarity* berkisar pada interval [0...1].

3. Memberi skor awal simpul berupa nilai acak untuk menentukan iterasi.
4. Melakukan iterasi algoritma *TextRank* sampai *error rate* tiap simpul konvergen atau mencapai ambang batas di bawah *threshold*. *Error Rate* disini adalah perbedaan antar dua skor simpul yang dihitung pada iterasi yang berurutan dengan rumus x.
5. Kemudian simpul diurutkan berdasarkan skor akhirnya. Lalu diambil kalimat-kalimat dari ranking teratas sebanyak 50% sebagai hasil ekstraksi ringkasannya. Proses ini diulangi hingga tiga kali hingga 12,5%.
6. Selanjutnya adalah proses perhitungan total skor fitur per kalimat. Fitur-fitur yang sudah dihitung sebelumnya akan dikalikan dengan bobot tiap fitur yang sudah didapat dari proses training dengan membangkitkan secara acak kemudian dijumlahkan menjadi total skor. Total skor akan diurutkan berdasarkan nilai tertinggi hingga terendah. Setelah mendapatkan urutan total skor, dilanjutkan dengan pemotongan jumlah kalimat dengan cara mengalikan rasio ringkasan sistem dengan jumlah kalimat. Kemudian sistem akan menampilkan hasil ringkasan.

BAB IV: HASIL DAN PEMBAHASAN

4.1 Hasil Proyek

Berikut ini adalah hasil ringkasan teks artikel:

```
== HASIL SUMMARIZATION ==  
Tingginya harga minyak goreng di daerah juga diakui warga lainnya. Susi, warga Kecamatan Marpoyan mengatakan dirinya terkejut saat membeli minyak goreng di ritel modern yang tiba-tiba penuh di etalase. Seorang ibu rumah tangga Erita menyampaikan minyak goreng kemasan merek Shania dan Fortune, masing-masing dibanderol seharga Rp49 ribu dan Rp50 ribu untuk ukuran dua liter. Senada, di Kota Tanjungpinang, Provinsi Kepulauan Riau, harga minyak goreng kemasan dibanderol Rp25 ribu per liter.
```

Berikut ada 9 teks percobaan lainnya :

- 1) File teks “bola.txt”

```
Summerization.generate_summary("bola.txt", 4)  
C:\Users\LENOVO  
  
== HASIL SUMMARIZATION ==  
Tien Linh mencetak gol pembuka Vietnam saat membantai Indonesia 3-0 pada laga pertama Grup A SEA Games yang berlangsung di Stadion Viet Tri, Phu Tho, Vietnam, Jumat (6/5). Sebagian besar pendukung Indonesia tidak puas dengan sikap Tien Linh saat pertandingan," tulis Zing News. Pertama saya ucapan selamat kepada Vietnam tetapi gol pertama sudah pasti offside tapi wasit mengabaikannya," ujar Shin Tae Yong. Pelatih Timnas Indonesia U-23 Shin Tae Yong menyoroti gol pertama Vietnam yang dicetak Tien Linh karena offside
```

Gambar 4.1 percobaan I

- 2) File teks “cedera.txt”

```
Summerization.generate_summary("cedera.txt", 4)  
C:\Users\LENOVO  
  
== HASIL SUMMARIZATION ==  
Timnas Indonesia U-23 akan melakoni laga kedua di Grup A melawan Timor Leste pada 10 Mei mendatang. Mantan pemain PSM Makassar itu memperlihatkan foto dirinya sedang memegang paspor. Kehadiran pemain berusia 22 itu diyakini akan meningkatkan kualitas permainan Timnas Indonesia U-23 yang baru saja takluk dengan skor telak 0-3 dari tuan rumah Vietnam. Kabar baiknya Asnawi tidak mengalami cedera yang sempat dikawatirkan banyak pihak
```

Gambar 4.2 Percobaan 2

- 3) File teks “running man.txt”

```
Summerization.generate_summary("running man.txt", 4)  
C:\Users\LENOVO  
  
== HASIL SUMMARIZATION ==  
Yoo Jae Seok terungkap kembali terluka saat syuting program SBS "Running Man". Yoo Jae Seok mengatakan, "Orang-orang mungkin bertanya jenis luka apa ini saat menonton ini. Sebab itu berarti dia sangat sering terluka saat syuting "Running Man" hingga menganggap lukanya sebagai sesuatu yang normal."Aku harap Yoo Jae Seok tidak terluka," ujar seorang netizen. Mungkin ada orang yang bertanya-tanya mengapa aku terluka dengan meletakkan lingkaran ini di sini
```

Gambar 4.3 Percobaan 3

4) File teks “hujan.txt”

```
Summerization.generate_summary("hujan.txt", 4)
C:\Users\LENOVO

== HASIL SUMMARIZATION ==
Suhu maksimum tertinggi di Indonesia pada bulan April selama 4-5 tahun terakhir sekitar 38,8 derajat Celcius di Palu lembang pada tahun 2019, sedangkan di bulan Mei sekitar 38,8 derajat Celcius di Temindung Samarinda pada tahun 2018. Selain itu, dominasi cuaca yang cerah dan tingkat awan yang rendah tersebut dapat mengoptimalkan penerimaan sinar matahari di permukaan Bumi. Sehingga menyebabkan kondisi suhu yang dirasakan oleh masyarakat menjadi cukup terik pada siang hari. Saat itu tingkat pertumbuhan awan dan fenomena hujan akan sangat berkurang, sehingga cuaca cerah pada pagi menjelang siang hari akan cukup mendominasi
```

Gambar 4.4 Percobaan 4

5) File teks “jokowi.txt”

```
Summerization.generate_summary("jokowi.txt", 4)
C:\Users\LENOVO

== HASIL SUMMARIZATION ==
Sementara itu, Menteri Luar Negeri AS Antony Blinken mengungkapkan sejumlah isu yang akan dibahas dalam KTT ini. Pada Februari 2022 lalu, juru bicara Gedung Putih Jen Psaki menyatakan KTT ini merupakan "prioritas utama bagi pemerintahan Biden." Pertemuan ini juga akan menjadi perayaan 45 tahun terjalannya hubungan antara AS dan ASEAN. Namun, pertemuan yang bakal dihadiri Presiden Amerika Serikat Joe Biden dan para pemimpin ASEAN itu ditunda. Kabar kunjungan Jokowi itu disampaikan melalui akun sosial media Twitter resmi Kementerian Luar Negeri RI pada Minggu (8/5)
```

Gambar 4.5 Percobaan 5

6) File teks “reog.txt”

```
Summerization.generate_summary("reog.txt", 4)
C:\Users\LENOVO

== HASIL SUMMARIZATION ==
Siswandi mengenang, Malaysia pernah merebut dua pulau, Sipadan dan Ligatan. Sejumlah seniman di Surabaya menggalang dukungan agar pemerintah memprioritaskan kesenian tradisional Reog untuk didaftarkan ke Organisasi Pendidikan, Kependidikan dan Kebudayaan Perserikatan Bangsa-bangsa (UNESCO) sebagai warisan budaya tak benda. Saya tidak mau kebudayaan kita diklaim oleh Malaysia. Para seniman khawatir, jika tidak segera didaftarkan, kesenian reog akan dicaplok menjadi milik negara tetangga Malaysia
```

Gambar 4.6 Percobaan 6

7) File teks “saham.txt”

```
Summerization.generate_summary("saham.txt", 4)
C:\Users\LENOVO

== HASIL SUMMARIZATION ==
Setelah pasar ditutup, nilai kekayaan bersih Michael Bambang Hartono sebesar US$ 21,9 miliar atau Rp 317,5 triliun. Setelah pasar ditutup kekayaan pria yang akrab disapa CT ini tergerus 6,08% atau US$ 519 juta, setara Rp 7,52 triliun. Saat ini nilai kekayaan bersih Bambang Hartono sebesar US$ 21,0 miliar atau setara Rp 304,5 triliun. Pemilik CT Corp, Chairul Tanjung juga tak luput dari penurunan nilai kekayaan
```

Gambar 4.7 Percobaan 7

8) File teks “sea games.txt”

```
Summerization.generate_summary("sea_games.txt", 4)
C:\Users\LENOVO

== HASIL SUMMARIZATION ==
Untuk pertama kalinya saya tidak puas dengan ini," kata Shin Tae Yong. Timnas Indonesia U-23 masih berada di papan bawah tanpa poin. Selanjutnya, Skuad Garuda Muda akan kembali berlaga di partai kedua menghadapi Timor Leste, Selasa (10/5) pukul 18.15 WIB. Sebab di pertandingan lain, Filipina mengalahkan Timor Leste dengan skor 4-0
```

Gambar 4.8 Percobaan 8

9) File teks “wabah.txt”

```
Summerization.generate_summary("wabah.txt", 4)
C:\Users\LENOVO

== HASIL SUMMARIZATION ==
Penduduk Beijing khawatir mereka akan memiliki nasib serupa dengan 25 juta orang Shanghai di rumah selama beberapa minggu. Para pejabat di sana mengatakan kota pembangkit tenaga listrik di timur itu memenangkan pertempurannya melawan wabah terburuk di China sejak pandemi dimulai. Namun penguncian Shanghai telah meningkat, menyebabkan kemarahan dan protes yang jarang terjadi di ekonomi besar terakhir yang masih terpaku pada kebijakan nol-Covid. Jutaan orang di Beijing, China masih harus menjalani pembatasan lockdown untuk mencegah wabah Covid-19
```

Gambar 4.9 Percobaan 9

4.2 Pembahasan

Berikut ini adalah kode yang kami gunakan untuk meringkas teks secara otomatis menggunakan bahasa *Python*.

```
import nltk

import os
print(os.getcwd() + "\n")

from tkinter import *
from tkinter import ttk
from tkinter import filedialog as fd
import tkinter.messagebox

from nltk import tokenize
from nltk.corpus import stopwords
from nltk.cluster.util import cosine_distance
import numpy as np
import networkx as nx
from pathlib import Path

# Untuk membaca artikel
class Summerization:
    def read_article(file_name):
        file = open(file_name, "r")
        filedata = file.readlines() # Untuk membaca tiap barisnya
        article = filedata[0].split(". ") # untuk membatasi tiap kalimat
        sentences = []
```

```

        for sentence in article:
            sentences.append(sentence.replace("[^a-zA-Z]", " ").split(
                " ")) # untuk menambahkan kalimat dan merubah spasi tiap
antarkata menjadi batas pemisah kata
            sentences.pop()
        return sentences

    # untuk menghasilkan matriks kesamaan
    def gen_sim_matrix(sentences, stop_words): # similarity matrix
        similarity_matrix = np.zeros((len(sentences), len(sentences))) # untuk menunjukkan kesamaan kalimat
        for idx1 in range(len(sentences)):
            for idx2 in range(len(sentences)):
                if idx1 == idx2:
                    continue
                similarity_matrix[idx1][idx2] = sentence_similarity(
                    sentences[idx1], sentences[idx2], stop_words) # stopword merupakan kamus dari nltk
        return similarity_matrix

    # untuk meringkas teks
    def generate_summary(file_name, top_n):
        stop_words = stopwords.words('indonesian')
        summarize_text = []
        sentences = Summerization.read_article(file_name) # untuk membaca file yang diinput
        sentence_similarity_matrix =
Summerization.gen_sim_matrix(sentences, stop_words) # untuk mengambil matrix dari fungsi gen similarity matrix
        sentence_similarity_graph =
nx.from_numpy_array(sentence_similarity_matrix)
        scores = nx.pagerank(sentence_similarity_graph) # untuk penilaian kata yang mirip
        ranked_sentence = sorted( # mengurutkan kata berdasarkan perangkingan
            ((s) for i, s in enumerate(sentences)), reverse=True)
        for i in range(top_n):
            summarize_text.append(
                " ".join(ranked_sentence[i])) # merangkum teks yg telah dirangking dan membuang kata yang tidak diperlukan

        # mengeluarkan hasil summary
        return ". ".join(summarize_text, )

```

1) Script untuk User Interface :

```
# Root window
root = Tk()
root.title('Text Summarization')
root.geometry('600x500')

# Text editor ( Input Open File )
text = Text(root, height=10)
text.grid(column=0, row=0, sticky='nsew')

def open_text_file():
    # file type
    filetypes = (
        ('text files', '*.txt'),
        ('All files', '.')
    )
    # show the open file dialog
    f = fd.askopenfile(filetypes=filetypes)
    # read the text file and show its content on the Text
    text.insert('1.0', f.readlines())

# open file button
open_button = ttk.Button(
    root,
    text='Open a File',
    command=open_text_file
)
open_button.grid(column=0, row=1, sticky='w', padx=10, pady=10)

# Input top_n
labeltop_n = Label(root, text = 'Masukkan Jumlah Kalimat Yang Ingin
Diringkas').grid(column=0, row=2)

inputtop_n = Text(root, height=2)
inputtop_n.grid(column=0, row=3)

# Output
labelname = Label(root, text = "Hasil Ringkasan:").grid(column=0, row=4)

ringkasan_text = Text(root, height=10)
ringkasan_text.grid(column=0, row=5, sticky='nsew')

def Summerization_Process():
    top_n = int(inputtop_n.get())
    open_article = str(text.insert())
    read_article = Summerization.read_article(open_article)
    gen_sim_matrix = Summerization.gen_sim_matrix(sentences, stop_words)
    generate_summary = Summerization.generate_summary(file_name, top_n)
```

```

ringkasan_text.input(END, generate_summary)

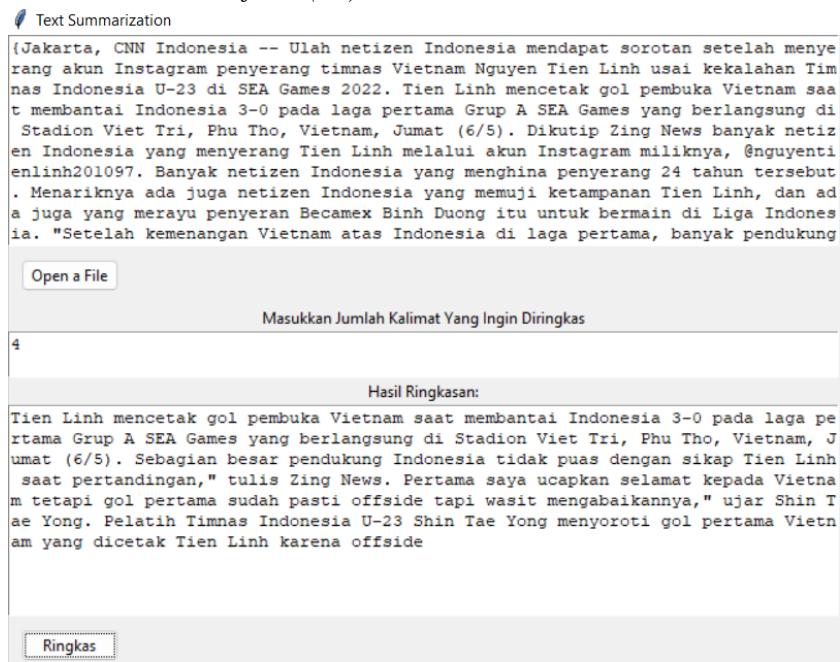
ringkas_button = ttk.Button(root, text='Ringkas',
command=Summerization_Process).grid(column=0, row=6, sticky='W', padx=10,
pady=10)

root.mainloop()

Summerization.generate_summary("cedera.txt", 4)

```

2) Penjelasan dari *User Interface (UI)*:



Gambar 4.10 Tampilan text setelah dirangkum menggunakan UI

Langkah-langkah untuk menggunakan *UI Text Summarization* :

- **Open a file :**
Digunakan untuk memilih file .txt yang akan dirangkum atau bisa juga memilih untuk file lainnya.
- **Jumlah kata :**
Digunakan untuk memasukkan hasil yang diinginkan menjadi berapa kalimat. Contohnya bisa 4 kalimat atau lebih.
- **Ringkas :**
Lalu yang terakhir klik tombol ‘ringkas’ dan akan keluar hasil rangkuman/ringkasan dari text tersebut.

3) Penjelasan dari *Executable (EXE)*:

1. Langkah pertama buka CMD (*Command Prompt Anaconda*).
2. Lalu masukan/ketik **pip install auto-py-to-exe**.

- Setelah itu tekan *enter* lalu tunggu sampai proses *install* atau *download* selesai.

```
C:\Windows\system32\cmd.exe - auto-py-to-exe
Microsoft Windows [Version 10.0.22000.739]
(c) Microsoft Corporation. All rights reserved.

(base) C:\Users\pelan> pip install auto-py-to-exe
Requirement already satisfied: auto-py-to-exe in c:\users\pelan\anaconda3\lib\site-packages (2.20.1)
Requirement already satisfied: pyinstaller>=4.6 in c:\users\pelan\anaconda3\lib\site-packages (from auto-py-to-exe) (5.1)
Requirement already satisfied: Eel==0.14.0 in c:\users\pelan\anaconda3\lib\site-packages (from auto-py-to-exe) (0.14.0)
Requirement already satisfied: bottle-websocket in c:\users\pelan\anaconda3\lib\site-packages (from Eel==0.14.0->auto-py-to-exe) (0.2.9)
Requirement already satisfied: whichcraft in c:\users\pelan\anaconda3\lib\site-packages (from Eel==0.14.0->auto-py-to-exe) (0.6.1)
Requirement already satisfied: future in c:\users\pelan\anaconda3\lib\site-packages (from Eel==0.14.0->auto-py-to-exe) (0.18.2)
Requirement already satisfied: pyparsing in c:\users\pelan\anaconda3\lib\site-packages (from Eel==0.14.0->auto-py-to-exe) (3.0.4)
Requirement already satisfied: bottle in c:\users\pelan\anaconda3\lib\site-packages (from Eel==0.14.0->auto-py-to-exe) (0.12.21)
Requirement already satisfied: altgraph in c:\users\pelan\anaconda3\lib\site-packages (from pyinstaller>=4.6->auto-py-to-exe) (0.17.2)
Requirement already satisfied: pyinstaller-hooks-contrib>=2021.4 in c:\users\pelan\anaconda3\lib\site-packages (from pyinstaller>=4.6->auto-py-to-exe) (2022.7)
Requirement already satisfied: pywin32-ctypes>=0.2.0 in c:\users\pelan\anaconda3\lib\site-packages (from pyinstaller>=4.6->auto-py-to-exe) (0.2.0)
Requirement already satisfied: setuptools in c:\users\pelan\anaconda3\lib\site-packages (from pyinstaller>=4.6->auto-py-to-exe) (58.0.4)
Requirement already satisfied: pefile>=2017.8.1 in c:\users\pelan\anaconda3\lib\site-packages (from pyinstaller>=4.6->auto-py-to-exe) (2022.5.30)
Requirement already satisfied: gevent in c:\users\pelan\anaconda3\lib\site-packages (from bottle-websocket>Eel==0.14.0->auto-py-to-exe) (0.10.1)
Requirement already satisfied: gevent in c:\users\pelan\anaconda3\lib\site-packages (from gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (21.8.0)
Requirement already satisfied: greenlet<2.0,>=1.1.0 in c:\users\pelan\anaconda3\lib\site-packages (from gevent>gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (1.1.1)
Requirement already satisfied: cffi>=1.12.2 in c:\users\pelan\anaconda3\lib\site-packages (from gevent>gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (1.14.6)
Requirement already satisfied: zope.event in c:\users\pelan\anaconda3\lib\site-packages (from gevent>gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (4.5.0)
Requirement already satisfied: zope.interface in c:\users\pelan\anaconda3\lib\site-packages (from gevent>gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (5.4.0)
Requirement already satisfied: pycparser in c:\users\pelan\anaconda3\lib\site-packages (from cffi>=1.12.2>gevent>gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (2.20)
WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the 'C:\Users\pelan\anaconda3\python.exe -m pip install --upgrade pip' command.

(base) C:\Users\pelan> auto-py-to-exe
```

Gambar 4.11 Proses install auto-py-to-exe

- Setelah proses *download/install* selesai, buka *py installer* dengan memasukkan/mengetik **auto-py-to-exe** kemudian menunggu proses membuka *py installer* selesai.

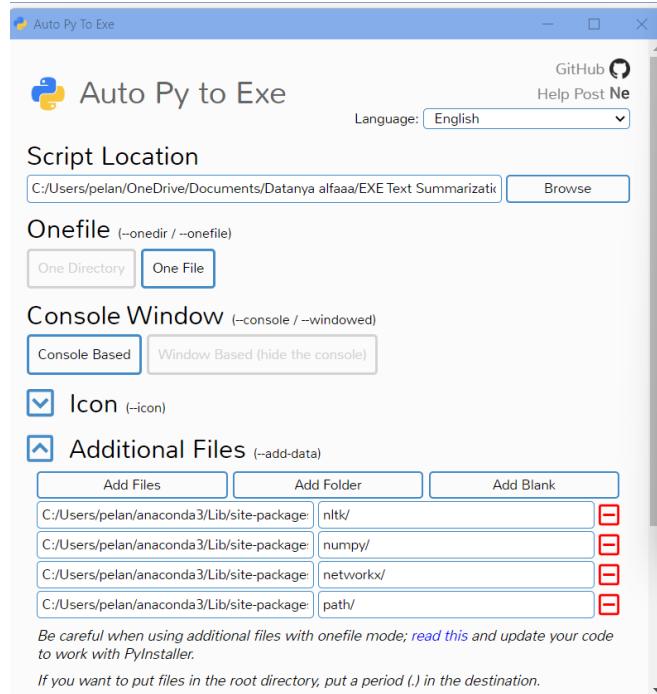
```
C:\Windows\system32\cmd.exe - auto-py-to-exe
Requirement already satisfied: altgraph in c:\users\pelan\anaconda3\lib\site-packages (from pyinstaller>=4.6->auto-py-to-exe) (0.17.2)
Requirement already satisfied: pyinstaller-hooks-contrib>=2021.4 in c:\users\pelan\anaconda3\lib\site-packages (from pyinstaller>=4.6->auto-py-to-exe) (2022.7)
Requirement already satisfied: pywin32-ctypes>=0.2.0 in c:\users\pelan\anaconda3\lib\site-packages (from pyinstaller>=4.6->auto-py-to-exe) (0.2.0)
Requirement already satisfied: setuptools in c:\users\pelan\anaconda3\lib\site-packages (from pyinstaller>=4.6->auto-py-to-exe) (58.0.4)
Requirement already satisfied: pefile>=2017.8.1 in c:\users\pelan\anaconda3\lib\site-packages (from pyinstaller>=4.6->auto-py-to-exe) (2022.5.30)
Requirement already satisfied: gevent in c:\users\pelan\anaconda3\lib\site-packages (from bottle-websocket>Eel==0.14.0->auto-py-to-exe) (0.10.1)
Requirement already satisfied: gevent in c:\users\pelan\anaconda3\lib\site-packages (from gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (21.8.0)
Requirement already satisfied: greenlet<2.0,>=1.1.0 in c:\users\pelan\anaconda3\lib\site-packages (from gevent>gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (1.1.1)
Requirement already satisfied: cffi>=1.12.2 in c:\users\pelan\anaconda3\lib\site-packages (from gevent>gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (1.14.6)
Requirement already satisfied: zope.event in c:\users\pelan\anaconda3\lib\site-packages (from gevent>gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (4.5.0)
Requirement already satisfied: zope.interface in c:\users\pelan\anaconda3\lib\site-packages (from gevent>gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (5.4.0)
Requirement already satisfied: pycparser in c:\users\pelan\anaconda3\lib\site-packages (from cffi>=1.12.2>gevent>gevent-websocket>bottle-websocket>Eel==0.14.0->auto-py-to-exe) (2.20)
WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the 'C:\Users\pelan\anaconda3\python.exe -m pip install --upgrade pip' command.

(base) C:\Users\pelan> auto-py-to-exe
```

Gambar 4.12 Proses membuka py installer

- Setelah proses selesai secara otomatis akan membuka aplikasi *py to exe* seperti gambar dibawah ini.
- Kemudian masukkan *file script* di **Script Location**.
- Lalu pilih **One File** dan **Console Based** untuk window-nya.

8. Setelah itu masukkan *file library* ke **Additional File**. Masukkan semua *file library* jika ada 4 *library* masukkan semua agar nanti tidak *crash* saat membuka *exe*.



Gambar 4.13 Proses memasukkan library

9. Setelah melakukan langkah-langkah tersebut, maka akan terjadi *process convert to exe* seperti berikut ini.

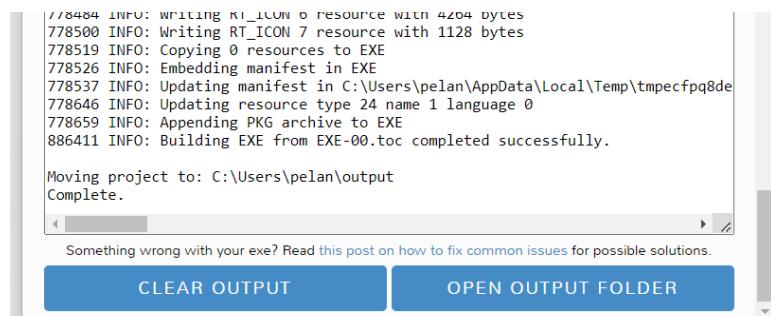
```

Running auto-py-to-exe v2.20.1
Building directory: C:\Users\pelan\AppData\Local\Temp\tmpecfpq8de
Provided command: pyinstaller --noconfirm --onefile --console --add-data "C:/Users/pelan/OneDrive/Documents/Datanya alfaaa/EXE Text Summarizatik"
Recursion Limit is set to 5000
Executing: pyinstaller --noconfirm --onefile --console --add-data C:/Users/pelan/OneDrive/Documents/Datanya alfaaa/EXE Text Summarizatik
207535 INFO: PyInstaller: 5.1
207543 INFO: Python: 3.9.7 (conda)
207563 INFO: Platform: Windows-10-10.0.22000-SP0
207569 INFO: wrote C:/Users/pelan/AppData/Local/Temp/tmpecfpq8de/EXE Text Summarizatik
207583 INFO: UPX is not available.
207605 INFO: Extending PYTHONPATH with paths
['C:\\\\Users\\\\pelan\\\\OneDrive\\\\Documents\\\\Datanya alfaaa']
208885 INFO: Checking Analysis
208890 INFO: Building Analysis because Analysis-00.toc is non existent
208898 INFO: Initializing module dependency graph...
208916 INFO: Caching module graph hooks...
208998 INFO: Analyzing base_library.zip...
218063 INFO: Processing pre-safe module path hook distutils from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\distutils'
218070 INFO: distutils: retargeting to non-venv dir 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\distutils'
228514 INFO: Caching module dependency graph...
229195 INFO: running Analysis Analysis-00.toc
229249 INFO: Adding Microsoft.Windows.Common-Controls to dependent assemblies o
required by C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\distutils
230410 INFO: Analyzing C:\\Users\\pelan\\OneDrive\\Documents\\Datanya alfaaa\\EXE Text Summarizatik
243181 INFO: Processing pre-fini module path hook site from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\site'
243188 INFO: site: retargeting to fake-dir 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\site'
259448 INFO: Processing pre-safe import module hook urlib3.packages.six.moves
269131 INFO: Processing pre-safe import module hook six.moves from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\six.moves'
284690 INFO: Processing pre-safe import module hook win32com from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\win32com'
380852 INFO: Processing module hooks...
380859 INFO: Loading module hook 'hook-appdirs.py' from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\appdirs'
380890 INFO: Loading module hook 'hook-argon2.py' from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\argon2'
380908 INFO: Loading module hook 'hook-bcrypt.py' from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\bcrypt'
380923 INFO: Loading module hook 'hook-bokeh.py' from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\bokeh'
384320 INFO: Loading module hook 'hook-certifi.py' from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\certifi'
384344 INFO: Loading module hook 'hook-cryptography.py' from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\cryptography'
385223 INFO: Loading module hook 'hook-dask.py' from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\dask'
385333 INFO: Loading module hook 'hook-docutils.py' from 'C:\\Users\\pelan\\AppData\\Local\\Temp\\tmpecfpq8de\\EXE Text Summarizatik\\docutils'

```

Gambar 4.14 Proses convert to exe

10. Setelah proses *convert to exe* selesai, tekan *open output folder* untuk membuka *file exe* nya.



```
//8484 INFO: Writing RT_ICON 0 resource with 4204 bytes
778500 INFO: Writing RT_ICON 7 resource with 1128 bytes
778519 INFO: Copying 0 resources to EXE
778526 INFO: Embedding manifest in EXE
778537 INFO: Updating manifest in C:\Users\pelan\AppData\Local\Temp\tmpecfpq8de
778646 INFO: Updating resource type 24 name 1 language 0
778659 INFO: Appending PKG archive to EXE
886411 INFO: Building EXE from EXE-00.toc completed successfully.

Moving project to: C:\Users\pelan\output
Complete.
```

Something wrong with your exe? Read [this post](#) on how to fix common issues for possible solutions.

CLEAR OUTPUT OPEN OUTPUT FOLDER

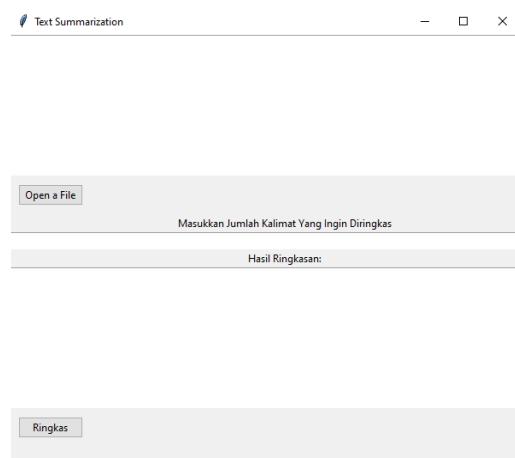
Gambar 4.15 Hasil convert

11. Untuk melihat *file exe* akan otomatis tersimpan pada *folder output* yang ada di laptop berupa *file application* seperti gambar di bawah ini.



Gambar 4.16 Tampilan file type

12. Setelah dibuka, tampilannya akan sama persis seperti tampilan *User Interface* yang sudah dibuat sebelumnya. Untuk langkah-langkahnya sama seperti yang sudah dijelaskan di atas.



Gambar 4.17 Tampilan User Interface

BAB V: KESIMPULAN

Berdasarkan hasil penelitian yang kami lakukan, kesimpulan yang dapat kami ambil adalah metode ekstraktif untuk teks artikel mampu mengurangi dari 431 kata menjadi 56 kata. Algoritma yang digunakan untuk *Text Summarization* bisa berfungsi dengan baik dan dapat menghasilkan ringkasan ekstraktif yang mewakili informasi penting dari suatu artikel dengan lebih cepat.

Persentase keberhasilan projek ini kami dapatkan melalui percobaan 10 teks artikel. Dari 10 percobaan yang telah dilakukan, diperoleh hasil bahwa program mampu meringkas teks dan sesuai dengan hasil ringkasan versi aslinya. Maka persentase keberhasilan yang didapatkan adalah 100%.

Adapun kelebihan dan kekurangan dalam program *Automatic Text Summarization* dengan metode ekstraktif yang sudah kami buat dalam penelitian ini yaitu :

Kelebihan :

- 1) Mampu meringkas teks secara otomatis dan sesuai dengan hasil rangkuman versi *human*.
- 2) Tidak ada batas maksimal kata untuk meringkas teks tersebut.
- 3) Pengguna dapat memperoleh hasil ringkasan dari sistem secara otomatis dengan waktu yang lebih sedikit karena melibatkan proses perhitungan dengan kode diatas, jika dibandingkan dengan merangkum secara manual.
- 4) Program telah di-*convert* dalam bentuk exe, sehingga lebih efektif untuk melakukan ringkasan tanpa harus membuka kode programnya terlebih dahulu.

Kekurangan :

- 1) Teks yang diringkas harus berbentuk *file* txt.
- 2) Teks artikel yang ingin diringkas harus dijadikan satu paragraf terlebih dahulu.
- 3) Program ini bisa dijalankan dan hanya dapat dioperasikan sekali pakai setiap meringkas.
- 4) Untuk bentuk exe masih belum efektif, karena *file* txt harus digabung dahulu dalam satu *folder* yang berisi *file* exe tersebut. Setelah itu program baru dapat meringkas text artikel tersebut.

DAFTAR PUSTAKA

- Adi, S. Care, K. & Arsa, S. (2018). *Automatic Text Summarization Menggunakan Metode Graph dan Ant Colony Optimization*. *Teknologi Elektro*, 7(1), 124-130.
- Atmala, Ridwan. *Text Summarization*. (2018). Diakses pada 15 Maret 2022, dari <https://ridwanatmala.blogspot.com/2018/12/text-summarization.html?m=1>
- Ivanedra, K. & Mustikasari, M. (2019). *Implementasi Metode Recurrent Neural Network pada Text Summarization dengan Teknik Abstraktif*. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIK)*, 6(4), 377-382.
- Jayadianti, H. Damayanti, R. & Juwairiah. (2020). *Latent Semantic Analysis (LSA) dan Automatic Text Summarization (ATS) dalam Optimasi Pencarian Artikel Covid-19 Nasional Informatika*, 52-58. Diakses pada 15 Maret 2022, dari UPN Veteran Yogyakarta.
- M. Adib, Z. Cahyo, C. & Khadijah, F. (2020). *Sistem Automatic Text Summarization Menggunakan Algoritma Textrank*, 111-116. Diakses pada 15 Maret 2022, dari UIN Malang.
https://www.researchgate.net/publication/350095971_SISTEM_AUTOMATIC_TEXT_SUMMARIZATION_MENGGUNAKAN_ALGORITMA_TEXTRANK/link/6050b7e1458515e8344e3c84/download
- Praveen, D. (2018). *Understand Text Summarization and Create Your Own Summarizer in Python*. Diakses pada 28 Maret 2022, dari <https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70>.
- Payah Tidur. *Deteksi Persentase Kemiripan Teks Menggunakan Algoritma Cosine Similarity*. payahtidur.com. Diakses tanggal 26 April 2022, dari <https://payahtidur.com/project/cosine-similarity>.
- Sarah, S. *Pengenalan Kemiripan Teks (Text Similarity) di Python*. (2022). Diakses pada 26 April 2022, dari <https://algoritmaonline.com/kemiripan-teks/>.
- Wirawan, D. “Text Summarization dengan cara Extractive Summarization” Youtube, diunggah oleh Wirawan Dwi Prasetya, 13 juni 2021, <https://www.youtube.com/watch?v=U7hb5oYTIyg>.

LAMPIRAN



Gambar 1. Dokumentasi diskusi kelompok melalui google meet

Gambar 2. Dokumentasi penggerjaan laporan di google docs

The screenshot shows a Visual Studio Code window titled "Text Summarization dengan cara Extractive Summarization". The code editor displays a Python script named "eine.py". The script imports necessary libraries (nltk, numpy, networkx) and defines functions for reading an article from a file and calculating sentence similarity based on stopword removal and cosine distance.

```
C:\Users\USER\Documents\KULIAH\SEMESTER 6\NLP\UAS NLP> eine.py
1 import nltk
2 from nltk import tokenize
3 from nltk.corpus import stopwords
4 from nltk.cluster.util import cosine_distance
5 import numpy as np
6 import networkx as nx
7
8 #untuk membaca artikel
9 def read_article(file_name):
10     file = open(file_name, "r")
11     filedata = file.readlines()
12     article = filedata[0].split(". ")
13     sentences = []
14     for sentence in article:
15         sentences.append(sentence.replace("[^a-zA-Z]", " ").split(" "))
16     sentences.pop()
17     return sentences
18
19 #untuk membandingkan kata-kata yang sama
20 def sentence_similarity(sent1, sent2, stopWords=None):
21     if stopWords is None:
22         stopWords = []
23     stopWords = [w.lower() for w in stopWords]
24     sent1 = [w.lower() for w in sent1]
25     sent2 = [w.lower() for w in sent2]
26     allWords = list(set(sent1+sent2))
27
28     vector1 = [0] * len(allWords)
29     vector2 = [0] * len(allWords)
30     for w in sent1:
31         if w in stopWords:
32             continue
```

Gambar 3. Dokumentasi referensi dari youtube

The screenshot shows a Jupyter Notebook cell titled "Kelompok 2 Algo.ipynb". The cell contains Python code for installing nltk, importing it, printing the current working directory, and defining a function to read an article from a file. The code is run in a Python 3 kernel.

```
[1]: conda install -c anaconda nltk
Note: you may need to restart the kernel to use updated packages.

[2]: import nltk

[3]: import os
print(os.getcwd() + "\n")
C:\Users\acer\tugas

[4]: from nltk import tokenize
from nltk.corpus import stopwords
from nltk.cluster.util import cosine_distance
import numpy as np
import networkx as nx
from pathlib import Path

[79]: #Untuk membaca artikel
def read_article(file_name):
    file = open(file_name, "r")
    filedata = file.readlines() #Untuk membaca tiap barisnya
```

Gambar 4. Dokumentasi script pada jupyterlab