Peringkas Teks Otomatis Bahasa Indonesia secara Abstraktif menggunakan Metode *Long Short-Term Memory*

Muhammad Alfhi Saputra[1], Wikky Fawwaz Al Maki[2]

[1,2]Fakultas Informatika, Universitas Telkom, Bandung
[1]alfhisa@students.telkomuniversity.ac.id, [2]wikkyfawwaz@telkomuniversity.ac.id,

Abstrak

Salah satu topik dalam bidang *Natural Language Processing* (NLP) yang cukup menantang adalah peringkas teks otomatis. Dalam praktiknya peringkas teks otomatis terbagi menjadi dua pendekatan, yaitu ekstraktif dan abstraktif. Pendekatan abstraktif dinilai lebih baik karena cara kerjanya mendekati cara kerja manusia ketika meringkas teks atau yang disebut parafrase. Metode yang digunakan pada penelitian ini adalah *Long Short-Term Memory* (LSTM) yang mana metode tersebut telah sukses melakukan peringkasan dalam Bahasa Inggris. Dataset yang digunakan adalah kumpulan artikel berita media daring Bahasa Indonesia. Hasil terbaik yang didapatkan pada pengujian dengan metode LSTM menggunakan metode evaluasi ROUGE-1 adalah 0.13846.

Kata kunci: peringkas teks otomatis, abstraktif, Bahasa Indonesia, *long short-term memory, ROUGE*

Abstract

One topic about natural language processing that is quite challenging is automatic text summarization. Automatic-text-summarization is practically divided into two kinds of approach, namely extractive and abstractive. Abstractive-approach is considered better since it resembles how humans work in terms of text summarizing or paraphrasing. A method used in this study is Long Short-Term Memory (LSTM) which has succeeded to summarize texts in English. Datasets that have been used are a number of online news articles in Bahasa Indonesia. The best result gained using LSTM based on the ROUGE-1 evaluation is 0.13846.

Keywords*: *automatic text summarization, Bahasa Indonesia, long short-term memory, ROUGE

1. Pendahuluan

1.1 Latar Belakang

Automatic Text Summarization (ATS) adalah suatu sistem yang melakukan pemendekan suatu dokumen secara otomatis tanpa mengurangi informasi yang penting dari dokumen tersebut. ATS merupakan salah satu area yang paling banyak diteliti di dalam *Natural Language Processing* (NLP) [1]. Beberapa contoh penerapan dari *Automatic Text Summarization* adalah seperti ringkasan yang ditampilkan pada hasil pencarian di suatu search engine, ringkasan mengenai suatu produk berdasarkan *review* nya pada *e-commerce*, dan pada artikel berita suatu media.

Secara umum *Automatic Text Summarization* terbagi menjadi dua metode, yaitu ekstratif dan abstraktif. Metode ekstraktif melakukan peringkasan dengan mengambil bagian-bagian penting dari suatu dokumen tanpa ada mengubah kata apapun. Sedangkan, metode abstraktif melakukan ringkasan seperti melakukan parafrase, sehingga memungkinkan terdapat kata-kata yang sebelumnya tidak ada pada dokumen serta hasil ringkasan dapat mendekati dengan ringkasan manusia [2].

Penelitian mengenai *Automatic Text Summarization* sebenarnya sudah banyak dilakukan, tetapi kebanyakan masih menggunakan pendekatan ekstraktif [3]. Contoh penelitian kasus ini dengan pendekatan ekstraktif menggunakan metode *Naïve Bayes* dan dilatih dengan 100 artikel berita Indonesia mendapatkan hasil dengan *f-measure* mencapai 0.7152 [4]. Hasil lain menunjukkan dengan metode *maximum marginal relevance* mendapatan hasil *f-measure* 0.66 [5]. Metode *Term Frequency* juga pernah dipakai dalam kasus serupa mendapatkan hasil *f-measure* 0.78 [6]. Penelitian dengan pendekatan abstraktif untuk bahasa Indonesia juga telah dilakukan dengan menggunakan Bi-GRU RNN dan mendapatkan hasil terbaik dengan ROUGE-1 0.11975 [3].

Metode yang cukup populer untuk kasus *abstractive summarization* adalah RNN. Salah satu model dari RNN adalah LSTM, model LSTM sudah berhasil melakukan *abstractive summarization* dengan kasus bahasa Inggris yang menghasilkan *precision* 88%. Dengan hasil yang cukup baik dalam bahasa Inggris, maka metode inilah yang akan digunakan untuk penelitian ini.

1.2 Topik dan Batasannya

Penelitian yang akan dilakukan adalah menerapkan metode *long short-term memory* untuk melakukan peringkasan secara otomatis dengan pendekatan abstraktif untuk studi kasus Bahasa Indonesia. Batasan-batasan yang diterapkan pada penelitian ini, yaitu dataset yang digunakan pada penelitian ini adalah dataset kumpulan artikel berita berbahasa Indonesia INDOSUM [7], jumlah dataset yang digunakan pada penelitian ini adalah 10 ribu sampel data dari INDOSUM. Pada dataset INDOSUM, terdapat pengkategorian artikel berita menjadi enam kategori, yaitu hiburan, inspirasi, olahraga, industri hiburan, *headline*, dan teknologi, tetapi pada penelitian ini semua kategori bertia dianggap sama. Kemudian metode yang digunakan untuk pengujian adalah *long short-term memory*, lalu untuk melakukan evaluasi performansi menggunakan metode ROUGE-1.

1.3 Tujuan

Tujuan dari penelitian ini adalah dapat menunjukkan bahwa metode LSTM dapat diterapkan untuk kasus peringkas teks otomatis pada teks bahasa indonesia dengan pendekatan abstraktif.

2. Studi Terkait

2.1 Penelitian Terdahulu

Secara umum *Automatic Text Summarization* terbagi menjadi dua metode, yaitu ekstratif dan abstraktif. Metode ekstraktif melakukan peringkasan dengan mengambil bagian-bagian penting dari suatu dokumen tanpa ada mengubah kata apapun. Di samping itu, metode abstraktif melakukan ringkasan seperti melakukan parafrase, sehingga memungkinkan terdapat kata-kata yang sebelumnya tidak ada pada dokumen serta hasil ringkasan dapat mendekati dengan ringkasan manusia [2].

Penelitian mengenai *Automatic Text Summarization* sebenarnya sudah banyak dilakukan, tetapi kebanyakan masih menggunakan pendekatan ekstraktif [3]. Contoh penelitian kasus ini dengan pendekatan ekstraktif menggunakan metode *Naïve Bayes* dan dilatih dengan 100 artikel berita Indonesia mendapatkan hasil dengan *f-measure* mencapai 0.7152 [4]. Hasil lain menunjukkan dengan metode *maximum marginal relevance* mendapatan hasil *f-measure* 0.66 [5]. Metode *Term Frequency* juga pernah dipakai dalam kasus serupa mendapatkan hasil *f-measure* 0.78 [6]. Penelitian dengan pendekatan abstraktif untuk Bahasa Indonesia juga telah dilakukan dengan menggunakan Bi-GRU RNN dan mendapatkan hasil terbaik dengan ROUGE-1 0.11975 [3]. Berdasarkan pemaparan studi terkait di atas, maka dapat disimpulkan bahwa penulis tertarik untuk meneliti topik ini karena penelitian dengan pendekatan abstraktif masih relatif sedikit untuk kasus Bahasa Indonesia. Lalu peneliti juga ingin mencoba metode lain apakah dapat lebih baik daripada metode Bi-GRU RNN.

2.2 Automatic Text Summarization

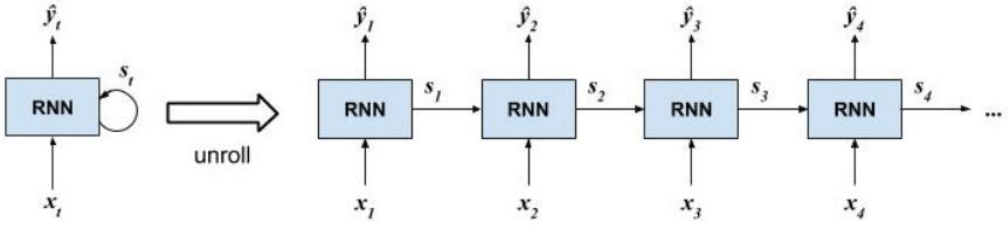
Automatic Text Summarization (ATS) atau dalam Bahasa Indonesia Peringkas Teks Otomatis adalah suatu sistem yang melakukan peringkasan atau pemendekkan teks yang panjang secara otomatis tanpa menghilangkan informasi penting yang ada pada teks. Secara umum, peringkas ATS terbagi atas dua pendekatan, yaitu pendekatan ekstraktif dan abstraktif. Pendekatan ekstraktif melakukan peringkasan dengan cara mengekstrak kalimat-kalimat penting di dalam teks lalu disusun kembali sebagai suatu teks baru. Sedangkan pendekatan abstraktif melakukan peringkasan seperti melakukan parafrese pada teks, sehingga memungkinkan muncul kata-kata baru yang tidak terdapat pada kalimat awal [1].

2.3 INDOSUM

INDOSUM [7] merupakan dataset yang diklaim sebagai benchmark untuk kasus peringkas teks otomatis Bahasa Indonesia. Dataset tersebut berisi 19 ribu pasangan artikel berita dan *gold summaries. Gold summaries* didefinisikan sebagai hasil ringkasan yang dibuat secara alami oleh dua orang penutur asli bahasa Indonesia. Artikel-artikel pada dataset tersebut merupakan artikel berita yang diambil dari portal berita *online* seperti CNN Indonesia, Kumparan, dan lain-lain. Artikel-artikel pada dataset tersebut terdiri dari enam kategori berita, yaitu hiburan, inspirasi, olahraga, industri hiburan, *headline*, dan teknologi. Pada penelitian ini penulis hanya menggunakan 10 ribu sampel data dari dataset tersebut dan setiap kategori berita dianggap sama.

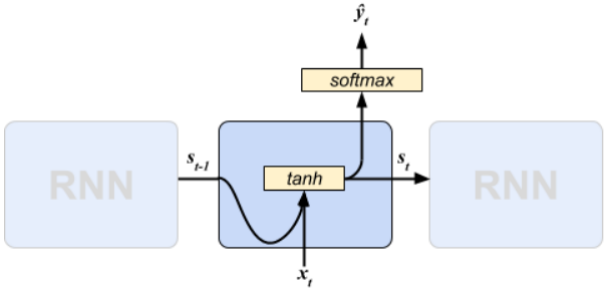
2.4 Recurrent Neural Network

Recurrent Neural Network adalah salah satu pengembangan dari *Artificial Neural Network* yang menggunakan hasil dari prosesnya sebagai masukan dan terus berulang. Berikut adalah ilustrasi proses RNN.



Gambar 1 Ilustrasi proses RNN (src : indoml.com)

Berdasarkan gambar 1, berikut adalah keterangan dari gambar tersebut. **xt** adalah masukan dari setiap iterasi (*timestamp*) yang dalam kasus pemrosesan kalimat adalah kata ke-t pada kalimat yang sedang diproses. Lalu **st** adalah *hidden state* pada setiap timestamp **t**. Nilai **st** dipengaruhi oleh *hidden state* **t-1** dan berdasarkan *input* yang sedang diproses. Selanjutnya **yt** adalah nilai keluaran dari setiap iterasi. Unit RNN dapat digambarkan sebagai berikut.



Gambar 2 Ilustrasi proses satu unit RNN (source : indoml.com)

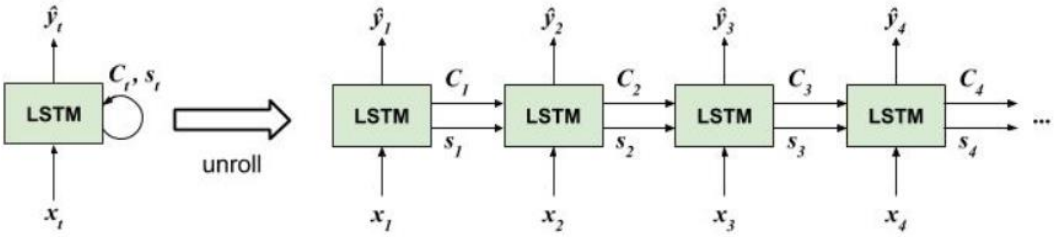
Berikut persamaan yang berlaku dalam satu unit RNN.

$$s_t = \tanh\left(U.x_t + W.s_{t-1}\right)$$

$$\hat{y} = softmax(V.s_t)$$

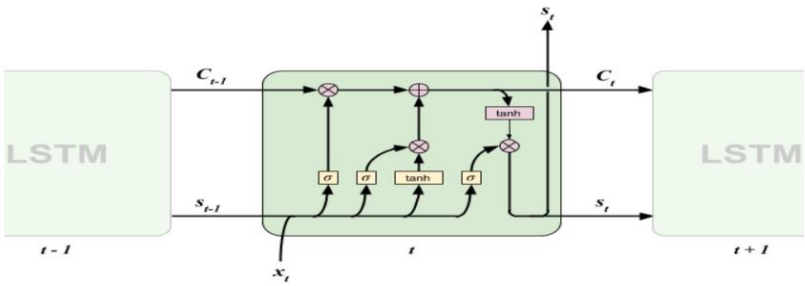
2.5 Long Short-Term Memory

LSTM adalah salah satu bentuk arsitektur dari RNN. Perbedaan LSTM dengan RNN adalah pada LSTM terdapat parameter tambahan yang biasa disebut konteks, sehingga ilustrasi dari LSTM adalah sebagai berikut.



Gambar 3 Ilustrasi proses LSTM (source : indoml.com)

Berikut merupakan ilustrasi satu unit LTSM



Gambar 4 Ilustrasi proses 1 unit LSTM (source : indoml.com)

Berikut adalah persamaan yang digunakan dalam LSTM.

$$f_t = \sigma\left(W_f \,.\, [s_{t-1}, x_t] + b_f\right)$$

$$i_t = \sigma(W_i \,.\, [s_{t-1}, x_t] + b_i)$$

$$\widetilde{C_t} = \tanh(W_C \,.\, [s_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$$

$$o_t = \sigma(W_o \,.\, [s_{t-1}, x_t] + b_o)$$

$$s_t = o_t * \tanh(C_t)$$

Model LSTM ini sudah terbukti dapat menyesaikan beberapa kasus NLP, seperti *machine translation* [8] dan *text summarization* [9].

2.6 Encoder-Decoder

Metode encoder-decoder digunakan karena metode ini dapat menyelesaikan masalah *sequence-to-sequence* (*seq2seq*), yang mana panjang hasil ringkasan sebagai *output* berbeda dengan panjang *input*-nya. Metode *encoder-decoder* ini telah teruji dapat menyelesaikan kasus *automatic text summarization* [10].

1. *Encoder*
   Encoder membaca teks input lalu meringkas informasi dan dimasukkan ke dalam *hidden state* dan *cell state vectors.* Lalu *encoder* membuat *context vector* yang akan menjadi *input* untuk *decoder*.
2. *Decoder*
   Decoder berfungsi untuk membuat *output sequence* berdasarkan *context vector* yang sudah diproses pada *encoder* sebelumnya.

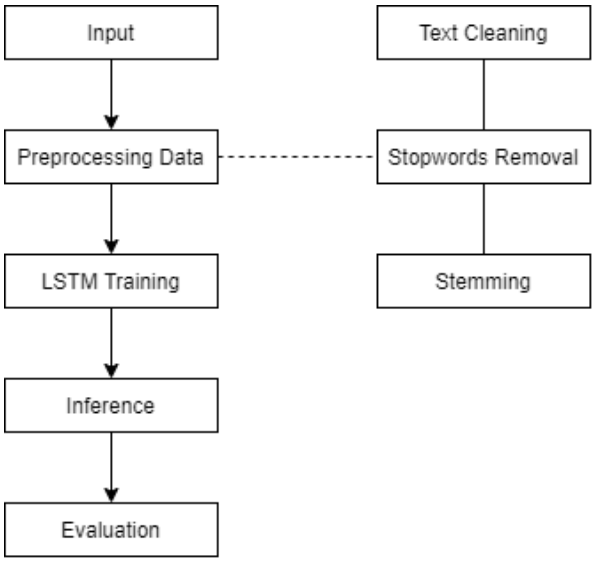
2.7 ROUGE

ROUGE atau *Recall-Oriented Understudy for Gisting Evaluation* merupakan suatu metriks pengukuran untuk menilai performansi hasil ringkasan teks. Cara mengevaluasi hasil ringkasan teks menggunakan ROUGE adalah dengan melihat kemiripan ringkasan asli dengan ringkasan yang dibuat oleh model. Terdapat beberapa jenis ROUGE, diantaranya adalah menggunakan *unigram* dan *bigram* [11]. Metode evaluasi ROUGE juga pernah dipakai dalam penelitian Bahasa Indonesia [12]. Pengukuran yang digunakan pada ROUGE ini adalah *recall*, *precision*, dan *f1-score*.

Recall pada ROUGE adalah perbandingan antara banyaknya kata yang sama yang muncul pada kedua jenis ringkasan (ringkasan oleh sistem dan *gold summaries*) dibagi dengan banyaknya kata pada *gold summary*. Lalu precision pada ROUGE adalah perbandingan antara banyaknya kata yang sama yang muncul pada kedua jenis ringkasan (ringkasan oleh sistem dan *gold summaries*) dibagi dengan banyaknya kata pada ringkasan oleh sistem. Selanjutnya *f1-score* dihitung berdasarkan *recall* dan *precision* yang didapatkan dengan formula sebagai berikut.

$$f1\ score = \frac{2\ (recall * precision)}{recall + precision}$$

3. Sistem yang Dibangun



Gambar 5 Alur Perancangan Sistem

3.1 Input Data

Penelitian ini menggunakan dataset INDOSUM [7]. Dataset tersebut merupakan dataset berbahasa indonesia untuk kasus peringkas teks otomatis, berisi kumpulan teks berita beserta ringkasannya (*gold summaries*). Dataset tersebut berisi 19 ribu pasangan artikel berita dan ringkasannya. Artikel-artikel pada dataset tersebut terbagi atas enam kategori berita, yaitu hiburan, inspirasi, olahraga, industri hiburan, *headline*, dan teknologi. Penelitian ini hanya mengambil 10 ribu sampel acak dari dataset tersebut, dikarenakan keterbatasan perangkat yang digunakan untuk menjalankan sistem. Lalu penelitian ini juga tidak mempertimbangkan kategori-kategori berita tersebut, artinya semua kategori berita dianggap sama.

3.2 Preprocessing

Preprocessing data dibutuhkan untuk membersihkan data dan mempersiapkan data supaya siap diolah dan masuk ke dalam sistem. *Preprocessing* yang dilakukan pada penelitian ini adalah *text cleaning* yang meliputi *lowercasing*, penghapusan simbol dan tanda baca, penghapusan karakter kosong dan penghapusan karakter numerik. Untuk memberikan tanda awal dokumen dan akhir dokumen, ditambahkan token **'startsum'** pada awal dokumen dan token **'endsum'** pada akhir dokumen. Selanjutnya dilakukan *stopwords removal* dan *stemming*. Proses *stopwords removal* dan *stemming* dilakukan untuk menguji mana hasil yang lebih baik jika dataset dilakukan proses keduanya, tanpa dilakukan proses keduanya, dan hanya dilakukan proses *stemming*. Maka dari itu, terdapat tiga skenario yang dilakukan pada penelitian ini. Pembagian skenario pada penelitian ini bertujuan untuk melihat bagaimana pengaruh dari teknik prapemrosesan *stemming* dan *stopwords removal* terhadap performansi peringkasan teks otomatis. Hasil dari perbandingan antara ketiga skenario ini diharapkan dapat menjadi landasan ide untuk dikembangkan dalam penelitian di masa yang akan datang.

Proses *text cleaning* diimplementasikan menggunakan bahasa pemrograman Python dan memanfaatkan *library* NLTK. *Stemming* yang dilakukan pada penelitian ini menggunakan *library* dari *sastrawi* menggunakan bahasa Python. Contoh hasil dari *stemming* ini seperti, kata "menjadikan" menjadi "jadi", kata "penelitian" menjadi "teliti", kata "memberikan" menjadi "beri", dan sebagainya. Selanjutnya *stopwords removal* juga diimplementasikan menggunakan *library* sastrawi. Kata-kata yang dihapus pada proses ini adalah seperti kata "yang", "di", "ke", dan sebagainya.

3.3 LSTM Training

Sebelum masuk ke proses *training*, dataset terlebih dahulu dibagi menjadi data latih dan data uji. Proporsi untuk data latih adalah sebanyak 80% atau 8 ribu data dari dataset dan data uji sebanyak 20% atau 2 ribu data dari total dataset. Selanjutnya setelah data dibagi menjadi data latih dan data uji, masing-masing kelompok data tersebut dilakukan *embedding* atau vektorisasi. Langkah pertama yang dilakukan

untuk vektorisasi ini adalah mengubah semua kata (*vocabulary*) pada dataset menjadi bentuk *integer*. Kemudian data dalam bentuk *integer* tersebut dikodekan menjadi *vector* menggunakan *embedding layer* dari *Library Keras* dengan dimensi 200.

Metode yang digunakan adalah *Long Short Term Memory* (LSTM). LSTM merupakan salah satu modifikasi dari metode *Recurrecnt Neural Network* (RNN). Metode RNN telah terbukti dapat menyelesaikan kasus peringkasan teks. RNN juga dinilai baik untuk kasus ini karena RNN memproses suatu *input* yang merupakan *output* dari proses RNN sebelumnya atau dapat dikatakan *sequence based*. Salah satu pengembangan dari RNN yaitu LSTM. LSTM dinilai dapat bekerja lebih baik daripada RNN dikarenakan LSTM memiliki arsitektur yang mampu mengingat dan melupakan informasi yang akan diproses. Artinya LSTM dapat menilai apakah suatu informasi tersebut penting untuk dijadikan sebagai *input* atau tidak. LSTM juga memiliki mekanisme yang disebut sebagai *attention*. Mekanisme ini membantu proses LSTM agar *output* yang dihasilkan memiliki keterkaitan erat dengan *output* yang sebelumnya. Model LSTM dalam kasus ini dibangun dalam model *encoder-decoder* dengan menggunakan 3 layer LSTM.

3.4 Inference

Inference merupakan proses untuk melakukan prediksi atau melakukan peringkasan oleh model yang sudah dilatih. Proses yang dilakukan pada tahap ini adalah memasukkan token **'sumstart'** sebagai *input* lalu *decoder* mengeluarkan probabilitas kata selanjutnya, lalu sistem memilih kata dengan probabilitas tertinggi untuk selanjutnya kata tersebut menjadi *input* lagi. Proses tersebut terus berulang hingga *inference* menemukan token **'endsum'**.

3.5 Evaluasi

Proses evaluasi menggunakan metode ROUGE. Model ROUGE yang digunakan adalah model ROUGE-1, yaitu model ROUGE yang menggunakan *unigram* untuk meilhat kemiripan ringkasan hasil model dengan ringkasan asli (*gold summaries*). ROUGE-1 digunakan karena penelitian ini ingin menilai kata-kata yang dihasilkan dari proses peringkasan jika dilihat dari kata per kata (individu) bukan sebagai *sequence*. Cara kerja model ROUGE untuk mengevaluasi sistem yang dibangun adalah untuk setiap data uji dihitung kata-kata yang sama yang muncul dalam ringkasan yang dibuat oleh sistem dan *gold summaries* secara *unigram* atau kata per kata (*overlap*). Selanjutnya dihitung *recall*, *precision*, dan *f1-score*.

Untuk menghitung *recall*, *precision*, dan *f1-score*, hasil ringkasan terbagi menjadi dua, yaitu hasil ringkasan yang dibuat oleh sistem yang selanjutnya dinamakan *candidate*, lalu ringkasan yang berasal dari dataset (*gold summaries*) yang selanjutnya dinamakan *reference*. Lalu terdapat kata-kata yang sama-sama muncul pada *reference* dan *candidate* yang selanjutnya dinamakan kata *overlapping*.

Perhitungan yang dilakukan untuk mendapatkan nilai *recall* adalah jumlah kata-kata yang *overlap* dibagi dengan banyaknya kata pada *reference*, sedangkan *precision* adalah pembagian antara kata-kata yang *overlap* dengan banyaknya kata pada *candidate*. Selanjutnya agar evaluasi lebih ideal, maka dilakukan perhitungan *f1-score*. Perhitungan evaluasi tersebut diimplementasi kepada seluruh data uji (2 ribu data), artinya masing-masing data uji menyimpan nilai evaluasinya masing-masing. Selanjutnya untuk mengukur performansi secara keseluruhan, terdapat perhitungan rata-rata nilai ROUGE dan maksimum nilai ROUGE. Proses menghitung rata-rata adalah dengan menjumlahkan seluruh nilai ROUGE pada data untuk masing-masing metrik (*recall*, *precision*, dan *f1-score*) lalu dibagi dengan banyaknya data uji. Untuk perhitungan nilai maksimum adalah dengan mengambil nilai ROUGE paling besar dari seluruh data uji untuk masing-masing kategori metrik (*recall*, *precision*, dan *f1-score*).

3.6 Skenario Penelitian

Pada Tugas Akhir ini, penulis mendefinisikan tiga skenario. Pada skenario pertama, penulis mengimplementasikan *stemming* dan *stopwords removal* pada dataset saat melakukan *preprocessing*. Skenario kedua penulis hanya mengimplementasikan *stemming* dan tanpa mengimplementasikan *stopwords removal*. Lalu pada skenario ketiga, penulis tidak mengimplementasikan *stemming* dan *stopwords removal*. Mekanisme yang digunakan untuk melakukan *stemming* dan *stopwords* removal adalah menggunakan *library Sastrawi* Python.

Penulis mendefinisikan tiga skenario ini adalah untuk melihat bagaimana pengaruh dari *stemming* dan *stopwords removal* pada sistem peringkasan teks ini. Jika terdapat pengaruh yang cukup signifikan ataupun tidak terdapat pengaruh, maka diharapkan hasil perbandingan ini dapat menjadi acuan bagi penelitian selanjutnya untuk mempertimbangkan apakah akan menggunakan *stemming* dan *stopwords* removal atau tidak sama sekali.

4. Hasil dan Pembahasan

4.1 Hasil Pengujian Sistem yang Dibangun

Pada bagian ini akan dijelaskan mengenai hasil dari sistem yang dibangun dengan menerapkan tiga skenario percobaan. Perbedaan pada masing-masing skenario terdapat pada bagian *preprocessing*. Pada skenario pertama, penulis mengimplementasikan *stemming* dan *stopwords removal* pada dataset saat melakukan *preprocessing*. Skenario kedua penulis hanya mengimplementasikan *stemming* dan tanpa mengimplementasikan *stopwords removal*. Lalu pada skenario ketiga, penulis tidak mengimplementasikan *stemming* dan *stopwords* removal. Tujuan membandingkan ketiga skenario tersebut adalah untuk melihat apakah terdapat pengaruh dari *stemming* dan *stopwords removal* terhadap sistem peringkasan yang dibangun. Hasil dari ketiga skenario tersebut akan dievaluasi menggunakan ROUGE-1 dengan penghitungan *recall*, *precision*, dan *f1-score*. Untuk membandingkan hasil ketiga skenario, metrik yang digunakan adalah *f1-score* rata-rata karena dianggap representatif dari sistem dan seluruh data uji.

Semua skenario dilatih dengan spesifikasi yang sama, kecuali pada bagian *text_len* dan *summ_len*. Kedua variabel tersebut merupakan variabel untuk mendefinisikan panjang maksimum teks berita dan ringkasan yang dilatih. *Text_len* dan *summ_len* pada tiap skenario didefinisikan berdasarkan jumlah kata maksimum pada suatu artikel dalam keseluruhan dataset setelah *preprocessing*. *Text_len* dan *summ_len* pada skenario 1 masing-masing adalah 894 kata dan 60 kata, untuk skenario 2 masing-masing 1144 kata dan 77 kata, lalu untuk skenario 3 sama dengan skenario 2. Hal tersebut disebabkan karena pada skenario 2 dan 3 tidak diimplementasikan *stopwords removal* sehingga tidak terdapat pengurangan kata, sedangkan pada skenario 1 terdapat pengurangan kata, maka dari itu *text_len* dan *summ_len* pada skenario 1 lebih sedikit.

Selain kedua variabel tersebut, semua variabel atau parameter sistem didefinisikan sama sehingga tidak terdapat perbedaan perlakuan untuk setiap skenario. Berikut merupakan kondisi yang diberlakukan untuk pengujian ketiga skenario tersebut.

1. *Embedding_dim* = 200
   Merupakan variabel untuk mendefinisikan berapa besaran dimensi yang digunakan untuk proses *embedding*. Angka 200 untuk *embedding_dim* dipilih berdasarkan percobaan kepada sistem dengan mempertimbangkan kapasitas perangkat yang digunakan agar proses pengujian tidak terlalu lama.
2. *Latent_dim* = 200
   Merupakan variabel untuk mendefinisikan besaran dimensi data untuk dikompresi agar menghindari *bottleneck* ketika data diproses. Angka 200 untuk *latent_dim* dipilih berdasarkan percobaan kepada sistem dengan mempertimbangkan kapasitas perangkat yang digunakan agar proses pengujian tidak terlalu lama.
3. *Epoch* = 20
   Epoch adalah perulangan yang dilakukan dalam pelatihan model. Satu *epoch* artinya data dilatih satu kali. *Epoch* dilakukan sebanyak 20 kali agar proses *training* tidak terlalu lama dan mengakibatkan hasil yang terlalu *overfitting.*
4. *Batchsize* = 64
   Batchsize adalah pengelompokan data dari total data latih untuk dilatih dalam satu waktu. Fungsi dari menggunakan *batchsize* ini adalah agar proses pelatihan menjadi lebih cepat. *Batchsize* diatur sebanyak 64 untuk membagi agar dalam satu *epoch*, hanya terdapat 125 langkah latihan. Jika tidak, maka dalam satu *epoch* akan terdapat 2000 langkah, yang mana hal tersebut akan membuat proses latihan menjadi sangat lama.
5. *Dropout* = 0.4
   Dropout merupakan variabel yang menyatakan berapa unit yang akan di-*drop* pada *transformasi linear* pada input. *Dropout* diharapkan dapat mengurangi *overfitting*. Angka 0.4 untuk *dropout* dipilih berdasarkan percobaan kepada sistem.
6. *Recurrent_dropout* = 0.4
   Recurrent Dropout merupakan variabel yang menyatakan berapa unit yang akan di-*drop* pada *transformasi linear* pada *state* data tersebut diproses. *Recurrent_dropout* juga diharapkan dapat mengurangi *overfitting.* Angka 0.4 untuk recurrent_*dropout* dipilih berdasarkan percobaan kepada sistem.
7. *Activation function = softmax*
   Activation function merupakan sebuah fungsi yang digunakan untuk membuat proses menjadi non-linear. *Activation function* yang dipilih adalah *softmax* karena dinilai bagus untuk kasus *multiclass classification.*

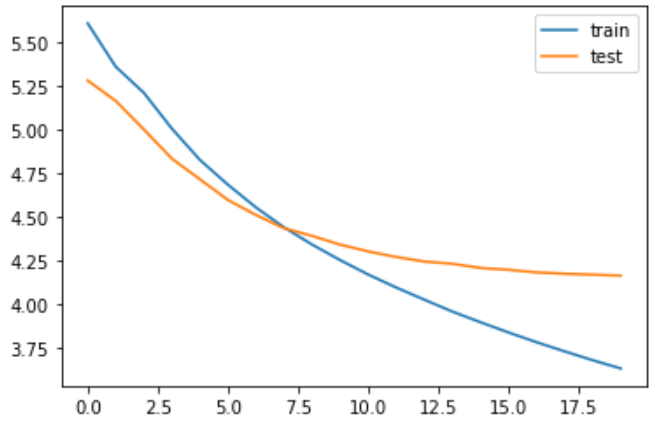
8. *Optimizer = rmsprop*
   Optimizer adalah suatu algoritma yang berfungsi untuk memperbarui bobot dari *network* secara iterative. Pada kasus ini *optimizer* yang digunakan adalah *rmsprop* karena dinilai sebagai salah satu *optimizer* terbaik sebagai *adaptive optimizer*.
9. *Loss = spars_categorical_crossentropy*
   Loss function berfungsi untuk melihat bagaimana performansi sistem ketika *training* pada setiap iterasinya. *Spars_categorical_entropy* dipilih agar hasil prediksi sistem lebih ditoleransi, artinya hasil prediksi tidak harus sama persis tetapi boleh hanya sekedar mendekati.

Semua pengujian dilakukan dengan perangkat keras yang sama, dengan spesifikasi sebagai berikut.

1. Nama perangkat : Laptop Asus Zenbook UX410UQ
2. RAM : 8 Gb
3. GPU : Nvidia Geforce 940mx
4. CPU : Intel core i7 gen 7
5. *Software* : Google Colaboratory

4.1.1 Hasil Pengujian Skenario 1

Sebagaimana yang telah dijelaskan di atas, bahwa skenario pertama ini adalah dimana penulis mengimplementasikan *stemming* dan *stopwords removal* pada *preprocessing* data. Hasil pengujian untuk skenario 1 ini dapat dilihat pada gambar 6 dan tabel 1 di bawah.



Gambar 6 Grafik Loss Function Skenario 1

Tabel 1 Hasil ROUGE Skenario 1

Skenario 1	ROUGE-1 (Average)	ROUGE-1 (Maximum)
F1-Score	0.08612	0.45977
Precision	0.08495	0.48781
Recall	0.08890	0.54054

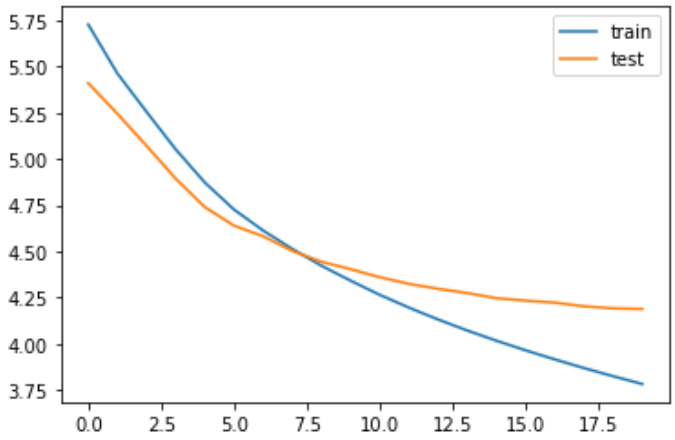
Berdasarkan hasil di atas, diketahui bahwa untuk skenario pertama ini mendapatkan hasil rata-rata *F1-score* untuk semua data uji sebesar 0.08612 dan nilai maksimum *F1-score* sebesar 0.45977. Nilai tersebut dapat dikatakan sangat rendah. Hal tersebut disebabkan karena dengan diimplementasikannya *stopwords removal*, maka panjang kata dalam suatu data (artikel berita) menjadi berkurang. Dengan dihapusnya *stopwords* hal ini memungkinkan *stopwords* yang memiliki peran penting dalam suatu kalimat dapat terhapus. Maka dari itu hasil peringkasan dengan skenario ini sangat rendah. Berikut ini contoh hasil peringkasan dengan skenario 1.

Teks berita :
rimanews sidik kpk periksa mantan ketua dpr ade komarudin bagai saksi kasus duga korupsi e ktp gedung kpk jakarta hari sapa akrab ade komarudin tiba pukul wib langsung masuk gedung kpk jalan periksa lima menit kemudian mantan anggota dpr fraksi partai golkar periode harahap susul mau periksa kata wartawan kpk panggil jumlah anggota dpr mantan anggota dpr kait korupsi e ktp rujuk terang mantan bendahara umum partai demokrat mohammad nazaruddin ketua dpr setya novanto gubernur jawa tengah juga mantan anggota komisi ii dpr ganjar pranowo turut kpk lebih saksi panggil periksa kait kasus kasus kpk tetap sangka mantan direktur jenderal duduk catat sipil kemendagri irman mantan direktur kelola informasi administrasi duduk ditjen dukcapil kemendagri sekaligus jabat buat komitmen sugiharto irman sugiharto jerat

pasal ayat pasal uu nomor tahun bagaimana ubah uu nomor tahun berantas tindak pidana korupsi jo pasal ayat jo pasal ayat kuhp irman duga gelembung harga dalam perkara manfaat wenang bagai kuasa buat anggar dasar hitung badan awas uang bangun bpkp rugi negara akibat kasus adalah rp triliun gelembung harga total nilai anggar rp triliun	
Ringkasan (gold summaries) : sidik kpk periksa mantan ketua dpr ade bagai saksi kasus duga korupsi e ktp gedung kpk jakarta hari belum kpk panggil jumlah anggota dpr mantan anggota dpr kait korupsi e ktp lebih saksi panggil periksa kait kasus	**Ringkasan (hasil dari sistem) :** kpk panggil ketua dpr setya novanto panggil bagai sangka kasus duga korupsi proyek e ktp setya novanto kpk periksa bagai saksi sangka kasus duga korupsi e ktp setya novanto bagai sangka kasus duga korupsi proyek e ktp setya novanto bagai sangka kasus duga korupsi proyek e ktp setya novanto kpk periksa bagai sangka kasus e ktp

4.1.2 Hasil Pengujian Skenario 2

Berikut merupakan hasil pengujian skenario 2, yaitu dengan mengimplementasikan *stemming*, tetapi tidak mengimplementasikan *stopwords removal*. Dapat dilihat pada gambar 7 dan tabel 2.



Gambar 7 Grafik Loss Function Skenario 2

Tabel 2 Hasil ROUGE Skenario 2

Skenario 2	ROUGE-1 (Average)	ROUGE-1 (Maximum)
F1-Score	0.13846	0.51429
Precision	0.14057	0.50943
Recall	0.13787	0.51923

Berdasarkan hasil di atas, diketahui bahwa untuk skenario pertama ini mendapatkan hasil rata-rata *F1-score* untuk semua data uji sebesar 0.13846 dan nilai maksimum *F1-score* sebesar 0.51429. Hasil pada skenario 2 ini cukup baik, bahkan jika dibandingkan dengan skenario 1, hasil pada skenario 2 ini meningkat sangat signifikan. Maka dari itu dapat dinilai bahwa *stopwords removal* sangat mempengaruhi performansi dari sistem. Berikut ini contoh hasil peringkasan dengan skenario 2.

Teks berita : rimanews sidik kpk periksa mantan ketua dpr ade komarudin bagai saksi untuk kasus duga korupsi ada e ktp di gedung kpk jakarta hari ini sapa akrab ade komarudin tiba sekitar pukul wib dan langsung masuk gedung kpk untuk jalan periksa lima menit kemudian mantan anggota dpr dari fraksi partai golkar periode harahap susul ya mau periksa kata kepada wartawan kpk sudah panggil jumlah anggota dpr dan mantan anggota dpr kait dengan korupsi e ktp rujuk pada terang mantan bendahara umum partai demokrat mohammad nazaruddin seperti ketua dpr setya novanto gubernur jawa tengah yang juga mantan anggota komisi ii dpr ganjar pranowo turut kpk ada lebih dari saksi sudah panggil untuk periksa dalam kait dengan kasus ini dalam kasus ini kpk sudah tetap dua sangka yaitu mantan direktur jenderal duduk dan catat sipil kemendagri irman dan mantan direktur kelola informasi administrasi duduk ditjen dukcapil kemendagri sekaligus jabat buat komitmen sugiharto irman dan sugiharto jerat dengan pasal ayat atau pasal uu nomor tahun bagaimana ubah dengan uu nomor tahun tentang berantas tindak pidana korupsi jo pasal ayat ke jo pasal ayat kuhp irman duga gelembung harga dalam perkara dengan manfaat wenang bagai kuasa buat anggar dasar hitung badan awas uang dan bangun bpkp rugi negara akibat kasus ini adalah rp triliun karena gelembung harga dari total nilai anggar rp triliun	
Ringkasan (gold summaries) : sidik kpk periksa mantan ketua dpr ade bagai saksi untuk kasus duga korupsi ada e ktp di gedung kpk jakarta hari ini belum kpk sudah panggil jumlah anggota dpr dan mantan	**Ringkasan (hasil dari sistem) :** kpk akan periksa bagai sangka kasus duga korupsi proyek e ktp setya novanto dalam sidang lanjut sidang lanjut kasus duga korupsi e ktp setya novanto setnov kata juru bicara

anggota dpr kait dengan korupsi e ktp sudah lebih dari saksi sudah panggil untuk periksa dalam kait dengan kasus ini	kpk febri diansyah kata bahwa pihak akan segera segera segera periksa bagai sangka kasus duga korupsi proyek e ktp setya novanto dan sangka duga korupsi proyek e ktp

4.1.3 Hasil Pengujian Skenario 3

Berikut ini merupakan hasil pengujian skenario 3, yaitu tanpa mengimplementasikan *stemming* dan *stopwords removal*. Dapat dilihat pada gambar 8 dan tabel 3.

Gambar 8 Grafik Loss Function Skenario 3

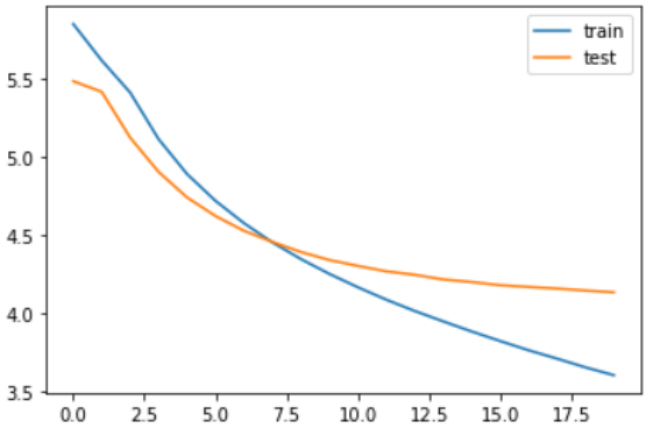


Table 3 Hasil ROUGE Skenario 3

Skenario 3	ROUGE-1 (Average)	ROUGE-1 (Maximum)
F1-Score	0.13826	0.50485
Precision	0.14211	0.5
Recall	0.13595	0.5089

Berdasarkan hasil di atas, diketahui bahwa untuk skenario pertama ini mendapatkan hasil rata-rata *F1-score* untuk semua data uji sebesar 0.13826 dan nilai maksimum *F1-score* sebesar 0.50485. Jika dilihar berdasarkan hasil rata-rata *f1-score* nya, maka skenario ini juga dinilai mendapatkan hasil yang cukup baik. Jika dibandingkan dengan skenario 2, skenario 3 ini mendapatkan hasil yang sangat sedikit lebih rendah daripada skenario 2. Seperti yang dijelaskan sebelumnya, bahwa perbedaan skenario 2 dan skenario 3 terletak pada implementasi *stemming*. Skenario 2 mengimplementasikan *stemming*, sedangkan skenario 3 tidak, dan keduanya sama-sama tidak mengimplementasikan *stopwords removal*. Dari sini dapat disimpulkan bahwa *stemming* memungkinkan untuk mempengaruhi performansi sistem, tetapi tidak signifikan. Hal tersebut disebabkan karena *stemming* tidak mengurangi kata sedikitpun, hanya mengubah bentuk kata kembali ke kata dasar, hal ini dinilai pengubahan kata ke dalam kata dasar di dalam suatu kalimat tidak mengubah banyak makna dari kalimat tersebut. Maka dari itu hasil performansi pada skenario 2 dan 3 sangat mirip. Berikut ini contoh hasil peringkasan dengan skenario 3.

Teks berita : rimanews penyidik kpk memeriksa mantan ketua dpr ade komarudin sebagai saksi untuk kasus dugaan korupsi pengadaan e ktp di gedung kpk jakarta hari ini sapaan akrab ade komarudin tiba sekitar pukul wib dan langsung masuk gedung kpk untuk menjalani pemeriksaan lima menit kemudian mantan anggota dpr dari fraksi partai golkar periode harahap menyusul ya mau diperiksa kata kepada wartawan kpk sudah memanggil sejumlah anggota dpr dan mantan anggota dpr berkaitan dengan korupsi e ktp merujuk pada keterangan mantan bendahara umum partai demokrat mohammad nazaruddin seperti ketua dpr setya novanto gubernur jawa tengah yang juga mantan anggota komisi ii dpr ganjar pranowo menurut kpk ada lebih dari saksi sudah dipanggil untuk diperiksa dalam kaitannya dengan kasus ini dalam kasus ini kpk sudah menetapkan dua tersangka yaitu mantan direktur jenderal kependudukan dan catatan sipil kemendagri irman dan mantan direktur pengelola informasi administrasi kependudukan ditjen dukcapil kemendagri sekaligus pejabat pembuat komitmen sugiharto irman dan sugiharto dijerat dengan pasal ayat atau pasal uu nomor tahun sebagaimana diubah dengan uu nomor tahun tentang pemberantasan tindak pidana korupsi jo pasal ayat ke jo pasal ayat kuhp irman diduga harga dalam perkara dengan memanfaatkan kewenangannya sebagai kuasa pembuat anggaran berdasarkan perhitungan badan pengawasan keuangan dan pembangunan bpkp kerugian negara akibat kasus ini adalah rp triliun karena penggelembungan harga dari total nilai anggaran rp triliun	
Ringkasan (gold summaries) :	**Ringkasan (hasil dari sistem) :**

penyidik kpk memeriksa mantan ketua dpr ade sebagai saksi untuk kasus dugaan korupsi pengadaan e ktp di gedung kpk jakarta hari ini sebelumnya kpk sudah memanggil sejumlah anggota dpr dan mantan anggota dpr berkaitan dengan korupsi e ktp sudah lebih dari saksi sudah dipanggil untuk diperiksa dalam dengan kasus ini	kpk kembali memanggil ketua umum partai golkar setya novanto sebagai tersangka kasus dugaan korupsi proyek e ktp setya novanto di pengadilan tipikor jakarta kamis malam waktu setempat kpk akan menetapkan tersangka pemeriksaan sebagai saksi untuk tersangka kasus dugaan suap e ktp setya novanto dan kpk akan diperiksa sebagai saksi untuk tersangka kasus e ktp

4.1.4 Perbandingan Hasil Pengujian

Perbandingan hasil dari ketiga skenario pengujian dilakukan berdasarkan nilai *f1-score*. *F1-score* dipilih karena dinilai lebih baik dan lebih representatif untuk mengevaluasi suatu model karena melibatkan *recall* dan *precision* sekaligus. Berikut hasil perbandingan yang didapatkan.

Table 4 Perbandingan Hasil Ketiga Skenario

	F1-Score ROUGE-1 (average)	**F1-Score ROUGE-1 (maximum)**
Skenario 1	0.08612	0.45977
Skenario 2	0.13846	0.51429
Skenario 3	0.13825	0.50485

Berdasarkan tabel di atas, terlihat bahwa skenario yang mendapatkan hasil terbaik adalah skenario 2 baik itu berdasarkan nilai *f1-score* rata-rata nya maupun nilai *f1-score* maksimumnya.

Jika diobservasi lebih lanjut, dapat diidentifikasi bahwa nilai evaluasi *ROUGE-1* pada skenario ke-2 dan ke-3 tidak berbeda secara signifikan, sedangkan jika dibandingkan dengan skenario ke-1 nilainya sangat signifikan berbeda. Artinya faktor *stopwords* removal sangat signifikan memberikan pengaruh kepada hasil pengujian, sedangkan untuk stemming juga memberikan pengaruh, tetapi tidak terlalu signifikan.

Stopwords removal memberikan peranan sangat besar pada pengujian karena jika pada dataset diimplementasikan *stopwords removal*, maka hal tersebut akan menghapus banyak *stopwords* yang dikhawatirkan stopwords tersebut sebenarnya sangat penting bagi teks untuk memberikan makna kalimat yang lebih lengkap.

Stemming pada pengujian peringkas otomatis ini memberikan sedikit pengaruh, terbukti pada hasil pengujian skenario 2 dan 3 yang mendapatkan hasil tidak jauh berbeda, dengan nilai skenario 2 sedikit lebih baik. Pengaruh yang diberikan oleh *stemming* adalah pengaruh yang baik, artinya dengan mengimplementasikan *stemming* maka sistem mendapatkan performansi lebih baik. Hal ini disebabkan karena aturan Bahasa Indonesia yang memungkinkan satu kata dasar memiliki beberapa kata bentukan dengan penambahan beberapa imbuhan yang membuat kata-kata tersebut dapat berbeda makna meskipun kata dasarnya sama. Misalnya kata "memperlakukan", "berlaku", "perlakuan", dan "laku". Semua kata tersebut setelah di-*stemming*, maka akan menjadi satu kata yang sama, yaitu "laku". Dengan begitu, jumlah *vocab* pada sistem yang mengimplementasikan *stemming* berkurang, hal ini membuat sistem tidak bingung untuk memilih antara kata "memperlakukan", "berlaku", "perlakuan", dan "laku". Sedangkan pada skenario 3 yang tidak mengimplementasikan *stemming* semua bentukan kata berimbuhan dari kata dasar dianggap berbeda. Maka sebaliknya ini akan membuat sistem menjadi bingung.

Namun, meskipun secara nilai evaluasi skenario 2 mendapatkan hasil yang sedikit lebih baik, hasil ringkasan pada skenario 2 terkesan tidak natural karena mengubah bentuk kata ke kata dasar. Sedangkan pada skenario ke-3 hasil yang didapatkan lebih natural mendekati hasil ringkasan alami oleh manusia. Contoh kalimat "saya akan berlaku adil" terdengar lebih baik daripada "saya akan laku adil".

5. Kesimpulan

Penelitian ini dikerjakan dengan tiga skenario. Bagian yang membedakan masing-masing skenario adalah pada bagian *preprocessing*-nya, yang mana pada skenario 1 diimplementasikan *stemming* dan *stopwords removal*, pada skenario 2 diimplementasikan *stemming* tanpa *stopwords removal*, dan pada skenario 3 tidak diimplementasikan keduanya. Semua skenario tersebut dilatih dengan metode yang sama, dan kondisi yang sama. Berdasarkan hasil pada Bab IV, maka kesimpulan yang dapat diambil pada penelitian ini adalah pengujian terbaik yang didapatkan adalah pengujian dengan skenario 2, yaitu dengan mengimplementasikan *stemming* tanpa *stopwords removal* dengan nilai evaluasi ROUGE-1 0.13846. Tetapi hasil tersebut hanya berbeda sedikit daripada skenario 3 yang tidak mengimplementasikan *stemming* dan *stopwords removal*. Artinya *stemming* hanya memberikan sedikit pengaruh lebih baik kepada sistem. Sedangkan jika skenario 2

dan 3 dibandingkan dengan skenario 1, maka hasilnya jauh berbeda. Skenario 1 mendapatkan hasil yang jauh lebih rendah. Artinya *stopwords removal* sangat memberikan pengaruh besar terhadap performansi sistem.

Daftar Pustaka

[1] S. Gupta, S. K. Gupta. 2019. Abstractive Summarization: An overview of the state of the art. ExpertSystems with Applications. 121:49 – 65.
[2] C. Khatri, G. Singh, N. Parikh. 2018. Abstractive and Extractive Text Summarization using Document Context Vector and Recurrent Neural Networks. arXiv:1807.08000.
[3] R. Adelia, Suyanto, U.N. Wisesty. 2019. Indonesian Abstractive Text Summarization Using Bidirectional Gated Recurrent Unit. Procedia Computer Science, 157, pp.581-588.
[4] A. Najibullah. 2015. Indonesian Text Summarization based on Naïve Bayes Method. Proceeding of the International Seminar and Conference 2015: The Golden Triangle (Indonesia-India-Tiongkok) Interrelations in Religion, Science, Culture, and Economic, Semarang, Indonesia, p. 12.
[5] Mustaqhfiri, M., Abidin, Z., Kusumawati, R.,2012, Peringkasan Teks Otomatis Berita Berbahasa Indonesia Menggunakan Metode Maximum Marginal Relevance. DOI: 10.18860/mat.v0i0.1578
[6] M. Fachrurrozi, N. Yusliani, R. U. Yoanita. 2013. Frequent Term based Text Summarization for Bahasa Indonesia. Proceedings of the International Conference on Innovations in Engineering and Technology, Bangkok, Thailand, p. 3.
[7] K. Kurniawan, S. Louvan. 2018. Indosum: A New Benchmark Dataset for Indonesian Text Summarization. International Conference on Asian Language Processing (IALP).
[8] K. Cho, dkk. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. arXiv:1406.1078.
[9] K. Ivanedra, M. Mustikasari. 2018. Implementasi Metode Recurrent Neural Network pada Text Summarization dengan Teknik Abstraktif. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK). 6(4) : 377-382.
[10] R. Nallapati, B. Xiang, B. Zhou. 2016. Sequence-To-Sequence Rnns For Text Summarization. Proceedings of The 20th SIGNLL. DOI: 10.18653/v1/K16-1028
[11] C. Y. Lin. 2004. Rouge: A package for automatic evaluation of summaries. Text Summarization Branches out: Proceedings of the ACL-04 Workshop, vol. 8. Barcelona, Spain.
[12] D. T. Massandy, M. L. Khodra. 2014. Guided Summarization for Indonesian News Articles. in 2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA), Aug. 2014, pp. 140–145.