LAPORAN AKHIR PEMROSESAN BAHASA ALAMI

Peringkasan Teks Berita Bahasa Inggris dengan Strategi Ekstraktif menggunakan Algoritma TextRank

Disusun oleh:

12S17005 Kiky Purnamasari Napitupulu

12S17006 Tripheni Simanjuntak

12S17023 Jessycha Royanti Tampubolon

11S4037 - PEMROSESAN BAHASA ALAMI

PROGRAM STUDI SARJANA SISTEM INFORMASI

FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO

INSTITUT TEKNOLOGI DEL

JANUARI 2021

Daftar Isi

Daftar Isi 2
Daftar Gambar 3
Daftar Tabel 4
1. PENDAHULUAN 5
1.1 Latar Belakang 5
1.2 Tujuan 6
1.3 Manfaat 6
1.4 Ruang Lingkup 6
2. ISI 7
2.1 Analisis 7
2.1.2 Analisis Preprocessing 16
2.1.2 Analisis Metode 17
2.2 Desain 19
2.2.1 *Dataset* 19
2.2.2 *Data Preprocessing* 19
2.2.3 Vector Representation of Sentences with GloVe 20
2.2.4 Similarity Matrix Preparation 20
2.2.5 Modelling with TextRank 21
2.2.6 Evaluation and Results 22
2.3 Implementasi 22
2.4 Hasil 27
3. PENUTUP 28
3.1 Pembagian Tugas dan Tanggung Jawab 28
3.2 Kesimpulan 29
3.3 Saran 30
DAFTAR PUSTAKA 31

Daftar Gambar

Gambar 1 Word Frequency 11
Gambar 2 Vocabulary Frequency 16
Gambar 3 Matriks dengan 4 Kalimat 17
Gambar 4 Desain model 19
Gambar 5 Histogram Frekuensi kata 25
Gambar 6 Word Cloud pada berita business 25

Daftar Tabel

Tabel 1Word Frequency.....8
Tabel 2 Vocabulary Frequency.....12
Tabel 3 Contoh proses Case Folding 20

1. PENDAHULUAN

Bab ini berisi penjelasan mengenai latar belakang pengerjaan proyek, tujuan yang ingin dicapai dalam proyek, manfaat yang diperoleh dalam pengerjaan proyek dan ruang lingkup pengerjaan proyek.

1.1 Latar Belakang

Perkembangan teknologi informasi dan komunikasi berdampak menciptakan ledakan informasi seperti bertambahnya jumlah data teks berita dengan cepat dan dalam jumlah besar. Hal ini menyebabkan semua informasi berita dituntut untuk bisa diakses dengan cepat dan tidak butuh banyak waktu untuk dibaca. Teknologi peringkasan teks adalah solusi untuk membantu permasalahan tersebut. Peringkasan teks mengacu pada tugas mengompresi sejumlah besar data teks atau artikel teks yang panjang menjadi bentuk yang lebih ringkas dengan proses pemilihan informasi penting untuk memudahkan pembaca dalam memahami teks [1]. Dengan banyaknya informasi tekstual seperti artikel berita, peringkasan teks penting untuk akses data teks, dimana dapat membantu pembaca melihat konten atau poin utama dalam data teks tanpa harus membaca semua teks. Ringkasan teks dapat memberikan gambaran yang lebih baik kepada pembaca tentang informasi apa yang dikandung dokumen sebelum memutuskan untuk membacanya secara keseluruhan [1].

Terdapat dua strategi dalam peringkasan teks yaitu strategi *extractive summarization* dan strategi *abstractive summarization*. Peringkasan teks dengan strategi *extractive summarization* merupakan strategi peringkasan yang menerapkan fitur linguistik dan statistik dalam membangun kalimat sehingga tidak melakukan perubahan kata pada dokumen hasil ringkasan [2]. Sedangkan, peringkasan teks dengan strategi *abstractive summarization* merupakan strategi peringkasan yang lebih alami dan memiliki komputasi yang lebih sulit karena menerapkan proses *paraphrase* pada seluruh isi dokumen hasil ringkasan [3]. Berdasarkan kedua strategi ini, strategi *extractive* akan menghasilkan ringkasan teks yang lebih kaku dibandingkan dengan strategi peringkasan *abstractive* karena hasil ringkasan pada strategi *extractive* diperoleh berdasarkan frekuensi kemunculan kata pada dokumen asli. Metode yang dapat diterapkan dalam

strategi peringkasan *extractive* adalah seperti *Term Frequency-Inverse Document Frequency (TF-IDF) method* yang memanfaatkan model *bag of words, cluster based method, graph theoretic approach* yang direpresentasikan dalam *undirected graph,* LSA *Method, Machine Learning approach* dengan K Nearest Neighbour dan Naive Bayes, serta metode peringkasan menggunakan *neural network* [4]. Strategi peringkasan *abstractive* dapat dikelompokkan kedalam dua metode yaitu *structure base methods* dan *semantic based methods* [5].

Peringkasan ekstraktif lebih sederhana dan merupakan praktik umum pada penelitian peringkasan teks otomatis saat ini. Salah satu metode yang dapat diterapkan dalam strategi peringkasan *extractive* yaitu *graph theoretic approach* dengan metode TextRank. TextRank adalah teknik perangkingan kalimat berbasis *graph*. Pada TextRank, setiap kalimat dianggap sebagai sebuah *vertex.* Konsepnya adalah semakin tinggi skor sebuah *vertex*, maka semakin penting *vertex* tersebut [7]. TextRank menghasilkan ekstraksi kalimat sebagai ringkasan. Salah satu kelebihan dari algoritma ini, tidak diperlukannya pelatihan menggunakan data *training* pada algoritma yang digunakan.

1.2 Tujuan

Proyek ini bertujuan untuk untuk memberikan kemudahan kepada pembaca dalam mencari informasi secara cepat pada teks berita berbahasa Inggris. Pembaca dapat memperoleh intisari dari sebuah teks tanpa harus membaca keseluruhan isi teks yang ada, sehingga tidak butuh waktu yang cukup lama bagi pembaca untuk membaca dan memahami teks.

1.3 Manfaat

Manfaat dari pengerjaan proyek ini bagi mahasiswa adalah sebagai salah satu syarat lulus mata kuliah pemrosesan bahasa alami.

1.4 Ruang Lingkup

Luaran yang disajikan dari pengerjaan proyek adalah ringkasan masing – masing teks dalam 5 kategori yaitu bisnis, politik, olahraga, teknologi dan hiburan yang dikembangkan dalam bahasa pemrograman python menggunakan tool Jupyter Notebook**.**

2. ISI

Bab ini berisi penjelasan meliputi analisis data dan metode, desain pemrosesan bahasa alami berupa *flowchart* atau diagram alir, implementasi berupa kode program dan cuplikan hasil, dan hasil evaluasi atas implementasi pemrosesan bahasa alami yang dilakukan.

2.1 Analisis

Sub bab ini menjelaskan tentang analisis terhadap dataset yang digunakan dan metode yang diterapkan dalam menghasilkan ringkasan teks.

2.1.1 Analisis Data

Pada penelitian ini, *dataset* yang akan digunakan yaitu BBC New Summary. Data *BBC New Summary* adalah kumpulan data untuk *extractive text summarization* yang memiliki 417 artikel berita sejak tahun 2004 hingga 2005 yang berasal dari BBC. Dalam data, terdapat kumpulan pasangan artikel dan ringkasannya masing-masing dalam bahasa Inggris dengan 5 kategori yaitu bisnis, politik, olahraga, teknologi dan hiburan dengan format file txt. Baris pertama dari teks artikel adalah judulnya masing-masing.

1. Word Frequency

Pada *dataset* dilakukan perhitungan frekuensi kata pada artikel dan ringkasannya asing - masing. Dari perhitungan frekuensi kata, maka diperoleh jumlah masing – masing kata berdasarkan peringkatnya dalam dataset. Tabel 1 berikut menampilkan 50 kata yang frekuensinya paling besar. Dari tabel tersebut, terlihat bahwa frekuensi kemunculan kata mempunyai pola tertentu yang paling sering muncul yaitu kata- kata yang termasuk ke dalam *stopword.* Sesuai dengan *Zipf's Law,* kata *"the"* dengan 65220 kemunculan kata, kata *"to"* di tempat kedua dengan 36037 kemunculan kata, dan diikuti oleh kata *"of"* dengan 28812 kemunculan kata. Selain *stopword,* jenis kata yang paling sering muncul adalah kata ganti yaitu kata ganti orang seperti *“I”, “his”, “they”, “their”*, kata ganti penunjuk seperti *“that””this”,* dan kata ganti penghubung yaitu *“who”, “which”.*

Tabel 1*Word Frequency*

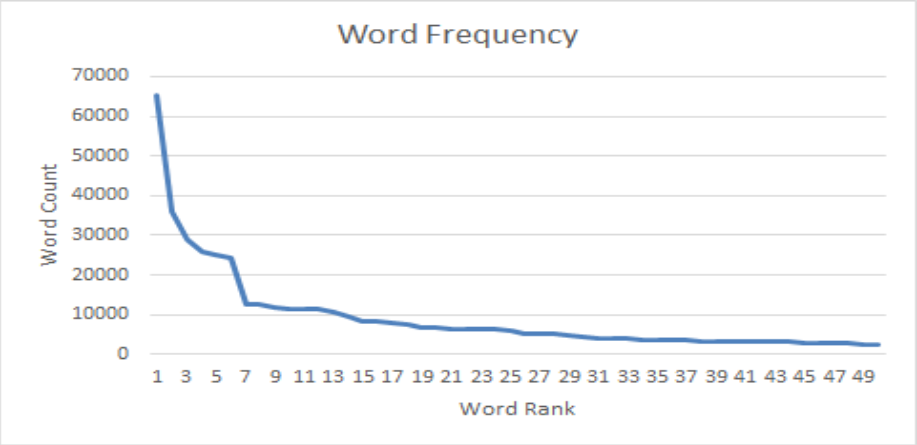
Rank	Count	Word
1	65220	the
2	36037	to
3	28812	of
4	25686	and
5	25034	a
6	24265	in
7	12738	s
8	12710	for
9	11921	is
10	11535	that
11	11528	said
12	11494	The
13	10580	on

14	9646	was
15	8246	it
16	8229	be
17	7761	has
18	7436	with
19	6691	as
20	6670	have
21	6419	by
22	6384	at
23	6283	he
24	6260	will
25	6012	are
26	5302	I
27	5060	from
28	5055	Mr

29	4973	not
30	4238	his
31	3976	would
32	3855	but
33	3837	an
34	3822	had
35	3748	which
36	3631	been
37	3557	they
38	3417	their
39	3328	year
40	3247	its
41	3234	up
42	3221	more
43	3150	were

44	3115	this
45	3013	who
46	3009	also
47	2891	people
48	2772	It
49	2525	about
50	2517	But

Tabel tersebut dapat disajikan dalam bentuk *line chart* seperti gambar berikut.



Gambar 1 *Word Frequency*

2. Vocabulary Frequency

Pada *dataset*, dilakukan perhitungan *frekuensi vocabulary* yaitu jumlah kata yang unik per total kata dalam teks yang ditampilkan pada Tabel 2. Berdasarkan hasil

perhitungan *vocabulary frequency* dengan Heaps' Law, setiap kata dalam teks terdiri dari beberapa kata unik yang lebih sedikit dari total kata pada keseluruhan teks dimana total kata ini juga bukan merupakan hasil penggandaan dari jumlah kata yang unik. Pada teks dengan total kata yaitu 442 kata memiliki jumlah kata unik yang lebih sedikit dari total kata yaitu 243 kata, di tempat kedua teks dengan total kata yaitu 840 kata memiliki 414 jumlah kata unik, dan diikuti oleh teks dengan total kata yaitu 1112 kata memiliki 528 jumlah kata unik. Maka, jumlah kata yang unik berkisar setengah dari total kata dalam teks.

Tabel 2 *Vocabulary Frequency*

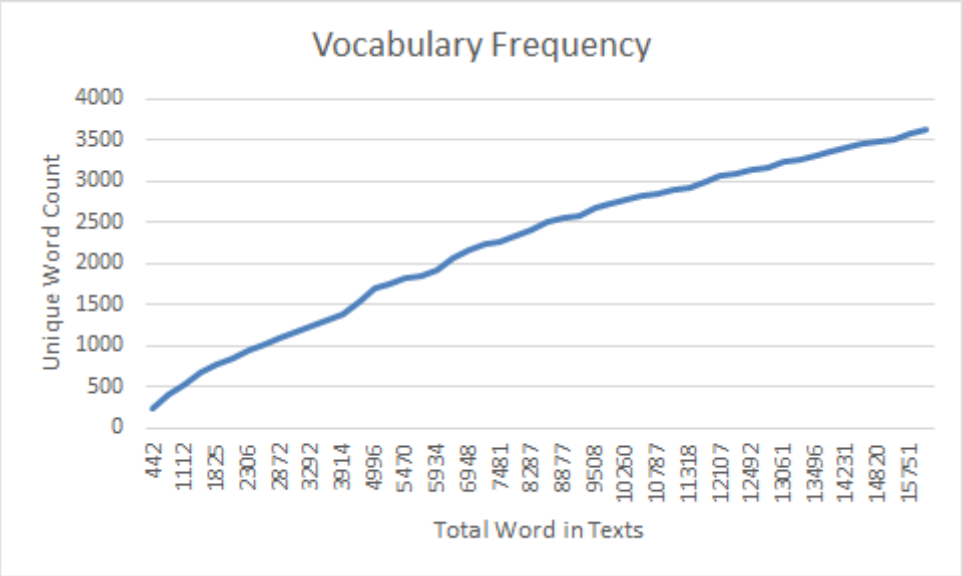
Texts analyzed	Unique Word Count	Total words in texts
1	243	442
2	414	840
3	528	1112
4	673	1546
5	778	1825
6	844	2015
7	931	2306
8	1012	2635
9	1093	2872
10	1173	3104

11	1224	3292
12	1307	3603
13	1389	3914
14	1540	4379
15	1690	4996
16	1754	5248
17	1813	5470
18	1854	5656
19	1926	5934
20	2077	6589
21	2161	6948
22	2229	7262
23	2266	7481
24	2330	7829
25	2419	8287

26	2502	8590
27	2545	8877
28	2582	9096
29	2673	9508
30	2731	9949
31	2776	10260
32	2818	10554
33	2848	10787
34	2890	11105
35	2922	11318
36	2989	11677
37	3072	12107
38	3099	12273
39	3135	12492
40	3174	12729

41	3232	13061
42	3276	13298
43	3324	13496
44	3371	13912
45	3421	14231
46	3455	14536
47	3477	14820
48	3510	15098
49	3593	15751
50	3638	16067

Tabel tersebut dapat disajikan dalam bentuk *line chart* seperti Gambar 2 berikut.



Gambar 2 *Vocabulary Frequency*

2.1.2 Analisis Preprocessing

Untuk mengimplementasikan algoritma TextRank untuk proyek ini, dataset harus diproses terlebih dahulu. Langkah-langkah yang dilakukan sampai menghasilkan vektor kalimat yaitu :

1. Memisahkan beberapa paragraf menjadi kalimat terpisah yang disebut *Sentence Segmentation*, yaitu tahapan memecah paragraf atau teks menjadi kumpulan kalimat berdasarkan tanda pemisah kalimat. Pemisahan kalimat dapat dilakukan berdasarkan adanya simbol titik, simbol tanda tanya, atau simbol tanda seru
2. Menggunakan kata-kata *stopwords* dari nltk package dan menghapus karakter khusus, membersihkan kalimat *(clean sentences)* yang diekstrak dari langkah sebelumnya. Semua kalimat kemudian ditulis dengan huruf kecil *(lower case).*
3. Kata-kata dari kalimat yang sudah dibersihkan kemudian diberi *tokenized.* Setiap kata dari kalimat tersebut diberi nilai token berdasarkan koefisien kata-kata yang disematkan *(embedded wording coefficient).*
4. Kata diubah menjadi representasi vektor *(vector representation)* menggunakan glove dan akhirnya kalimat menjadi representasi vektor dengan merata-ratakan semua vektor kata individual dalam sebuah kalimat.

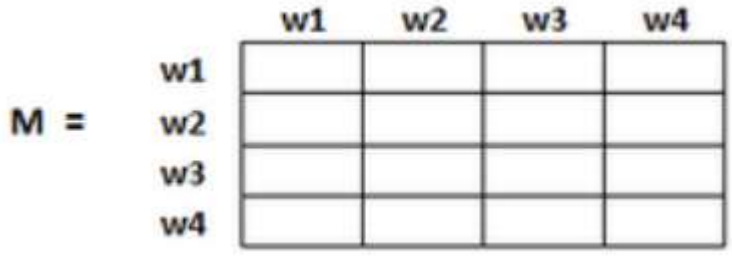
2.1.2 Analisis Metode

TextRank adalah *graph-based ranking algorithm* (algoritma graf dengan model pemeringkatan) untuk pemrosesan teks pada pemrograman bahasa alami atau NLP. Pentingnya sebuah vertex dalam sebuah graph, berdasarkan informasi global yang diambil secara rekursif dari keseluruhan *graph.* Ketika satu *vertex* terhubung ke *vertex* lain, itu pada dasarnya memberikan *vote* untuk *vertex* tersebut. Semakin tinggi jumlah *vote* yang diberikan untuk sebuah *vertex,* semakin penting *vertex* tersebut. Kalimat yang paling penting adalah kalimat yang paling mirip dengan kalimat lainnya. *Cosine similarity* digunakan sebagai *similarity metric*. Semua kalimat direpresentasikan sebagai vektor dan *cosine similarity* antara dua vektor kalimat didefinisikan sebagai berikut.

$$similarity = cos(\Theta) = \frac{A \,.\, B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i \, B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

A dan B adalah representasi vektor kalimat dari dua kalimat. Nilai *cosine similarity* berkisar antara -1 dan 1. -1 mewakili kalimat yang sama sekali berbeda dan 1 mewakili kalimat yang sangat mirip.

Untuk menentukan peringkat kalimat yang berbeda dalam korpus, dihitung skor kesamaan semua kalimat. Untuk menangkap persamaan, dibuat matriks persegi M, yang memiliki n baris dan n kolom, di mana n adalah jumlah kalimat. Contoh matriks dengan 4 kalimat ditunjukkan di bawah ini. Matriks diinisialisasi ke nilai kesamaan yang dihitung menggunakan *cosine similarity.*



Gambar 3 Matriks dengan 4 Kalimat

Graf yang dihasilkan oleh TextRank merupakan graf yang tidak mempunyai arah *(undirected)* dan berbobot *(weighted)* atau disebut juga *undirected weighted graph*. Persamaan dari algoritma TextRank adalah sebagai berikut.

$$W\,S(V_i) = (1-d) + d * \sum_{V_j \in In\,(V_i)} \frac{w_{ji}}{\sum_{V_k \in out\,(V_j)} w_{jk}}\, W\,S(V_j)$$

Dimana :

- WS = *Weight Sentence* (Bobot Kalimat)
- w = nilai *content overlap similarity*
- Vi = *vertex* yang dihitung skor-nya
- Vj = *vertex* yang bertetanggaan dengan Vi
- Vk = *vertex* yang bertetanggaan dengan Vj
- d = *damping factor* yang nilainya antara 0 dan 1, biasanya 0.85

Langkah-langkah yang digunakan untuk menerapkan algoritma TextRank yaitu :

1. Lakukan ekstraksi kalimat dengan menjadikan seluruh kalimat sebagai *vertex* dalam graf.
2. Identifikasi hubungan antarkalimat dengan membuat *edges* antara *vertex-vertex* yang dapat diidentifikasi dengan menggunakan *content overlap similarity.*
3. Beri skor awal *vertex* untuk menentukan iterasi.
4. Lakukan iterasi algoritma TextRank sampai *error rate* tiap *vertex konvergen* di bawah *threshold. Error Rate* adalah perbedaan antar dua skor *vertex* yang dihitung pada iterasi yang berurutan dengan rumus:

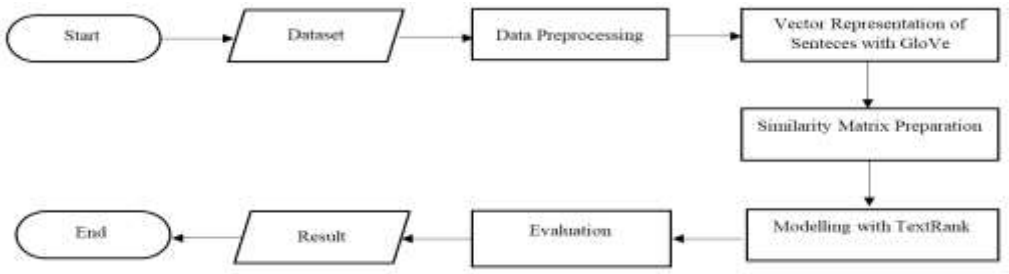
   $$S^{k+1}\,(V_i) - S^{k}\,(V_i)$$

   Dimana :

   - Vi = V*ertex* yang dihitung skornya
   - k = pada iterasi ke-k
5. Setelah graf terbentuk, *vertex* diurutkan berdasarkan skor akhirnya dan diambil *top-rank* kalimat sebagai hasil ekstraksi ringkasannya.

2.2 Desain

Pada sub bab ini dijelaskan desain pemrosesan bahasa alami yaitu peringkasan teks bahasa inggris yang ditampilkan dalam bentuk *flowchart* atau diagram alir seperti pada gambar berikut.



Gambar 4 Desain model

2.2.1 *Dataset*

Data yang akan diringkas adalah data yang dapat diperoleh secara open source yaitu BBC News Summary. Data ini dipublikasikan oleh kaggle dan dapat diakses pada link berikut: https://www.kaggle.com/pariza/bbc-news-summary/data.

2.2.2 *Data Preprocessing*

Tahap *preprocessing* bertujuan untuk menyiapkan teks yang akan diringkas menjadi data yang siap untuk diproses pada tahap berikutnya. *Dataset* yang berasal dari suatu sumber dapat merupakan data terstruktur dan data yang tidak terstruktur. Maka sebelum diolah, data harus melalui tahapan *preprocessing*, dimana tahap ini akan memproses data awal dengan mentransformasi data yang awalnya tidak terstruktur menjadi data yang sudah terstruktur [8].

Tahapan ini akan melibatkan langkah-langkah seperti *data cleaning, stopword removal, tokenization* dan *case folding*.

1. *Data Cleaning*

   Data yang digunakan dari dataset perlu untuk dibersihkan karena data yang kotor kemungkinan besar akan berpengaruh terhadap model dan hasil peringkasan yang akan dihasilkan. Dataset ini dapat dilakukan pembersihan dengan menghapus data yang kosong atau dengan mengisi

data yang kosong tersebut dan dengan memperbaiki struktur kata dalam dataset.

2. *Stopword Removal*

   Stopword removal merupakan tahapan menghapus *stopword*, yaitu kata-kata yang tidak relevan dan yang sering muncul pada teks [9]. *Stopword* dapat berupa kata seru, kata depan, atau kata sambung.

3. *Case Folding*

   Case folding adalah proses mengkonversi setiap huruf kapital menjadi huruf kecil dalam suatu kalimat [10]. Hal ini diperlukan untuk mempermudah proses pencarian kata yang sama. Contoh *case folding* dapat dilihat pada Tabel … berikut.

Tabel 3 Contoh proses *Case Folding*

Text	***Case Folding***
I am the happiest girl	i am the happiest girl

4. Tokenization

   Pada tahap *tokenization*, setiap kalimat akan dipisah per kata berdasarkan tanda spasi antar kalimat. Setelah setiap kata dipisahkan, maka tanda baca akan dihapus.

2.2.3 Vector Representation of Sentences with GloVe

GloVe Word Embedding merupakan representasi vektor dari sejumlah kata yang digunakan untuk membuat vektor dalam sebuah kalimat. Dalam representasi vektor ini juga dapat menggunakan pendekatan Bag of Words atau *Term Frequency – Inverse Document Frequency (TF-IDF)*, namun pada GloVe ini kan mengabaikan urutan dari kata. Data ini dipublikasikan oleh Stanford dan dapat diakses pada link berikut: http://nlp.stanford.edu/data/glove.6B.zip.

2.2.4 Similarity Matrix Preparation

Sebuah dokumen dapat direpresentasikan sebagai vektor dalam sebuah *vector space model* [11]. Dalam hal ini, dokumen yang serupa akan memiliki vektor yang serupa juga karena dalam suatu dokumen yang serupa akan memiliki kata yang sama juga. Seperti itu juga halnya dengan konsep kata, kata yang sama akan memiliki vektor yang sama karena akan cenderung muncul pada dokumen yang sama [11]. Untuk menghitung kemiripan antara vektor kata ini dapat dilihat

berdasarkan sudut dari dua buah vektor dalam melakukan *inner product* dan dihitung yaitu dengan menggunakan pendekatan *cosine similarity*. Berikut merupakan persamaan *cosine similarity.*

$$cosine(\vec{v},\vec{w}) = \frac{\vec{v}\,.\,\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum w_i^2}}$$

Dimana,

- v_i = jumlah kata v dalam konteks i.
- w_i = jumlah kata w dalam konteks i.

2.2.5 Modelling with TextRank

Setelah semua kalimat dalam artikel direpresentasikan dalam vektor dan dihitung kemiripannya, selanjutnya adalah membangun model peringkasan teks bahasa inggris dengan menggunakan algoritma TextRank. TextRank merupakan sebuah algoritma berbasis grafik yang dapat digunakan untuk memproses data teks [12] . Algoritma ini akan menghasilkan ekstraksi kalimat yang disebut dengan hasil ringkasan teks tanpa diperlukannya data latih. Pada TextRank akan dibangun sebuah graf yang berisi hubungan antarkalimat dalam dokumen yang berisi vertex. Vertex direpresentasikan sebagai peringkat dalam teks dan memiliki *similarity* yang dihubungkan oleh *edges* dan dihitung oleh *cosine similarity*.

Graf yang akan dihasilkan oleh TextRank breupa graf yang tidak mempunyai arah (*undirected*) dan berbobot (*weighted*). Berikut adalah rumus dari TextRank adalah [12]:

$$WS(V_i) = (1-d) + d * \sum_{V_j \in Adj(V_i)} \frac{W_{jk}}{\sum V_k \in Adj(V_j) W_{jk}} WS\,(v_i)$$

Keterangan :

- WS = *Weight Sentence* (Bobot Kalimat)
- w = Nilai *content overlap similarity*
- Vi = Vertex yang dihitung skor-nya
- Vj = Vertex yang bertetanggaan dengan Vi

- Vk = Vertex yang bertetanggaan dengan Vj
- d = *Damping factor* yang nilainya antara 0 dan 1

2.2.6 Evaluation and Results

Pada tahap ini dilakukan evaluasi dan analisis terhadap hasil ringkasan yang telah diperoleh dengan algoritma TextRank. Tahap ini bertujuan untuk memastikan bahwa proyek ini dapat menghasilkan hasil ringkasan yang baik dan sesuai dengan makna dokumen aslinya. Adapun pendekatan yang akan digunakan untuk evaluasi hasil ringkasan tersebut adalah metode ROUGE-L. ROUGE-L merupakan matrik evaluasi yang menghitung nilai kesamaan struktur dokumen ringkasan dengan ringkasan pembanding secara statistik serta akan menghasilkan skor yang menunjukkan nilai perbandingan antara hasil ringkasan dengan ringkasan referensi. Nilai atau skor ROUGE ini juga akan dijadikan sebagai kriteria pembanding yang didasarkan pada skor ROUGE ringkasan referensi.

2.3 Implementasi

Pada subbab ini akan dijelaskan implementasi model peringkasan dengan algoritma text rank. Implementasi dimulai dengan mengimpor module *zip file* dan kemudian membuka object ZipFile dalam menuliskan mode dengan mengkhususkan parameter kedua sebagai 'r'. Parameter pertama adalah jalur untuk file itu sendiri. Berikut kode yang dibutuhkan:

```
# Load Data
import zipfile
with zipfile.ZipFile('BBC News Summary.zip', 'r') as zip_ref:
    zip_ref.extractall('')
```

Import os digunakan untuk pemanggilan fungsi dengan nama os.

```
import os
business_texts=os.listdir('BBC News Summary/News Articles/business')

import os
entertainment_texts=os.listdir('BBC News Summary/News Articles/entertainment')

import os
politics_texts=os.listdir('BBC News Summary/News Articles/politics')

import os
sport_texts=os.listdir('BBC News Summary/News Articles/sport')

import os
tech_texts=os.listdir('BBC News Summary/News Articles/tech')
```

1. Data Cleaning

Pada proses ini, teks *(string)* akan dikonversi menjadi huruf kecil. Digunakan beberapa fungsi seperti fungsi modifikasi string yaitu metode sub() dan isalpha() yang digunakan untuk mengembalikan *true* jika *string* memiliki minimal 1 karakter dan semua karakter adalah abjad dan *false* sebaliknya.

```
def clean_text(text):

    text=text.lower()
    text=' '.join([contraction_mapping[i] if i in contraction_mapping.keys() else i for i in text.split()])
    text=re.sub(r'\(.*\)',"",text)
    text=re.sub("'s","",text)
    text=re.sub('"','',text)
    text=' '.join([i for i in text.split() if i.isalpha()])
    text=re.sub('[^a-zA-Z]'," ",text)

    return text
```

2. Tokenization

Tokenization dilakukan dengan menggunakan *library NLTK* dan mengunduh punkt model. Pada bagian ini akan memecah artikel kedalam beberapa kalimat.

```
# tokenization
import nltk
nltk.download('punkt') # one time execution
from nltk.tokenize import sent_tokenize
sentences = []
root='BBC News Summary/BBC News Summary/News Articles/business'
for s in business_texts:
  file_ = open(root+'/'+s, "r")
  business_file=file_.read().replace('\n', '.')
  business_file=business_file.split('.')
  for sent in business_file:
    if sent!='':
      if sent[0].isdigit():
        sentences[-1][0]+=sent_tokenize(sent)[0]
      else:
        sentences.append(sent_tokenize(sent))
```

3. *Stopword Removal*

Pada tahap ini akan menghapus beberapa kata yang tergolong kedalam *stopword* seperti *could*, *would*, *more*, *however*, dan kata henti lainnya dalam bahasa inggris.

```
# function to remove stopwords
def remove_stopwords(sen):
    sen_new = " ".join([i for i in sen if i not in stop_words])
    return sen_new
```

```
# remove stopwords from the sentences
clean_sentences = [remove_stopwords(r.split()) for r in clean_sentences]
```

4. Vector Representation of Sentences

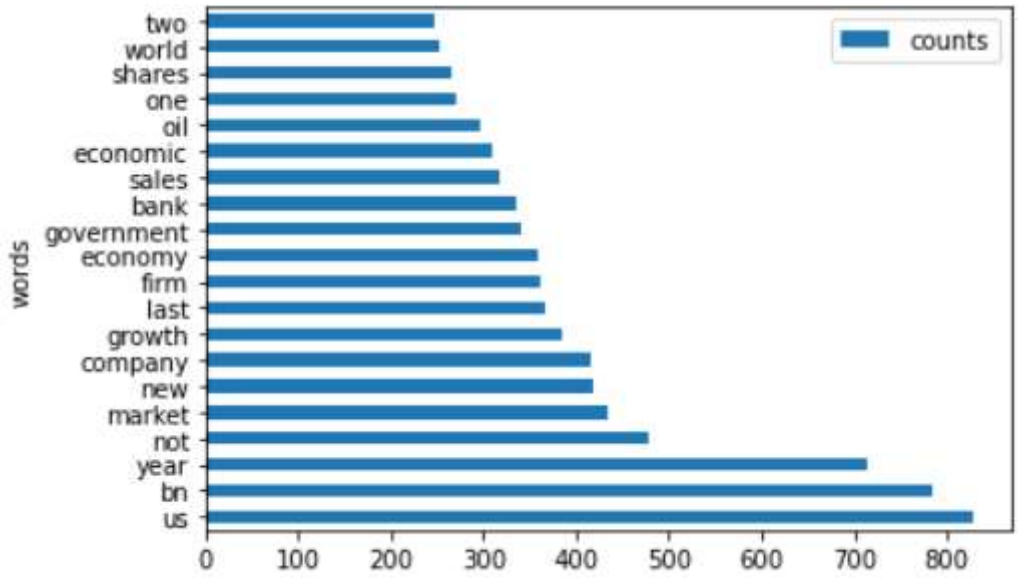
Pada tahap ini akan dilakukan ekstraksi kata dalam setiap artikel menjadi vektor dengan menggunakan data GloVe. Langkah awal yang dilakukan adalah dengan mengunduh data Glove tersebut ke dalam proyek file. Seperti yang ditunjukkan oleh potongan kode berikut.

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip glove*.zip
```

Langkah selanjutnya adalah mengekstrak kata menjadi bentuk vektor dengan menggunakan data Glove yang sudah diunduh sebelumnya.

```
# Extract word vectors
import numpy as np
word_embeddings = {}
f = open('glove.6B.100d.txt', encoding='utf-8')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    word_embeddings[word] = coefs
f.close()
```

Setelah tahap sebelumnya dilakukan, maka akan dijalankan kode program yang menampilkan 20 kata dengan frekuensi terbanyak seperti pada artikel dengan topik “Business” berikut.



Gambar 5 Histogram Frekuensi kata

Untuk menampilkan data teks secara visual digunakan metode *Word Cloud*. Grafik ini populer karena cukup mudah dipahami. Dengan menggunakan *word cloud*, gambaran frekuensi kata-kata dapat ditampilkan dalam bentuk yang menarik namun tetap informatif. Semakin sering satu kata digunakan, maka semakin besar pula ukuran kata tersebut ditampilkan dalam *word cloud* yang dapat dilihat pada artikel business berikut.

Gambar 6 *Word Cloud* pada berita business

5. Similarity Matrix Preparation

Setelah proses ekstraksi kata dilakukan, maka langkah selanjutnya adalah menghitung tingkat kemiripan antar kalimat dengan menggunakan pendekatan *cosine similarity*.

```
def cosine_similarities(mat):
  norm = (mat.T * mat.T).sum(0, keepdims=True) ** .5
  return (mat @ mat.T ) / norm.T / norm

sim_mat=cosine_similarities(sentence_vectors)
```

Langkah awal yang akan dilakukan adalah dengan mendefinisikan sebuah fungsi *cosine similarity* dengan menggunakan kalimat yang telah di vektorisasi. Selanjutnya adalah mendefinisikan *zero matrix* dengan dimensi n * n yang akan dijadikan sebagai inisialisasi matriks pada skor *cosine similarity* setiap kalimat kemudian matrix sim_mat akan dikonversi kedalam bentuk *graph*. Seperti yang ditunjukkan pada potongan kode berikut.

```
from scipy import sparse
sim_mat[sim_mat < 0.9]=0
for i in range(8684):
  sim_mat[i,i] = 0
sim_mat[np.isnan(sim_mat)] =0
sparse_mat = sparse.csr_matrix(sim_mat)
sparse_mat.eliminate_zeros()
```

6. Implementasi Algoritma TextRank

Pada bagian ini akan dilakukan pemodelan peringkasan teks bahasa inggris dengan menggunakan algoritma TextRank. Langkah awal yang dilakukan adalah dengan *import library* network dan menampilkan hasil konversi matriks sim_mat yaitu sparse_mat.

```
import networkx as nx
SimilarityGraph = nx.from_scipy_sparse_matrix(sparse_mat)
#print the graph
nx.draw_networkx(SimilarityGraph,pos=nx.shell_layout(SimilarityGraph), with_labels=True)
```

Setelah dilakukan pemodelan maka model tersebut dapat digunakan untuk mengekstraksi ringkasan teks di setiap artikel pada dataset BBC News.

```
scores = nx.pagerank(SimilarityGraph)
ranked_sentences = sorted(((scores[i],s) for i,s in enumerate(sentences)), reverse=True)
# Extract top 10 sentences as the summary
for i in range(10):
  print(ranked_sentences[i][1])
```

Pada potongan kode diatas, akan ditampilkan juga 10 kalimat ringkasan teratas pada suatu artikel. Misalnya pada artikel bisnis pada dataset BBC News.

```
[" However, analysts say they expected most shareholders would be holding back from selling all their shares immediately, as Go
ogle's good performance and future growth potential means demand will hold"]
["Thursday's new low for the dollar came after data was released showing year-on-year sales of new homes in the US had fallen 1
2% in November - with some analysts saying this could indicate problems ahead for consumer activity"]
['" Mr Majumdar also said an assessment should be made as to whether foreign investment is indeed beneficial to the country - i
n terms of employment and money generated - or just another way of international companies filling their deep pockets']
["Most EU countries have failed to put in place policies aimed at making Europe the world's most competitive economy by the end
of the decade, a report says"]
['" The CBI found some firms managed to increase prices for the first time in nine years - but many said increases failed to ke
ep up the rise in costs']
[" Job creation was one of last year's main concerns for the US economy"]
[" Job creation was one of last year's main concerns for the US economy"]
[" Jet's IPO is the first in a series of expected share offers from Indian companies this year, as they move to raise funds to
help them do business in a rapidly-growing economy"]
['Retail sales are seen as a major part of consumer spending - which in turn makes up two-thirds of economic output in the US']
[' The news and financial data seller said the year had begun well, adding it expected "further gradual improvement" in the sec
ond quarter of the year after good January sales']
```

2.4 Hasil

Secara umum metode evaluasi yang diterapkan pada suatu peringkasan teks adalah metode intrinsik. Metode ini akan membandingkan hasil ringkasan teks oleh mesin dengan hasil ringkasan ideal . Salah satu dari metode tersebut yang dapat mengukur hasil dari ringkasan teks adalah ROUGE. Metode evaluasi *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) merupakan metode yang melakukan perhitungan terhadap total n-gram kata yang overlap antara hasil ringkasan mesin dan ringkasan ideal. Dalam melakukan pengukurannya, ROUGE terdiri dari ROUGE lainnya seperti ROUGE-N, ROUGE-L, dan ROUGE-S. Variasi dari ROUGE ini akan menghitung jumlah unit dari n-gram yang saling bertindih, urutan kata, dan pasangan kata antara kandidat ringkasan dan referensi.

3. PENUTUP

3.1 Pembagian Tugas dan Tanggung Jawab

Berikut adalah tabel pembagian tugas setiap anggota kelompok.

Tabel 1. *Team Member and Roles*

Member	*Role*	*Task*
Kiky Purnamasari Napitupulu	*System Analyst*	Berperan dalam perencanaan, pengkoordinasian, pengerjaan serta menganalisis hasil dari yang sudah dikerjakan
	Programmer	Berperan untuk mengimplementasikan code untuk membangun sebuah sistem serta melakukan pengujian terhadap sistem yang sudah dibangun
Tripheni Simanjuntak	*System Analyst*	Berperan dalam perencanaan, pengkoordinasian, pengerjaan serta menganalisis hasil dari yang sudah dikerjakan
	Programmer	Berperan untuk mengimplementasikan code untuk membangun sebuah sistem serta melakukan pengujian terhadap sistem

		yang sudah dibangun
Jessycha Royanti Tampubolon	*System Analyst*	Berperan dalam perencanaan, pengkoordinasian, pengerjaan serta menganalisis hasil dari yang sudah dikerjakan
	Programmer	Berperan untuk mengimplementasikan code untuk membangun sebuah sistem serta melakukan pengujian terhadap sistem yang sudah dibangun

3.2 Kesimpulan

Peringkasan teks otomatis berfungsi untuk menghasilkan ringkasan singkat dengan tetap mempertahankan informasi utama dari korpus teks yang besar. Peringkasan ekstraktif bergantung pada identifikasi kalimat yang tepat. Untuk keperluan proyek ini, kami menerapkan algoritma TextRank dengan pendekatan ekstraktif menggunakan teknik probabilitas untuk memberikan skor pada kalimat dan mengekstrak kalimat terpenting berdasarkan skor tertinggi. Tujuan utama kami adalah meringkas berbagai kategori berita pada dataset yang berisi kalimat representatif paling penting. Dengan menggunakan algoritma TextRank untuk peringkasan artikel berita, kami dapat mengekstrak kata dan kalimat yang paling penting dan kata-kata yang paling sering diulang seperti yang terlihat pada histogram jumlah frekuensi, diambil dari kalimat terpenting dari artikel.

3.3 Saran

Saran yang mungkin dapat dicoba oleh pihak yang ingin mengembangkan sistem peringkas otomatis menggunakan algoritma TextRank ini lebih lanjut adalah:

1. Menggunakan metode *similarity* yang lain untuk mendapatkan nilai hubungan antar kalimat (nilai *edges*).

2. Melakukan pengukuran dan perbaikan dengan memperoleh terlebih dahulu evaluasi kuantitatif atas implementasi yang dilakukan.

DAFTAR PUSTAKA

[1] S. Massung and C. X. Zhai, *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. ACM / Association for Computing Machinery, 2016.

[2] Kumar, J. N. Madhuri dan R. Ganesh, "Extractive Text Summarization Using Sentence Ranking," *2019 International Conference on Data Science and Communication (IconDSC)*, 2019.

[3] R. Adelia, S. Suyanto dan U. N. Wisesty, "Indonesian abstractive text summarization using bidirectional gated recurrent unit," *Procedia Computer Science*, vol. 157, p. 581–588, 2019.

[4] V. Gupta and G. S. Lehal, "A Survey of Text Summarization Extractive Techniques," *Journal of Emerging Technologies in Web Intelligence*, vol. 2, no. 3, 2010.

[5] Y. Sabina, Tumpa, P. Basak, N. A. Mahjabin, Uddin, Md.Palash, A. E. a. Afjal dan M. Ibn, "Study on Abstractive Text Summarization Techniques," *American Journal of Engineering Research (AJER)*, vol. 6, no. 8, pp. 254-255, 2017.

[6] R. Indrianto, M. A. Fauzi and L. Muflikhah, "Peringkasan Teks Otomatis Pada Artikel Berita Kesehatan Menggunakan K-Nearest Neighbor Berbasis Fitur Statistik", Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2017.

[7] Syaliman, Y. Yuliska dan K. U., "Literatur Review Terhadap Metode, Aplikasi dan Dataset Peringkasan Dokumen Teks Otomatis untuk Teks Berbahasa Indonesia," *IT Journal Research and Development*, vol. 5, no. 1, p. 9–3, 2020.

[8] A. P. Widyassari, S. Rustad, G. F. Shidik, E. Noersasongko, A. Syukur, A. Affandy dan D. R. S. Ignatius Moses, "Review of automatic text summarization techniques & methods," *Journal of King Saud University - Computer and Information Sciences*, 2020.

[9] S. Babar dan P. D. Patil, "Improving Performance of Text Summarization," *Procedia Computer Science*, vol. 46, pp. 354 - 263, 2015.

[10] K. Shetty dan J. S. Kallimani, "Automatic extractive text summarization using K-means clustering," *IEEE*, p. 2017, 2018.

[11] D. Jurafsky dan J. H. Martin, Speech and language processing, Harlow: Pearson, 2014.

[12] Christanti. Viny, Eris, Pragantha. Jeanny, “Penerapan Algoritma TextRank untuk *Automatic Summarization* pada Dokumen Berbahasa Indonesia”, Jurnal Ilmu Teknik dan Komputer, vol.1, no.1, 2017.