INDONESIAN ABSTRACTIVE NEWS SUMMARIZATION BERBASIS *DEEP LEARNING* DENGAN METODE *SEQUENCE-TO-SEQUENCE LONG SHORT-TERM MEMORY*

Skripsi

diajukan untuk memenuhi salah satu syarat
untuk memperoleh gelar Sarjana Komputer
Program Studi Teknik Informatika

Oleh:

Moh Minhajul Mubarok
4611416009

JURUSAN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI SEMARANG
2021

PERNYATAAN

Saya menyatakan bahwa skripsi saya yang berjudul "*Indonesian Abstractive News Summarization* berbasis *Deep Learning* dengan Metode *Sequence-to-Sequence Long Short-Term Memory*" disusun atas dasar penelitian saya dengan arahan dosen pembimbing. Sumber informasi atau kutipan yang berasal dari karya yang diterbitkan telah disebutkan dalam teks dan dicantumkan dalam daftar pustaka di bagian akhir skripsi ini. Dan saya menyatakan bahwa skripsi ini bebas plagiat dan apabila di kemudian hari terbukti terdapat plagiat dalam skripsi ini, maka saya bersedia menerima sanksi sesuai ketentuan perundang – undangan.

Semarang, 05 Februari 2021

Moh Minhajul Mubarok
NIM 4611416009

PERSETUJUAN PEMBIMBING

Nama : Moh Minhajul Mubarok

NIM : 4611416009

Program Studi : Teknik Informatika S1

Judul Skripsi : *Indonesian Abstractive News Summarization* berbasis *Deep Learning* dengan Metode *Sequence-to-Sequence Long Short-Term Memory*

Skripsi ini telah disetujui oleh pembimbing untuk diajukan ke sidang ujian skripsi Fakultas Matematika dan Ilmu Pengetahuan Alam pada:

Semarang, 05 Februari 2021

Dosen Pembimbing

Zaenal Abidin S.Si., M.Cs., Ph.D.
NIP 198205042005011001

PENGESAHAN

Skripsi dengan judul:

Indonesian Abstractive News Summarization berbasis *Deep Learning* dengan Metode *Sequence-to-Sequence Long Short-Term Memory*

disusun oleh:

Moh Minhajul Mubarok

4611416009

Telah dipertahankan di hadapan sidang Panitia Ujian Skripsi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang pada tanggal 05 Februari 2021.

Panitia:

Ketua

Dr. Sugianto, M.Si.
NIP 196102191993031001

Sekretaris

Dr. Alamsyah, S.Si., M.Kom.
NIP 197405172006041001

Penguji 1

Anggyi Trisnawan Putra, S.Si., M.Si.
NIP 198707062014041003

Penguji 2

Riza Arifudin, S.Pd., M.Cs.
NIP 198005252005011001

Anggota Penguji/Pembimbing

Zaenal Abidin S.Si., M.Cs., Ph.D.
NIP 198205042005011001

MOTTO DAN PERSEMBAHAN

MOTTO

“Jangan pernah berhenti belajar”.

PERSEMBAHAN

Skripsi ini saya persembahkan kepada:

- Kedua orang tua saya Bapak M Imron dan Ibu Siti Fatimah dan juga saudara saya M Misbahul Ubaid.
- Teman-teman saya di jurusan Ilmu Komputer, Fakultas MIPA, serta teman-teman di Universitas Negeri Semarang.
- Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu hingga terselesaikannya penulisan skripsi ini.
- Almamater, Universitas Negeri Semarang.

PRAKATA

Puji syukur penulis panjatkan kepada *Rabb* seluruh alam semesta Allah *Subhanahu wa ta'ala*, atas berkat rahmat, pertolongan, petunjuk, bimbingan serta kebaikan-Nya, penulis dapat menyelesaikan skripsi yang berjudul "*Indonesian Abstractive News Summarization* berbasis *Deep Learning* dengan Metode *Sequence-to-Sequence Long Short-Term Memory*". Shalawat serta salam tak lupa selalu terpanjatkan kepada junjungan kita makhluk paling mulia di langit dan di bumi, suri tauladan bagi kita semua, manusia yang membawa peradaban kepada zaman yang terang-benderang, *Rasulullah Muhammad Shallallahu alaihi wassalam*, keluarganya, sahabatnya dan para pengikutnya.

Penulis menyadari bahwa penulisan skripsi ini tidak akan selesai tanpa adanya dukungan serta bantuan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih kepada:

1. Prof. Dr. Fathur Rokhman, M.Hum., Rektor Universitas Negeri Semarang.
2. Dr. Sugianto, M.Si., Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
3. Dr. Alamsyah, S.Si., M.Kom., Ketua Jurusan Ilmu Komputer FMIPA Universitas Negeri Semarang yang telah meluangkan waktu, membantu, membimbing, mengarahkan.
4. Zaenal Abidin S.Si., M.Cs., Ph.D., Dosen Pembimbing saya, dosen yang selalu

menginspirasi saya akan luasnya dunia kemungkinan-kemungkinan apa saja yang bisa kita ambil.

5. Bapak dan Ibu Dosen Jurusan Ilmu Komputer yang telah memberikan bekal kepada peneliti dalam penyusunan skripsi ini.
6. Kedua Orang Tua saya Bapak M Imron dan Ibu Siti Fatimah yang selalu tanpa lelah mencurahkan kasih sayang, doa, dan dukungannnya.
7. Teman-teman saya di jurusan Ilmu Komputer, khususnya teman-teman ilkom angkatan 2016, yang telah memberikan semangat dan dukungannya.
8. Semua pihak yang telah membantu terselesaikannya skripsi ini yang tidak dapat penulis sebutkan satu persatu, terimakasih atas bantuannya.

Semoga skripsi ini dapat memberikan manfaat bagi pembaca di masa yang akan datang.

Semarang, 05 Februari 2021

Moh Minhajul Mubarok
NIM 4611416009

ABSTRAK

Mubarok, Moh Minhajul. 2020. *Indonesian Abstractive News Summarization* berbasis *Deep Learning* dengan Metode *Sequence-to-Sequence Long Short-Term Memory*. Skripsi, Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam

Kata Kunci: *automatic text summarization, deep learning, abstractive, natural language processing*

Perkembangan teknologi yang sangat cepat di era globalisasi sekarang ini menuntut setiap orang untuk dapat selalu *update* informasi terbaru. Namun disisi lain ledakan informasi yang sangat besar membuat kita semakin kesulitan untuk memahami semua berita sekaligus. Teknologi *automatic text summarization* menjadi salah satu teknologi alternatif untuk menyelesaikan masalah di atas. *Automatic text summarization* adalah cabang penting dalam ilmu *Natural Language Processing* (NLP) yang bertujuan untuk merepresentasikan dokumen yang panjang dengan cara mengkompres informasi sehingga dapat dengan cepat dipahami dan dibaca oleh pengguna. Berbagai metode telah dilakukan untuk mengatasi masalah *automatic text summarization* khususnya untuk objek berbahasa Indonesia diantaranya ada metode yang berbasis *extractive* menggunakan metode *term frequency, latent semantic analysis* dan *naive Bayes.* Untuk metode yang berbasis *abstractive* masih sedikit sekali penelitian yang menggunakan bahasa Indonesia sebagai objeknya. Hal tersebut menjadi masalah karena selain masih sedikitnya penelitian pada bidang tersebut juga hasil evaluasi yang dihasilkan menunjukkan hasil yang tidak terlalu tinggi. *Dataset* yang digunakanpun sebagian dari mereka masih menggunakan *dataset* lokal sehingga susah untuk menjadi pembanding terhadap penelitian-penelitian sejenis. Untuk mengatasi masalah *abstractive text summarization* berbahasa Indonesia, pada penelitian ini digunakan *automatic text summarization* dengan metode berbasis *deep learning* yaitu *sequence-to-sequence*. Pada penelitian ini juga digunakan *dataset* publik agar bisa dilakukan *benchmarking* untuk penelitian-penelitian selanjutnya. Metode yang digunakan untuk evaluasi adalah ROUGE, yaitu metode yang sering digunakan dalam penelitian *automatic text summarization*. Dalam penelitian ini didapatkan hasil yang lebih baik dibandingkan dengan penelitian sejenis yaitu pada ROUGE-1 memiliki hasil 0.5371 dan pada ROUGE-2 0.3757.

DAFTAR ISI

PERNYATAAN ii
PERSETUJUAN PEMBIMBING iii
PENGESAHAN iv
MOTTO DAN PERSEMBAHAN v
PRAKATA vi
ABSTRAK viii
DAFTAR ISI ix
DAFTAR TABEL xi
DAFTAR GAMBAR xii
DAFTAR LAMPIRAN xiii
BAB I PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Masalah 5
1.3. Batasan Masalah 5
1.4. Tujuan Penelitian 6
1.5. Manfaat Penelitian 6
1.6. Sistematika Penulisan 7
1.6.1. Bagian Awal Skripsi 7
1.6.2. Bagian Isi Skripsi 7
1.6.3. Bagian Akhir Skripsi 8
BAB II TINJAUAN PUSTAKA 10
2.1. *Automatic Text Summarization* 10
2.2. *Natural Language Processing* 13
2.3. *Deep Learning* 15
2.4. *Sequence-to-Sequence* 17
2.5. *Long Short-Term Memory (LSTM)* 20

2.6. *Recall-Oriented Understudy for Gisting Evaluation (ROUGE)* 22
2.7. *Attention Mechanism* 24
2.8. Penelitian Terkait 25
BAB III METODE PENELITIAN 29
3.1. Studi Literatur 30
3.2. Pemilihan Dataset 31
3.3. Perancangan Algoritma 32
3.3.1. Proses *Training* 33
3.3.2. Proses *Testing* 37
3.4. Pengembangan Sistem 37
3.5. Analisis Hasil 38
3.6. Penarikan Kesimpulan 38
BAB IV HASIL DAN PEMBAHASAN 40
4.1. Hasil Penelitian 40
4.1.1. Persiapan Dataset 41
4.1.2. Pre-Process Dataset 44
4.1.3. Vektorisasi Dataset 51
4.1.4. *Training Model* 53
4.1.5. Prediksi Ringkasan 64
4.1.6. Implementasi Sistem 68
4.2. Pembahasan 73
BAB V PENUTUP 77
5.1 Simpulan 77
5.2 Saran 79
DAFTAR PUSTAKA 80
LAMPIRAN 84

DAFTAR TABEL

Table 2.1. Hasil evaluasi ROUGE score penelitian terkait.....28

Table 4.1. Struktur Dataset.....42

Table 4.2. Perbandingan dataset asli dengan hasil *case folding*.....45

Table 4.3. Perbandingan dataset asli dengan hasil *removing punctuation*.....46

Table 4.4. Perbandingan dataset asli dengan hasil *word tokenization*.....48

Table 4.5. Perbandingan dataset asli dengan hasil pembangunan dataset.....50

Table 4.6. Contoh representasi vektor.....51

Table 4.7. Contoh data input.....65

Table 4.8. Perbandingan hasil output mesin dengan ringkasan asli manusia 66

Table 4.9. perbandingan hasil evaluasi model peneliti dengan penelitian sejenis.....73

DAFTAR GAMBAR

Gambar 2.1. Representasi deep learning menjadi graf.....17
Gambar 2.2. Mekanisme sequence-to-sequence.....20
Gambar 2.3. Cell LSTM.....21
Gambar 3.1. Flowchart Tahap Metode Penelitian.....29
Gambar 3.2. Flowchart alur metode peneliti.....33
Gambar 3.3. Model sequence-to-sequence.....36
Gambar 4.1. Flowchart Tahap Penelitian.....41
Gambar 4.2. Contoh Format Data untuk Penelitian Extractive.....44
Gambar 4.3. Struktur Model.....54
Gambar 4.4. Ilustrasi model sequence-to-sequnce.....54
Gambar 4.5. Contoh hasil pre-processing dataset.....55
Gambar 4.6. Ilustrasi encoder-decoder.....56
Gambar 4.7. Ilustrasi LSTM.....57
Gambar 4.8. Ilustrasi bidirectional network.....58
Gambar 4.9. Ilustrasi Multilayer Network.....60
Gambar 4.10. Ilustrasi kata-kata dalam encoder-decoder.....61
Gambar 4.11. Ilustrasi output berdasarkan kondisional input.....61
Gambar 4.12. ilustrasi beam search dengan width 3.....62
Gambar 4.13. sequence-to-sequence dengan attention mechanism.....63
Gambar 4.14. Flowchart alur prediksi model.....64
Gambar 4.15. Arsitektur aplikasi yang akan dibuat.....67
Gambar 4.16. Home page aplikasi peringkas berita.....69
Gambar 4.17. Hasil output aplikasi peringkas berita.....71

DAFTAR LAMPIRAN

Lampiran 1. Source code model sequence-to-sequence 85

BAB I

PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi yang sangat cepat di era globalisasi sekarang ini menuntut setiap orang untuk dapat selalu *update* informasi terbaru. Salah satu cara termudah untuk selalu bisa *update* informasi terbaru adalah dengan membaca berita online (Hutama et al., 2017). Namun disisi lain, selain kebutuhan informasi yang besar oleh masyarakat, perkembangan media informasi di dunia juga mengalami perkembangan yang sangat cepat. Seiring dengan semakin banyaknya pengguna internet di Indonesia banyak juga media informasi offline baik koran, majalah bergeser ke situs berita online. Untuk media online saja terdapat berbagai kanal informasi yang sering kita temui setiap hari, sebagai contoh: Kompas, Detik, Kumparan, IDN Times, CNN Indonesia, dan berbagai kanal informasi lainnya. Setiap satu kanal informasi saja setiap harinya dapat menyebarkan lebih dari ratusan berita dengan berbagai tema. Menurut Puspitasari et al (2012) lebih dari 2000 berita diterbitkan oleh 32 website berita dalam satu hari.

Ledakan informasi yang sangat besar membuat kita semakin kesulitan untuk memahami semua berita sekaligus. Membaca semua berita sekaligus akan sangat memakan banyak waktu, dan ini tidak sejalan dengan keinginan masyarakat saat ini yaitu dapat menerima semua informasi secara cepat dan akurat. Disisi lain manusia

hanya memiliki kecepatan membaca yang sangat terbatas, sehingga sangat sulit bagi kita untuk dapat memahami banyak berita secara singkat. Salah satu alternatif untuk dapat memahami banyak berita sekaligus adalah dengan membaca ringkasan dari berita, yang berisi poin-poin terpenting dalam berita tersebut.

Teknologi *automatic text summarizer* menjadi salah satu teknologi alternatif untuk menyelesaikan masalah di atas. Menurut Joshi et al (2019), *automatic text summarizer* adalah cabang penting dalam ilmu *natural language processing* (NLP) yang bertujuan untuk merepresentasikan dokumen yang panjang dengan cara mengkompres informasi sehingga dapat dengan cepat dipahami dan dibaca oleh pengguna. *Text summarizer* memungkinkan pengguna dapat dengan mudah memahami suatu berita yaitu hanya dengan membaca ringkasan yang merupakan bagian-bagian penting dalam sebuah berita.

Text summarizer dapat diimplementasikan dalam dua cara: *abstractive text summarization* dan *extractive text summarization.* Menurut Adelia et al (2019), *Abstractive summarization* adalah metode peringkasan text dengan cara memparafrasekan teks sehingga ringkasan yang dihasilkan dapat memiliki banyak *vocabulary* daripada teks aslinya. Sedangkan *extractive text summarization* adalah metode peringkasan teks dengan cara memilih kalimat mana yang paling penting tanpa kehilangan informasi dalam dokumen tersebut (Shirwandkar & Kulkarni, 2018). Tujuan dari *text summarization* adalah untuk menghasilkan ringkasan dari suatu dokumen atau teks. Ringkasan tersebut seharusnya mengandung informasi penting dan

memiliki panjang teks yang lebih kecil daripada dokumen asli.

Text summarization juga dapat dikategorikan menjadi dua jenis yaitu *single-document summarization* dan *multi-document summarization* tergantung pada jumlah input yang diberikan dari dokumen tersebut (Zajic et al., 2008). Selain itu, *text summarization* juga dapat dikelompokkan menjadi *generic* atau *query-based text summarization. Generic summarization* adalah peringkasan yang berdasarkan semua konten dari dokumen, sedangkan *query-based text summarization* adalah peringkasan yang meringkas dokumen berdasarkan *query* dari *user* (Annisa & Khodra, 2017). Dalam penelitian ini akan digunakan metode *abstractive text summarization* untuk *single* dan *generic* dokumen khususnya untuk peringkasan berita berbahasa Indonesia.

Berbagai metode tradisional *text summarization* yang berbasis *extractive* kebanyakan menggunakan rekayasa manusia dalam penerapannya, seperti misalnya, kombinasi fitur statistik dan linguistik seperti *term frequency*, perbandingan panjang dan posisi kalimat, dan berbagai metode lainnya. Sebagai contoh Vanderwende et al (2007), menggunakan metode SumBasic yaitu peringkasan *text* yang menggunakan frekuensi kata untuk merangking kalimat dan memilih kalimat utama sebagai ringkasan. Steinberger dan Jezek (2004), menggunakan metode *latent semantic analysis* (LSA) yang mana mengukur kesamaan antar kalimat berdasarkan persamaan *matrix semantic* yang sudah dibuat. Selanjutnya berbagai metode diperkenalkan seperti peringkasan teks dengan *naïve Bayes*, peringkasan teks berbasis *neural network*, dan berbagai metode lainnya.

Dalam berbagai masalah tertentu menurut Joshi et al (2019) metode berbasis *deep learning* telah mencapai akurasi yang cukup tinggi dalam berbagai masalah NLP. Seperti *question-answering, natural language understanding, sentiment analysis, text classification,* dan *language translation.* Selain itu Ivanedra & Mustikasari (2019) mencoba menerapkan *abstractive text summarization* berbasis *deep learning*, yang mana menerapkan metode *recurrent neural network* (RNN) dan *long short-term memory* (LSTM) dan dengan hasil yang memuaskan. Namun dalam penelitian tersebut Ivanendra menggunakan *dataset* yaitu berita yang berbahasa India dan Inggris sehingga masih belum diketahui apakah metode *deep learning* yang serupa dapat memberikan hasil yang serupa untuk *dataset* berbahasa Indonesia.

Menurut Kurniawan & Louvan (2018) terlepas dari beberapa penelitian *text summarization* berbahasa Indonesia tidak satupun dari mereka menggunakan *dataset* publik yang besar untuk melatih ataupun mengevaluasi metode mereka. Mereka hanya menggunakan *dataset* yang mereka kumpulkan sendiri dan tidak dipublikasikan. Sehingga sangat susah sekali untuk menggunakan semacam *benchmark* yang standard untuk *text summarization* berbahasa Indonesia. Pada tahun 2018, Kurniawan & Louvan mempublikasikan *dataset* publik (IndoSum) untuk *text summarization* berbahasa Indonesia yang bisa dijadikan *benchmark* untuk *text summarization* berbahasa Indonesia. *Dataset* yang dibuat terdiri dari 20k artikel berita dengan berbagai kategori dan dari berbagai media (CNN Indonesia, Kumparan, dan media lainnya).

Berdasarkan *benchmark dataset* di atas, Kurniawan & Louvan (2018) juga

sudah menerapkan *text summarization* dengan tipe *extractive* berbasis *neural* bernama NeuralSum dan menghasilkan hasil yang cukup bagus. Namun, karena tipe peringkasan yang digunakan adalah *extractive,* ringkasan yang dihasilkan kurang halus dan masih patah-patah. Dari permasalahan di atas peneliti, ingin menyelidiki *abstractive text summarization* berbahasa Indonesia berbasis *deep learning* dengan metode *sequence-to-sequence. Dataset* yang akan digunakan peneliti adalah IndoSum (Kurniawan & Louvan, 2018) yang tersedia secara publik dan bisa digunakan sebagai *benchmark text summarization* yang peneliti rancang dengan metode lain. Dari pemaparan di atas, peneliti mengambil judul yaitu: "*Indonesian Abstractive News Summarization* Berbasis *Deep Learning* dengan Metode *Sequence-to-Sequence Long Short-Term Memory*".

1.2. Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam penelitian ini adalah:

1. Bagaimana cara kerja algoritma *sequence-to-sequence* dalam penerapannya untuk *abstractive text summarization* berbahasa Indonesia.
2. Bagaimana hasil evaluasi kualitas *abstractive text summarization* berbahasa Indonesia dengan algoritma *sequence-to-sequence* ?

1.3. Batasan Masalah

Pada penelitian ini diperlukan batasan-batasan agar tujuan penelitian dapat

tercapai. Adapun batasan masalah yang dibahas dalam penelitian ini adalah:

1. Metode *text summarization* yang digunakan adalah *abstractive text summarization* dengan *single* dan *generic document.*
2. Data yang digunakan dalam penelitian ini adalah IndoSum (Kurniawan & Louvan, 2018) yang tersedia secara publik.
3. Algoritma *abstractive text summarization* yang digunakan adalah berbasis *deep learning* yaitu *sequence-to-sequence long short-term memory.*

1.4. Tujuan Penelitian

Penelitian ini memiliki beberapa tujuan, antara lain:

1. Mengetahui cara kerja *sequence-to-sequence* dalam proses *abstractive text summarization* berbahasa Indonesia.
2. Mengetahui kualitas ringkasan yang dihasilkan berdasarkan hasil evaluasi terhadap prediksi *abstractive text summarization* berbahasa Indonesia dengan algoritma *sequence-to-sequence.*
3. Mengasilkan *abstractive text summarization* berbahasa Indonesia yang dapat dijadikan *benchmarking* terhadap penelitian yang sejenis.

1.5. Manfaat Penelitian

Manfaat penelitian ini adalah sebagai berikut.

1. Memahami penerapan algoritma *sequence-to-sequence* dalam penelitian

abstractive text summarization berbahasa Indonesia.

2. Mengetahui hasil evaluasi kualitas ringkasan yang dihasilkan menggunakan metode berbasis *sequence-to-sequence*.
3. Memberikan kontribusi terhadap penelitian *abstractive text summarization* berbahasa Indonesia.
4. Dengan penerapan *abstractive text summarization* berbahasa Indonesia dengan metode *sequence-to-sequence* dapat membantu meringkas jutaan informasi yang ada di internet menjadi informasi yang efektif dan berkualitas.

1.6. Sistematika Penulisan

Sistematika penulisan berguna untuk memudahkan dalam memahami jalan pemikiran secara keseluruhan skripsi. Penulisan skripsi ini secara garis besar dibagi menjadi tiga bagian, yaitu sebagai berikut.

1.6.1. Bagian Awal Skripsi

Bagian awal skripsi terdiri dari halaman judul, halaman pengesahan, halaman pernyataan, halaman motto dan persembahan, abstrak, kata pengantar, daftar isi, daftar gambar, daftar tabel dan daftar lampiran.

1.6.2. Bagian Isi Skripsi

Bagian isi skripsi terdiri dari lima bab, yaitu sebagai berikut.

1. BAB 1: PENDAHULUAN

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah,

tujuan dan manfaat penelitian serta sistematika penulisan skripsi.

2. BAB 2: TINJAUAN PUSTAKA

   Bab ini berisi penjelasan mengenai definisi maupun pemikiran-pemikiran yang dijadikan kerangka teoritis yang menyangkut dan mendasari pemecahan masalah dalam skripsi ini.

3. BAB 3: METODE PENELITIAN

   Bab ini berisi penjelasan mengenai studi pendahuluan, tahap pemilihan dataset, perancangan algoritma, pengembangan sistem, analisis hasil dan penarikan kesimpulan.

4. BAB 4: HASIL DAN PEMBAHASAN

   Bab ini berisi hasil penelitian berserta pembahasannya.

5. BAB 5: PENUTUP

   Bab ini berisi simpulan dari penulisan skripsi dan saran yang diberikan penulis untuk mengembangkan skripsi ini.

1.6.3. Bagian Akhir Skripsi

Bagian akhir skripsi ini berisi daftar pustaka yang merupakan informasi mengenai buku-buku, sumber-sumber dan referensi yang digunakan penulis serta lampiran-lampiran yang mendukung dalam penulisan skripsi ini.

BAB II

TINJAUAN PUSTAKA

Bab menjelaskan teori-teori yang menjadi referensi dalam penelitian. Penelitian ini menggunakan algoritma *sequence-to-sequence LSTM* berbasis *deep learning*.

2.1. *Automatic Text Summarization*

Text summarization adalah cabang penting dalam ilmu NLP yang bertujuan untuk merepresentasikan dokumen yang panjang dengan cara mengkompres informasi sehingga dapat dengan cepat dipahami dan dibaca oleh *user* (Joshi et al., 2019b). Tujuan utama dari *text summarization* adalah untuk menghasilkan ringkasan yang mana harus mengandung bagian-bagian penting dari dokumen dan dengan ukuran yang lebih pendek dari dokumen aslinya. Sebelumnya kita mungkin sudah sering melakukan tugas peringkasan teks secara manual, namun peringkasan teks secara manual membutuhkan waktu yang lama dan ketelitian yang tinggi untuk menghasilkan ringkasan yang bagus.

Seiring banyaknya informasi di era digital dan meningkatnya permintaan peringkasan teks maka dibuatlah peringkasan teks otomatis yang dilakukan oleh mesin. Peringkasan teks otomatis memiliki kelebihan yaitu mesin dapat memproses banyak informasi sekaligus secara cepat, sehingga dapat meminimalkan biaya dan sangat relevan dengan era sekarang yang serba cepat.

Saat ini *automatic text summarization* telah banyak digunakan untuk membantu

pekerjaan-pekerjaan manusia. Dalam dunia NLP, *automatic text summarization* juga banyak diterapkan dalam berbagai metode aplikasi, seperti *text classification, question answering*. Menurut Moreno & Marcus (2017) dalam kehidupan sehari-hari, kita dapat menemukan berbagai contoh ringkasan ada di sekitar kita. misalnya: abstrak artikel, ringkasan berita, review buku, trailer film, daftar isi. *Automatic text summarization* dapat bermanfaat di berbagai bidang karena fungsi utamanya yaitu dapat mengurangi panjang isi dari dokumen menjadi hanya bagian yang dibutuhkan saja.

Menurut Rahimi et al. (2018) dengan meningkatnya data di internet, website berita, dan beberapa media publikasi elektronik lainnya, salah satu tantangan utama peneliti pada abad ke-21 adalah bagaimana mendapatkan data yang akurat dan cepat. Hal ini menjadikan peringkasan teks otomatis menjadi sangat dibutuhkan mengingat banyaknya kebutuhan penghasil ringkasan dari informasi secara cepat dan tetap mempertahankan makna dari teks asli.

Berbagai metode telah dilakukan untuk menyelesaikan masalah *automatic text summarization*. Sebagian besar dari metode tersebut menggunakan teknik klasifikasi dengan cara mengklasifikasikan apakah kalimat ini layak masuk dalam ringkasan atau tidak. Berbagai pendekatan lain menggunakan metode *topic information, latent semantic analysis* (LSA), model *sequence-to-sequence, reinforcement learning*.

Secara umum, ada dua pendekatan utama dalam peringkasan otomatis, yaitu metode *extractive* dan *abstractive*.

1) *Extractive Text Summarization*

Peringkasan teks metode *extractive* adalah metode yang menghasilkan ringkasan dengan cara memberikan penilaian/*score* pada tiap kalimat berdasarkan kategori tertentu. Menurut Alami et al. (2019) metode *extractive* dikenal sebagai metode pemeringkatan kalimat yang terdiri dari kalimat yang di ekstrak berdasarkan tingkat kepentingan kalimat dalam teks. Proses ekstraksi yang dilakukan menggunakan metrik yang sudah ditentukan dan tanpa melakukan perubahan terhadap kata-kata inti yang ada dalam teks. Secara umum pendekatan ekstraktif lebih mudah karena metode ekstraktif hanya memilih beberapa bagian yang penting kemudian menyusunnya tanpa harus memperhatikan tata bahasa dan kecocokan kata.

2) *Abstractive Text Summarization*

Peringkasan teks metode *abstractive* adalah metode yang menghasilkan ringkasan dengan cara memahami konteks teks menggunakan NLP untuk memproduksi teks baru yang lebih pendek dari teks asli. Hasil yang dihasilkan oleh metode *abstractive* adalah hasil parafrase dari teks asli. Metode *abstractive* memungkinkan memiliki beberapa kata-kata yang sebelumnya tidak ada dalam teks asli. Metode ini adalah metode paling ideal dan memiliki potensi paling besar untuk menghasilkan ringkasan yang mirip dengan tulisan asli manusia (Reztaputra & Khodra, 2017). Secara umum pendekatan metode *abstractive* lebih susah karena

metode ini memerlukan kemampuan mesin untuk memahami topik dari teks. Selain itu hasil ringkasan juga harus tepat secara tata bahasa dan kecocokan kata, sehingga akan lebih mudah dibaca oleh manusia.

2.2. *Natural Language Processing*

Natural Language Processing (NLP) adalah suatu bidang studi interdisipliner yang mempelajari dan mengembangkan algoritma dan sistem yang memungkinkan komputer untuk memahami dan melakukan tugas-tugas yang melibatkan bahasa manusia (Sintoris & Vergidis, 2017). NLP juga dapat disebut sebagai komputasi linguistik, bahasa komputer, dan pemrosesan bahasa atau teknologi bahasa manusia. NLP berkaitan dengan bagaimana menganalisa bahasa manusia baik dalam bentuk tertulis maupun lisan.

NLP memiliki berbagai pengaplikasian seperti chatbot, *text summarization*, *machine translation,* dan aplikasi-aplikasi lain yang berhubungan dengan pengolahan bahasa manusia. NLP memungkinkan kita untuk dapat mengolah data dan informasi yang berkaitan dengan bahasa manusia.

Sebagian besar informasi yang ada di internet bersifat *unstructured,* yaitu berupa teks mentah baik dari sosial media, blog, berita. Hal tersebut menjadikan permasalahan NLP menjadi permasalahan yang cukup sulit untuk diselesaikan, karena NLP menuntut mesin untuk bisa memiliki kemampuan untuk membaca, memahami, dan mengambil informasi dari bahasa manusia yang bersifat *unstructured* dan abstrak bagi mesin.

Menurut Sorin et al. (2020) tujuan utama NLP adalah untuk mengkonversi bahasa manusia menjadi data yang terstruktur. Sebagai contoh ketika disediakan teks, mesin akan menggunakan algoritma untuk mengekstrak makna yang terkandung dalam setiap kalimat dan kemudian data dari kalimat tersebut dikumpulkan. Mungkin terkadang mesin akan gagal memahami makna dari suatu kalimat dengan jelas, karena sejatinya bahasa dari kalimat terkadang memiliki makna yang beragam dan dapat berubah sepanjang waktu.

Berikut ini adalah beberapa teknik dasar yang penting untuk diketahui dalam menyelesaikan permasalahan NLP:

1) *Sentence Segmentation*

   Menurut Indurkhya dan Damerau (2010), *sentence segmentation* adalah teknik dimana kita menentukan unit proses yang terdiri dari satu atau lebih kata. Teknik ini meliputi proses mengidentifikasi batas-batas kalimat antara kata-kata dalam kalimat yang berbeda.

2) *Tokenization*

   Menurut Sun et al. (2017), *tokenization* adalah teknik paling dasar dalam NLP. Metode ini memecah kalimat menjadi token yang berisi kata-kata atau frasa.

3) *Stemming* dan *Lemmatizations*

   Menurut Jivani (2012) fungsi dasar dari *stemming* dan *lemmatizations* adalah sama, kedua metode tersebut mengurangi varian kata dengan cara mencari akar katanya. Perbedaannya adalah dalam *stemming* mencari akar kata dengan cara menerapkan seperangkat aturan seperti menghilangkan imbuhan, Sedangkan

lemmatization mencari akar kata dengan cara memahami konteks kata tersebut dan mencari bentuk bakunya.

4) *Stopword Removal*

Stopword removal adalah proses menghapus kata-kata yang memiliki sedikit informasi untuk meningkatkan akurasi (Méndez et al., 2006). Kata-kata yang non-informatif biasa disebut *stopword*, Sebagai contoh dalam Bahasa Indonesia *stopword* biasa diidentikkan dengan kata penghubung semisal "dan", "atau", "yang", "ke". Dengan mengurangi frekuensi kata yang memiliki makna sedikit dapat menjadikan pemrosesan lebih cepat dan berkualitas karena hanya berfokus pada informasi yang penting.

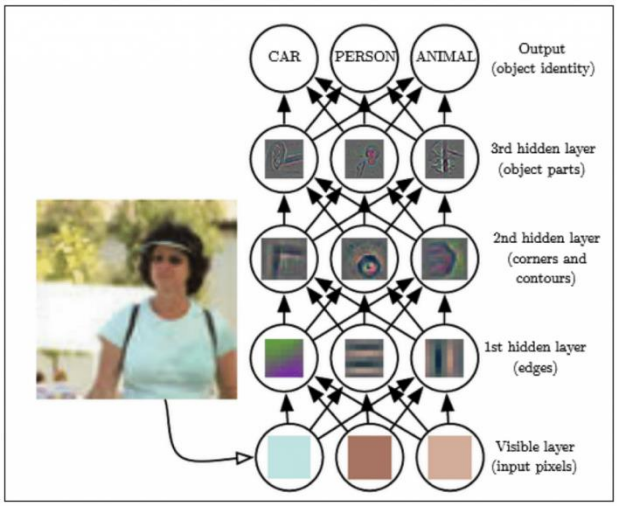
2.3. *Deep Learning*

Deep learning adalah bidang penelitian yang baru-baru ini muncul di *data science.* Menurut Sorin et al. (2020), *deep learning* pada dasarnya adalah bagian dari *machine learning* dimana komputer dapat belajar dari *attribute* tanpa harus melalui campur tangan manusia secara manual. Sebuah teknik yang memungkinkan komputer untuk meningkatkan kemampuan mereka melalui pengalaman dan data (Goodfellow et al., 2016). *Deep learning* terinspirasi oleh neuron otak dan telah terbukti mempunyai fleksibilitas yang lebih besar dan bisa membangun model yang lebih akurat dibandingkan *machine learning* biasa (Sharma, 2019). *Deep learning* juga sering disebut *deep neural network* yang terdiri dari banyak *layer* dengan jumlah *neuron* di

setiap *layer*. Jumlah *layer* bisa bervariasi dari hanya beberapa *layer* hingga ribuan *layer* dan setiap *layer* juga bisa berisi ribuan *neuron*.

Fondasi teoritis dari *deep learning* adalah berakar dari *neural network*. Perbedaan mendasar *deep learning* dengan *neural network* biasa adalah *deep learning* menggunakan lebih banyak *neuron* dan *hidden layer* (Ravi et al., 2017). Banyak orang percaya bahwa algoritma *deep learning* lebih baik daripada *neural network* biasa, karena dalam *deep learning* ada lebih banyak transformasi layer. *Deep learning* juga memiliki kemampuan untuk merubah data dari non-linear menjadi linear.

Dalam *neural network* setiap *hidden layer* bertanggung jawab untuk melatih serangkaian fitur unik berdasarkan output dari jaringan sebelumnya (Goodfellow et al., 2016). Algoritma ini akan menjadi semakin kompleks dan bersifat abstrak ketika jumlah *hidden layer* semakin bertambah banyak. Jaringan saraf yang dimiliki oleh *deep learning* terbentuk dari hierarki sederhana dengan beberapa lapisan hingga tingkat tinggi atau banyak lapisan (*multi layer*). *Deep learning* memiliki kekuatan dan fleksibilitas yang tinggi dengan cara merepresentasikan data sebagai konsep hierarki yang bersarang, dengan setiap konsep didefinisikan memiliki relasi dengan konsep yang lebih sederhana (Goodfellow et al., 2016). Gambar 2.1 adalah ilustrasi bagaimana *deep learning* merepresentasikan sebuah masalah menjadi suatu konsep graf.



Gambar 2.1. Representasi *deep learning* menjadi graf

(Goodfellow et al., 2016)

2.4. *Sequence-to-Sequence*

Model *sequence-to-sequence* pertama kali diperkenalkan oleh Google pada tahun 2014 (Kostadinov, 2019). Model *sequence-to-sequence* bertujuan untuk memetakan suatu panjang input ke suatu panjang input yang tetap. Namun, dengan output yang berbeda. Dalam berbagai penerapan model *deep learning* berbasis *sequence-to-sequence* telah berhasil diterapkan dalam berbagai permasalahan seperti *machine translation, speech recognition,* dan *video captioning* (Nallapati et al., 2016). Model *sequence-to-sequence* dapat menjadi solusi untuk masalah yang berbasis urutan (*sequence*), terutama *input* dan *output* memiliki ukuran dan kategori yang berbeda.

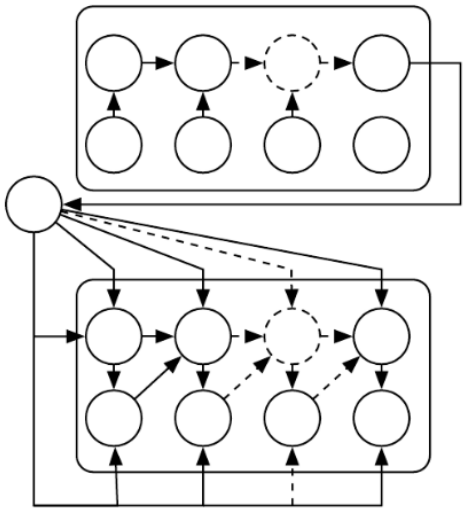
Contoh sederhana arsitektur *sequence-to-sequence* adalah penerapan *machine*

translation atau mesin penerjemah. Dalam menerjemahkan bahasa mesin akan menerima input berupa suatu *sequence* atau urutan kata kemudian mesin akan mempelajari pola dan akan menghasilkan output lagi berupa *sequence* atau urutan kata dengan panjang yang berbeda yang berbeda. Sebagai contoh dalam penerjemahan kata dari bahasa Inggris ke bahasa Korea. Ketika mesin menerima input kata berupa "I Love You" mesin akan mempelajari input tersebut sebagai *sequence* atau urutan tiga suku kata dalam bahasa Inggris, kemudian mesin akan mempelajari pola dan menghasilkan output berupa "Salanghae" dalam bahasa Korea yang dalam hal ini adalah *sequence* yang terdiri dari satu suku kata. Tentu saja dalam *machine translation* kata dari bahasa Inggris tersebut dapat diterjemahkan dalam berbagai bahasa dengan output jumlah urutan kata yang berbeda-beda tergantung bahasanya. Permasalahan tadi adalah salah satu penerapan metode *sequence-to-sequence* dalam kehidupan sehari-hari yaitu pada *machine translation* atau yang sering kita gunakan adalah Google Translate.

Model *sequence-to-sequence* memiliki dua bagian penting yaitu *encoder* dan *decoder* (Goodfellow et al., 2016). *Encoder recurrent neural network* (RNN) membaca input berupa *sequence* kemudian *decoder* output membangkitkan output berupa *sequence.* Dalam *question answering* input sequence adalah kumpulan semua kata dalam pertanyaan tersebut. Kemudian *decoder* membangkitkan *sequence* berupa jawaban berdasarkan pola yang sudah dibuat.

Dalam penelitian ini akan digunakan metode *sequence-to-sequence* dalam *abstractive text summarization* yang mana mesin akan menerima *input* berupa urutan

teks berita kemudian mempelajari polanya dan akhirnya akan menghasilkan *output* berupa urutan ringkasan. Namun menurut (Goodfellow et al., 2016), metode *sequence-to-sequence* memiliki suatu kelemahan yaitu susah untuk menghasilkan *output* yang akurat ketika input memiliki *sequence* yang panjang dan harus menghasilkan *output* dengan dimensi yang sangat kecil. Hal tersebut seringkali ditemui dalam peringkasan teks yang mana memiliki *input* berupa teks yang sangat panjang dan harus menghasilkan ringkasan yang singkat. Untuk mengatasi masalah tersebut (Bahdanau et al., 2015), mengusulkan sebuah *attention mechanism* yang mana mengaitkan elemen-elemen dari *input sequence* menjadi output *sequence*. Dalam penelitian ini, selain menggunakan metode *sequence-to-sequence* juga digunakan *attention mechanism* untuk mengatasi masalah tersebut. Gambar 2.2 adalah ilustrasi mekanisme *sequence-to-sequence*.



Gambar 2.2. Mekanisme sequence-to-sequence

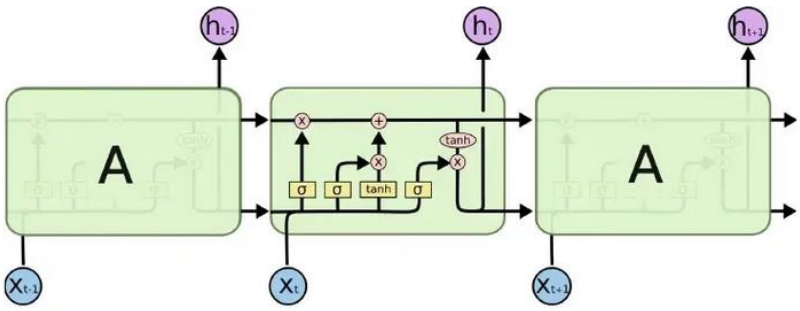
(Goodfellow et al., 2016)

2.5. *Long Short-Term Memory (LSTM)*

Model RNN biasa memiliki masalah yaitu susah untuk mempelajari input berupa *sequence* yang sangat panjang karena RNN memiliki memori yang sangat terbatas. RNN sederhana atau varian lainnya mengalami masalah *vanishing gradient* seiring bertambahnya panjang urutan data yang ingin dilatih kepada RNN tersebut. *Vanishing gradient* adalah situasi dimana nilai *gradient* yang digunakan untuk mengupdate *weight* pada neuron pada sesi *training artificial neural network* (ANN) bernilai 0 atau mendekati 0. Untuk mengatasi masalah tersebut Hochreiter dan Schmidhuber pada tahun 1997 memperkenalkan LSTM. LSTM dapat mengatasi permasalahan terkait dengan mengingat urutan dalam jangka waktu yang lama.

Sekarang LSTM sudah banyak sekali digunakan untuk banyak *task* yang berkaitan dengan *sequence* termasuk banyak permasalahan NLP oleh Google.

LSTM menyimpan informasi berupa pola-pola pada data. LSTM dapat mempelajari data mana saja yang akan disimpan dan data mana saja yang akan dibuang, karena pada setiap neuron LSTM memiliki beberapa *gates* yang mengatur memori pada setiap neuron itu sendiri. LSTM didasarkan pada ide bagaimana membuat suatu jalur (*path*) berdasarkan waktu yang memiliki turunan yang tidak akan mengalami *vanising gradient* atau *exploding gradient* (Goodfellow et al., 2016). LSTM memiliki isi *cell* yang kompleks yang menjadikan LSTM dapat mempelajari pola data yang panjang dan berurut, karena situasi seperti *vanising gradient* dapat dicegah. Perbedaan antara LSTM dan RNN biasa adalah isi cellnya. Ilustrasi isi cell LSTM dapat dilihat pada Gambar 2.3.



Gambar 2.3. Cell LSTM

2.6. *Recall-Oriented Understudy for Gisting Evaluation (ROUGE)*

ROUGE adalah serangkaian *metric* atau *library* yang digunakan untuk evaluasi *automatic text summarization* dan *machine translation* dalam *NLP* (Lin, 2004). Cara kerjanya adalah dengan cara membandingkan ringkasan atau terjemahan yang dihasilkan secara otomatis dengan ringkasan atau terjemahan asli yang dibuat oleh manusia. Sebagai contoh jika ada ringkasan asli buatan manusia "Seekor kucing ada di bawah kasur" dan ringkasan yang dibuat otomatis oleh mesin "Seekor kucing ditemukan di bawah kasur", ROUGE akan membandingkan ringkasan yang dibuat otomatis oleh mesin dan ringkasan yang dibuat manual oleh manusia.

Sederhananya, jika ROUGE adalah proses membandingkan ringkasan yang dihasilkan otomatis oleh mesin dan ringkasan yang dibuat manual oleh manusia maka dapat direpresentasikan sebagai berikut.

$$ROUGE = \frac{jumlahkatarelevanyangdihasilkansecaraotomatis}{totalkatayangadadalamreferensiringkasan}$$

Algoritma menghitung perbedaan unit seperti *n-gram, word sequence,* pasangan kata dari ringkasan yang di *generate* komputer dengan dengan ringkasan yang dibuat manual oleh manusia (Lin, 2004).

Suatu ringkasan dikatakan bagus selain karena memuat informasi utama yang ada dalam teks ringkasan tersebut juga harus mudah dibaca dan dipahami. Maka dalam sistem ROUGE tidak hanya membandingkan jumlah kata yang dihasilkan secara otomatis namun juga membandingkan kesesuaian katanya dengan kata-kata yang lain

agar mudah dibaca. Maka dari itu ada beberapa *metric* yang ada di dalam ROUGE yang bisa menjadi standar bagus tidaknya suatu ringkasan, standar *metric* tersebut antara lain: ROUGE-N, ROUGE-S & ROUGE-L. Berikut adalah penjelasan masing-masing *metric*:

- ROUGE-N: membandingkan antara N-grams buatan sistem dan ringkasan asli.
    - ROUGE-1 merujuk pada perbandingan *unigram* (setiap kata) antara ringkasan sistem dan ringkasan manual.
    - ROUGE-2 merujuk pada perbandingan *bigrams* (kombinasi dua kata) antara ringkasan sistem dan ringkasan manual.
- ROUGE-L: Perbandingan berdasarkan *Longest Common subsequence* (LCS). Adalah perbandingan berdasarkan urutan terpanjang memperhitungkan kesamaan struktur level kalimat secara alami dan mengidentifikasi kombinasi terpanjang dalam urutan n-gram secara otomatis.
- ROUGE-S: Statistik kombinasi Skip-bigram. Skip-bigram adalah pasangan kata dalam urutan kalimat.

2.7. *Attention Mechanism*

Attention mechanism pada dasarnya adalah metode yang digunakan untuk membantu model *neural network* dengan cara memberikan bobot yang berbeda pada input vektor sehingga dapat mengoptimalkan proses pembelajaran pada beberapa target. Dengan memberikan bobot kita bisa memfokuskan proses pembelajaran pada bagian paling relevan saja daripada semua bagian, hal ini dapat mengoptimalkan proses pembelajaran pada mesin. Sebagai contoh dalam dunia nyata ketika kita ingin membedakan jenis hewan akan lebih mudah bagi kita untuk membedakan berdasarkan bagian-bagian tubuh hewan tersebut daripada langsung membedakan semua bagian.

Menurut Ran et al (2019), *Attention Mechanism* telah berhasil diterapkan di berbagai metode, seperti *image classification, neural machine translation, multimedia recomendation, text summarition,* dll. Hal tersebut dikarenakan metode *attention* dapat memfokuskan proses pada bagian yang efektif dan dilakukan secara adaptif. Sebagai contoh penerapan dalam *image classification* akan sangat berat sekali apabila menerapkan CNN pada gambar yang besar, gambar yang besar dengan resolusi pixel yang besar dapat memperberat proses komputasi gambar. Dengan *attention mechanism* mesin dapat melakukan proses hanya berfokus pada bagian yang efektif saja sehingga dapat mengurangi beratnya proses komputasi.

Dalam penelitian ini *attention mechanism* digunakan dalam proses *text summarization,* yaitu untuk memfokuskan proses meringkas pada bagian-bagian penting saja dalam teks asal.

2.8. Penelitian Terkait

Penelitian ini dikembangkan dari beberapa referensi penelitian terdahulu yang mempunyai keterkaitan dengan metode dan objek penelitian. Penggunaan referensi ini ditujukan untuk memberikan batasan-batasan terhadap metode yang nantinya akan dikembangkan lebih lanjut. Berikut adalah beberapa penelitian terkait dengan penelitian yang diusulkan:

1. Cai et al. (2019) melakukan penelitian tentang *text summarization* berbahasa Indonesia dengan menggunakan pendekatan *extractive text summarization* yang mana menghasilkan ringkasan dengan cara memilih kalimat mana yang layak untuk dijadikan ringkasan. Metode yang digunakan adalah *sentence similarity clustering,* yang mana mengelompokkan kalimat-kalimat berdasarkan kesamaanya dengan kalimat lain. Dataset yang digunakan dalam penelitian ini yaitu IndoSum (Kurniawan & Louvan, 2018). Metode evaluasi yang digunakan adalah ROUGE (Lin, 2004) dan hasil dari evaluasi metode kali ini adalah adalah ROUGE-1 0.290 ROUGE-2 0.071 ROUGE-3 0.021. Berdasarkan hasil evaluasi tersebut diketahui bahwa model yang dihasilkan masih belum memiliki hasil yang cukup baik dibuktikan dengan hasil rata-rata ROUGE yang masih di bawah angka 0.5.
2. Garmastewira (2020). melakukan penelitian tentang peringkasan teks untuk berita berbahasa Indonesia. Garmastewira menggunakan *deep*

learning dalam metodenya. Garmastewira menghasilkan ringkasan berbasis *extractive* bukan *abstractive*. Penelitian ini menggunakan *Graph Convolutional Network* (GCN) yang memungkinkan untuk *word embedding sequence* dan *sentence relationship graph* sebagai *input* untuk ringkasan artikel berita berbahasa Indonesia. Sistem dalam penelitian ini terdiri dari empat komponen utama: preproses, konstruksi graf, *sentence scoring*, dan komponen *sentence selection*. Dalam kompenen *sentence scoring* menggunakan *neural network* dan g*raph convolutional network* untuk menghasilkan skor semua kalimat dan akhirnya skor tertinggi akan dipilih. Hasil terbaik yang dihasilkan adalah dengan rata-rata ROUGE-2 0.370.

3. Adelia et al. (2019) melakukan penelitian tentang *abstractive text summarization* berbahasa Indonesia dengan menggunakan metode *bidirectional gated recurrent unit* (GRU). Metode *bidirectional* GRU merupakan metode *sequence to sequence*. Mereka menggunakan GRU sebagai *cell* dalam RNN. Dataset yang mereka gunakan adalah 1000 jurnal berbahasa Indonesia dengan abstrak, namun tidak tersedia secara publik. Metode evaluasi yang mereka digunakan adalah ROUGE (Lin, 2004) dengan rata-rata skor ROUGE-1 (*unigram*) 0.11975 dan ROUGE-2 (*bigram*) 0.01199. Dari hasil tersebut diketahui bahwa hasil dari ROUGE-1 lebih besar daripada ROUGE-2. Hal ini membuktikan bahwa model bisa menghasilkan kata yang benar untuk ringkasan secara individual, tapi tidak

cukup bagus untuk kata yang berpasangan sehingga masih belum bisa menghasilkan ringkasan dengan kalimat yang mudah dibaca. Selain itu, dari hasil tersebut juga masih kalah jauh dengan *text summarization* berbahasa Inggris karena ukuran dari dataset yang kecil dan faktor linguistik.

4. Ivanedra & Mustikasari (2019) melaukan penelitian tentang *abstractive text summarization* berbahasa Inggris. Dalam penelitian mereka menggunakan metode *RNN* dan LSTM, namun dalam penelitian ini belum dijelaskan secara detail bagaimana struktur layer dalam *RNN* yang digunakan. Dataset yang digunakan adalah 4515 artikel berita online, namun dataset ini tidak tersedia secara publik. Berdasarkan hasil evaluasi nilai rata-rata *recall* artikel berita dengan teknik *stemming* sebesar 44%. Berdasarkan contoh hasil ringkasan yang mereka hasilkan, *input text* memiliki panjang rata-rata satu paragraf dan menghasilkan output rata-rata satu kalimat. Sehingga masih belum bisa dipastikan apakah akan berhasil untuk *input* berupa berita yang memiliki panjang rata-rata lebih dari satu paragraf.
5. Nallapati et al. (2016) melakukan penelitian tentang *abstractive text summarization* berbasis *deep learning* berbahasa Inggris. Metode yang digunakan adalah *attentional sequence-to-sequence.* Untuk *encoder* adalah *bidirectional* GRU RNN. Untuk *decoder* adalah *uni-directional* GRU RNN dengan jumlah *hidden state* yang sama. *Dataset* yang digunakan tersedia

secara publik. Mereka menggunakan tiga dataset untuk mengetahui performa model mereka dalam *dataset* yang berbeda-beda yaitu: Gigaword corpus, DUC corpus, CNN/Daily Mail corpus. Untuk metode evaluasi mereka menggunakan ROUGE sebagai *metric* evaluasi mereka. Dari model ini mereka mendapatkan hasil yang memuaskan.

6. Kurniawan & Louvan (2018) selain memperkenalkan *dataset* baru yang bisa dijadikan sebagai *benchmark* dalam penelitian *text summarization* berbahasa Indonesia, mereka juga mencoba menerapkan *benchmark* mereka untuk metode *extractive text summarization* berbasis *neural network* bernama NeuralSum dengan implementasi yang masih original dari penelitinya[1]. Untuk metode evaluasi mereka menggunakan ROUGE sebagai metric evaluasi mereka. *Source code* untuk evaluasi metode mereka tersedia secara online[2].

Ilustrasi perbandingan metode peringkasan teks dengan berbagai macam metode dan hasil evaluasinya diilustrasikan dalam Tabel 2.1.

Table 2.1. Hasil evaluasi ROUGE score penelitian terkait

Metode	Peneliti	Bahasa	Dataset	R-1	R-2
Extractive	Cai et al., (2019)	Indonesia	Publik (IndoSum)	0.29	0.071

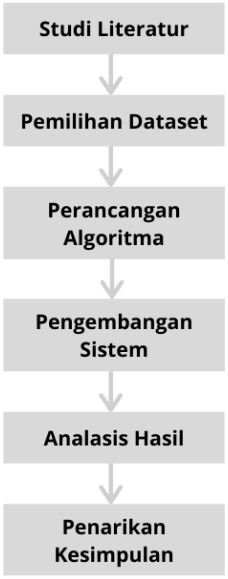
1 https://github.com/cheng6076/NeuralSum
2 https://github.com/kata-ai/indosum

	Garmastewira., (2020)	Indonesia	Tidak disebutkan	-	0.370
	Kurniawan & Louvan., (2018)	Indonesia	Publik (IndoSum)	0.46	0.48
Abstractive	Adelia et al., (2019)	Indonesia	Tidak Publik	0.1197	0.0119
	Ivanedra & Mustikasari., (2019)	Inggris	Tidak Publik	0.44	-
	Nallapati et al., (2016)	Inggris	Publik (CNN/Dailymail Corpus)	0.3546	0.1664

BAB III

METODE PENELITIAN

Bab ini menjelaskan metode dan cara kerja yang digunakan dalam penelitian ini. Pembahasan pada bab ini diharapkan dapat memberikan gambaran bagaimana membangun sistem peringkasan teks menggunakan metode *sequence-to-sequence*. Ada beberapa tahap yang dilakukan secara sistematis dalam melakukan penelitian ini, dimulai dari tahap pra-penelitian yang terdiri dari studi literatur dan penyusunan *dataset*, tahap penelitian yang terdiri perancangan algoritma dan pengembangan sistem, serta pasca penelitian yang terdiri dari analisis hasil dan penarikan kesimpulan. Tahapan penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1. *Flowchart* tahap metode penelitian

3.1. Studi Literatur

Studi literatur dilakukan untuk mendapatkan gambaran secara lengkap mengenai permasalahan yang ada dalam penerapan *automatic text summarization* berbahasa Indonesia berbasis *deep learning* terutama data berita berbahasa Indonesia. Pada tahap ini, dilakukan penyusunan dataset, informasi, serta rujukan dari buku, artikel, jurnal dan karya ilmiah lainnya yang berhubungan dengan *automatic text summarization*. Bahan referensi yang digunakan mencakup penemuan atau modifikasi terhadap penerapan *deep learning* untuk *text summarization* khususnya menggunakan metode RNN *sequence-to-sequence* LSTM.

Misalnya peneliti melakukan penelitian tentang *abstractive text summarization* berbahasa Indonesia seperti yang dilakukan (Adelia et al., 2019). Mengambil penerapan *sequence-to-sequence* untuk *text summarization* seperti yang dilakukan (Nallapati et al., 2016). Kemudian menggunakan *dataset* publik yang disediakan oleh (Kurniawan dan Louvan, 2019). Sebagai pembanding hasil yang dilakukan peneliti dan peneliti sebelumnya, pengujian penelitian ini menggunakan metode ROUGE. Selain itu, studi literatur lain juga dibutuhkan untuk implementasi dalam melakukan uji coba menggunakan Bahasa pemrograman *Python* dan library *tensorflow*.

3.2. Pemilihan Dataset

Pada tahap ini, peneliti mencari data sebagai objek dalam eksperimen penelitian. Setelah melakukan *research* terhadap beberapa *dataset text summarization* berbahasa Indonesia yang tersedia secara publik, akhirnya peneliti memutuskan untuk menggunakan *dataset* IndoSum[3] (Kurniawan & Louvan, 2018). Peneliti memilih menggunakan *dataset* tersebut karena *dataset* tersebut memiliki ukuran yang cukup besar sehingga memadai untuk digunakan sebagai objek penelitian berbasis *neural network*. Selain itu, *dataset* tersebut juga tersedia secara publik dan bisa diunduh oleh siapapun, sehingga dapat dijadikan sebagai *benchmarking* untuk penelitian *text summarization* berbahasa Indonesia.

Dataset IndoSum diperoleh dari Shortir, sebuah perusahaan *news agregator* berbahasa Indonesia. Ringkasan yang dihasilkan dibuat manual oleh dua penutur asli Indonesia. Total isi *dataset* lebih dari 20 ribu artikel berita, dimana setiap data memiliki *field* berupa judul, kategori, sumber berita (Misal: CNN Indonesia, Kumparan), url ke berita asli. *Dataset* terdiri dari 6 kategori *Entertainment, Inspiration, Sport, Showbiz, Headline,* dan *Tech* sehingga cukup memiliki sebaran topik yang kaya untuk model yang akan dihasilkan.

3 https://github.com/kata-ai/indosum

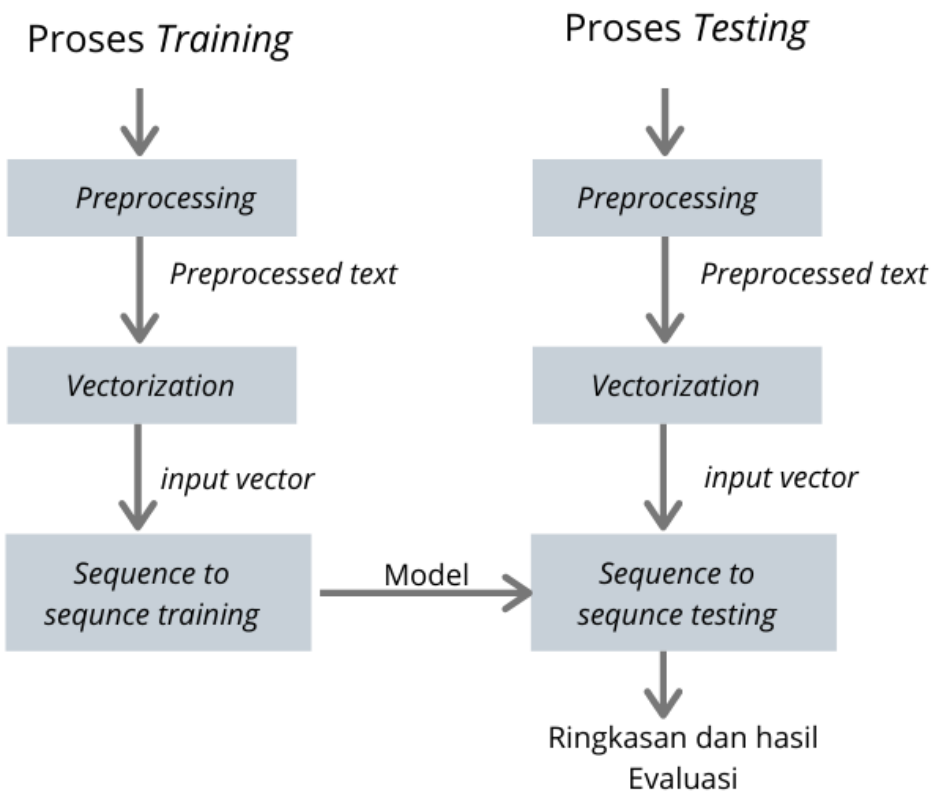
Dalam penelitian ini, data yang digunakan terbagi menjadi tiga jenis yaitu:

1) Data *training:* dalam proses *training, dataset* digunakan sebagai *input* yang menghasilkan model dari algoritma *deep learning* yang akan peneliti gunakan.
2) Data *testing*: dalam proses *testing, dataset* digunakan sebagai *input* untuk proses evaluasi model yang menghasilkan nilai evaluasi dari metric yang menggambarkan apakah model yang dihasilkan bagus atau tidak.
3) Data *development*: untuk proses *development, dataset* digunakan sebagai *input* yang menghasilkan bagaimana *output* sebenarnya yang dihasilkan oleh model.

Dalam penelitian ini untuk data *training, testing*, dan *development* berasal dari sumber yang sama namun dipecah menjadi tiga bagian.

3.3. Perancangan Algoritma

Setelah mendapatkan *dataset* sebagai objek penelitian ini, langkah selanjutnya adalah perancangan algoritma. Perancangan algoritma adalah tahap dimana peneliti menyusun alur dan metode yang digunakan dalam studi kasus peneliti kali ini, dari tahap *preprocessing dataset*, *training, testing hingga* evaluasi. Berikut peneliti ilustrasikan alur metode *text summarization* yang akan peneliti buat dalam Gambar 3.2.



Gambar 3.2. *Flowchart* alur metode peneliti

Berikut adalah penjelasan dari ilustrasi Gambar 3.2:

3.3.1. Proses *Training*

Proses *training* adalah proses pembentukan pola dari model *deep learning.* Tujuan dari proses *training* adalah untuk membuat model yang akurat berdasarkan fungsi tertentu, karena sejatinya metode *deep learning* adalah metode mempelajari pola abstrak yang dihasilkan oleh model, maka untuk menghasilkan model yang bagus kita membutuhkan banyak data sehingga

semakin banyak pola yang tersedia. Berikut adalah beberapa tahap yang ada dalam proses *training*:

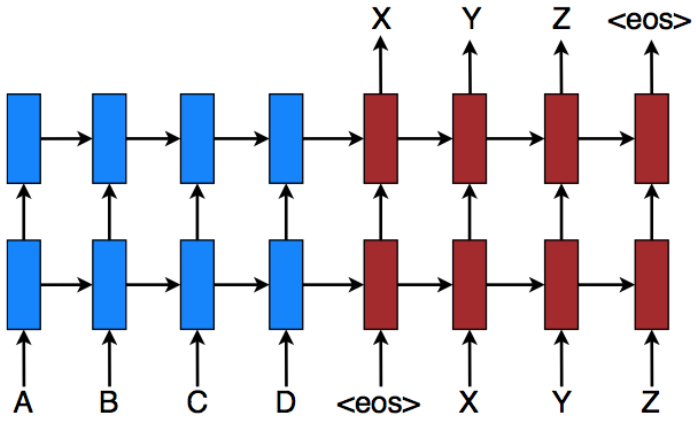
1) ***Preprocessing.*** Ini merupakan proses mengubah teks yang sebelumnya tidak terstruktur menjadi terstruktur (Adelia et al., 2019). Proses *preprocessing* pada penelitian ini melalui beberapa tahapan yakni *case folding*, dan *removing punctuation. Case folding* adalah proses mengkonversi semua karakter menjadi standar yang sama apakah itu *uppercase* atau *lowercase.* Dalam hal ini, peneliti menggunakan *lowercase* agar mudah dibaca. *Removing punctuation* adalah proses menghilangkan tanda baca, karena sebenarnya dalam ringkasan tanda baca tidak terlalu dibutuhkan.

   Setelah data dibersihkan dari kata-kata yang tidak diperlukan, tahap selanjutnya adalah tokenisasi. Proses tokenisasi adalah proses mengubah tiap artikel dari dataset menjadi bagian kecil yaitu urutan angka (Adelia et al., 2019). Caranya adalah membuat *word vocabulary* yang berisi semua kata unik yang ada dalam artikel kemudian memberikan label berupa angka untuk tiap kata/token dan akhirnya mengkonversi tiap kata dalam artikel berdasarkan *vocabulary* yang sudah dibuat. Untuk token yang tidak ada dalam *vocabulary* peneliti merepresentasikannya sebagai “<UNK>” atau *unknown.* ini bertujuan untuk mengatasi token yang tidak ada dalam *training set* (*vocabulary*). Peneliti tidak menerapkan metode *preprocessing*

yang mengubah bentuk kata seperti *lemmatization* atau *stemming.* Dalam penelitian ini peneliti menggunakan NLTK untuk proses tokenisasi.

2) ***Vectorization.*** Proses vektorisasi adalah proses mempresentasikan setiap kata dalam konteks vektor dengan N dimensi (Ivanedra & Mustikasari, 2019). Metode vektorisasi yang digunakan adalah metode *word embedding.* Metode *word embedding* adalah metode yang memetakan kata menjadi suatu *vector* yang dalam hal ini proses vektorisasi menggunakan word2vec. Dalam penelitian ini digunakan ribuan artikel dari Wikipedia bahasa Indonesia. Untuk dimensi vektor yang peneliti gunakan kali ini adalah 50 dimensi untuk percobaan. Karena dengan *vector* 50 dimensi dinilai cukup cepat untuk percobaan *training* data. Untuk penelitian selanjutnya, peneliti akan meningkatkan dimensi vektor guna mendapatkan hasil yang lebih optimal.

3) ***Sequence-to-Sequence Training,*** yaitu proses melakukan *training* terhadap model yang sudah dibuat. Metode yang digunakan adalah metode berbasis *sequence* bernama RNN. RNN pada umumnya menghasilkan *output* dengan panjang yang sama dengan *input*. Namun karena hasil ringkasan harus memiliki panjang yang berbeda dari *input* maka digunakan mekanisme *encoding decoding* dengan *attention mechanism.* Model penelitian ini diilustrasikan dalam Gambar 3.3.



Gambar 3.3. Model *sequence-to-sequence*

Penjelasan dari struktur model yang ditunjukkan oleh Gambar 3.3 adalah sebagai berikut.

a. ***Encoder***. Token dari dokumen adalah input untuk proses *training*. Melalui proses *word embedding* token akan diubah menjadi *vector*. Untuk setiap *sequence*, *bidirectional* RNN diterapkan untuk menghasilkan *hidden state* di setiap *sequence*.

b. ***Decoder.*** *Decoder* adalah bagian yang menghasilkan ringkasan berdasarkan *input* dari hasil *encoder*. *Decoder* terdiri dari *uni directional* RNN karena dalam proses *decode*, *decoder* hanya menerjemahkan *input* dari *encoder*.

3.3.2. Proses *Testing*

Model yang sudah dilatih sebelumnya disimpan dalam suatu direktori sehingga bisa digunakan lagi untuk prediksi maupun untuk *testing*. Proses *testing* adalah proses untuk mengetahui seberapa kualitas model yang dihasilkan oleh proses *training.* Untuk *metric* evaluasi menggunakan ROUGE (Lin, 2004) yang mana merupakan standar evaluasi untuk penelitian *text summarization*. Peneliti menggunakan *library Pythonrouge* untuk implementasi evaluasi ini.

3.4. Pengembangan Sistem

Tahap pengembangan sistem adalah tahap mentransformasi algoritma menjadi kode program. Sebelum melakukan pengembangan sistem langkah pertama yang dilakukan peneliti adalah melakukan analisis kebutuhan. Tahapan ini merupakan tahap analisis terhadap kebutuhan-kebutuhan yang diperlukan dalam penelitian untuk memperlancar proses analisis algoritma. Tahapan ini mencakup analisis yang sedang berjalan, analisis masalah, sistem usulan dan analisis kebutuhan fungsional. Untuk kebutuhan sistem yang akan digunakan menggunakan bahasa pemrograman *Python*, *library tensorflow* dan *dataset* dari IndoSum.

Tahap selanjutnya adalah penulisan kode program. Output dari kode program dalam penelitian ini adalah sebuah aplikasi berbasis *website*. Aplikasi dibagi menjadi

dua bagian yaitu server dan *client*. Server berisi program python yang berfungsi menyediakan layanan *text summarization.* Sementara *client* berisi *user interface* aplikasi yang berfungsi mengirimkan *request* dari *client* dan menerima *response* dari server untuk ditampilkan pada *client.*

3.5. Analisis Hasil

Analisis hasil merupakan tahapan yang paling penting dalam penyelesaian suatu kegiatan penelitian ilmiah. Dalam penelitian ini, analisis hasil dilakukan menggunakan analisis ROUGE.

ROUGE *analysis* dalah *metric* evaluasi untuk penelitian *automatic text summarization,* ROUGE menghitung perbandingan kesesuaian kata antara ringkasan yang dihasilkan oleh mesin dan ringkasan asli yang dibuat oleh manusia. Hasil dari ROUGE digunakan peneliti untuk membandingkan apakah hasil penelitian yang dilakukan oleh peneliti kurang baik atau lebih baik daripada peneliti-peneliti sebelumnya dengan penelitian sejenis.

3.6. Penarikan Kesimpulan

Kesimpulan dari penelitian ini diperoleh dari hasil algoritma terbaik dalam penyusunan algoritma *abstractive text summarization* berbahasa Indonesia dengan metode *deep learning* berdasarkan metode-metode pengujian yang ada pada algoritma yang diusulkan. Penarikan kesimpulan didasarkan pada studi pendahuluan, analisa *dataset* dan pengembangan sistem serta analisa dari hasil penelitian.

BAB IV

HASIL DAN PEMBAHASAN

4.1. Hasil Penelitian

Pada penelitian ini, algoritma *text summarization* yang digunakan adalah *sequence-to-sequence*, subjek yang digunakan dalam penelitian ini adalah *dataset* artikel berita. Penelitian ini mempelajari pola dari *dataset* yang sudah disediakan melalui proses *training*. Tahapan-tahapan yang harus dilakukan yaitu tahap persiapan *dataset*, *preprocessing* dataset, tokenisasi *dataset*, vektorisasi *dataset*, training model, prediksi ringkasan hingga implementasi sistem. Gambaran tahapan-tahapan penelitian dapat dilihat pada Gambar 4.1

Gambar 4.1. *Flowchart* tahap penelitian

Berikut adalah penjelasan dari *flowchart* pada Gambar 4.1.

4.1.1. Persiapan Dataset

Pada penelitian ini digunakan *dataset* berupa artikel berita untuk menjadi subjek terhadap pengenalan pola. *dataset* yang digunakan adalah IndoSum, yang berisi 20 ribu artikel berita berbahasa Indonesia. *Dataset* berisi isi berita, ringkasan, kategori, sumber, url sumber, sebagaimana ditunjukkan pada Tabel 4.1.

Table 4.1. Struktur *dataset*

Kategori	**Tajuk Utama**
Artikel **Berita**	Jakarta , CNN Indonesia Dokter Ryan Thamrin , yang terkenal lewat acara Dokter Oz Indonesia , meninggal dunia pada Jumat 4 8 dini hari . Dokter Lula Kamal yang merupakan selebriti sekaligus rekan kerja Ryan menyebut kawannya itu sudah sakit sejak setahun yang lalu . Lula menuturkan , sakit itu membuat Ryan mesti vakum dari semua kegiatannya , termasuk menjadi pembawa acara Dokter Oz Indonesia . Kondisi itu membuat Ryan harus kembali ke kampung halamannya di Pekanbaru , Riau untuk menjalani istirahat . Setahu saya dia orangnya sehat , tapi tahun lalu saya dengar dia sakit . Karena sakitnya , ia langsung pulang ke Pekanbaru , jadi kami yang mau jenguk juga susah . Barangkali mau istirahat , ya betul juga , kalau di Jakarta susah isirahatnya , kata Lula kepada CNNIndonesia.com , Jumat 4 8 . Lula yang mengenal Ryan sejak sebelum aktif berkarier di televisi mengaku belum sempat membesuk Ryan lantaran lokasi yang jauh . Dia juga tak tahu penyakit apa yang diderita Ryan . Itu saya enggak tahu , belum sempat jenguk dan enggak selamanya bisa dijenguk juga . Enggak tahu berat sekali apa bagaimana , tutur Ryan . Walau sudah setahun menderita sakit , Lula tak mengetahui apa penyebab pasti kematian Dr Oz Indonesia itu . Meski demikian , ia mendengar beberapa kabar yang menyebut bahwa penyebab Ryan meninggal adalah karena jatuh di kamar mandi . “ Saya tidak tahu , barangkali penyakit yang dulu sama yang sekarang berbeda , atau penyebab kematiannya beda dari penyakit sebelumnya . Kita kan enggak bisa mengambil kesimpulan , kata Lula . Ryan Thamrin terkenal sebagai dokter yang rutin membagikan tips dan informasi kesehatan lewat tayangan Dokter Oz Indonesia . Ryan menempuh Pendidikan Dokter pada tahun 2002 di Fakultas Kedokteran Universitas Gadjah Mada . Dia kemudian melanjutkan pendidikan Klinis Kesehatan

	Reproduksi dan Penyakit Menular Seksual di Mahachulalongkornrajavidyalaya University , Bangkok , Thailand pada 2004 .
Ringkasan	Dokter Lula Kamal yang merupakan selebriti sekaligus rekan kerja Ryan Thamrin menyebut kawannya itu sudah sakit sejak setahun yang lalu . Lula menuturkan , sakit itu membuat Ryan mesti vakum dari semua kegiatannya , termasuk menjadi pembawa acara Dokter Oz Indonesia . Kondisi itu membuat Ryan harus kembali ke kampung halamannya di Pekanbaru , Riau untuk menjalani istirahat .
Sumber	cnn indonesia
URL	https://www.cnnindonesia.com/hiburan/20170804120703-234-232443/lula-kamal-dokter-ryan-thamrin-sakit-sejak-setahun-lalu/

Dataset dibentuk menggunakan format .jsonl, yaitu sebuah file teks yang tiap barisnya berisi data json berita tersebut. Selain memiliki ringkasan dalam bentuk parafrase, *dataset* di atas juga bisa menghasilkan ringkasan yang bisa disusun berdasarkan kalimat yang ditandai dengan *gold labels*. *Gold labels* yang dimaksud adalah label bernilai 0 dan 1, yang artinya jika kalimat memiliki label 1 maka kalimat tersebut digunakan dalam ringkasan dan apabila memiliki label 0 maka kalimat tersebut tidak digunakan dalam ringkasan. Fitur ini biasa digunakan untuk mengolah data ringkasan yang bersifat *extractive*. Berikut peneliti lampirkan contoh artikel dengan *gold label* dalam ilustrasi berikut:

0	Jakarta , CNN Indonesia Dokter Ryan Thamrin , yang terkenal lewat acara D
1	Dokter Lula Kamal yang merupakan selebriti sekaligus rekan kerja Ryan men
1	Lula menuturkan , sakit itu membuat Ryan mesti vakum dari semua kegiatann
1	Kondisi itu membuat Ryan harus kembali ke kampung halamannya di Pekanbaru
0	Setahu saya dia orangnya sehat , tapi tahun lalu saya dengar dia sakit .
0	Karena sakitnya , ia langsung pulang ke Pekanbaru , jadi kami yang mau je
0	Barangkali mau istirahat , ya betul juga , kalau di Jakarta susah isiraha
0	Lula yang mengenal Ryan sejak sebelum aktif berkarier di televisi mengaku
0	Dia juga tak tahu penyakit apa yang diderita Ryan .
0	Itu saya enggak tahu , belum sempat jenguk dan enggak selamanya bisa dije
0	Enggak tahu berat sekali apa bagaimana , tutur Ryan .
0	Walau sudah setahun menderita sakit , Lula tak mengetahui apa penyebab pa
0	Meski demikian , ia mendengar beberapa kabar yang menyebut bahwa penyebab
0	" Saya tidak tahu , barangkali penyakit yang dulu sama yang sekarang berb
0	Kita kan enggak bisa mengambil kesimpulan , kata Lula .
0	Ryan Thamrin terkenal sebagai dokter yang rutin membagikan tips dan infor
0	Ryan menempuh Pendidikan Dokter pada tahun 2002 di Fakultas Kedokteran Un
0	Dia kemudian melanjutkan pendidikan Klinis Kesehatan Reproduksi dan Penya

Gambar 4.2. Contoh Format Data untuk Penelitian Extractive

4.1.2. Pre-Process Dataset

Pre-processing adalah proses menjadikan data yang sebelumnya tidak terstruktur menjadi terstruktur. Melalui proses *pre-processing* data mentah yang tersedia dalam *dataset* diolah dan di standarkan sehingga dapat diproses oleh algoritma yang akan digunakan. Sebelum data dapat diproses data harus dapat melalui beberapa tahapan seperti: *case folding*, *removing punctuation* dan *word tokenization*. Berikut adalah penjelasan dari masing-masing tahapan.

a) *Case Folding*

Tidak semua artikel konsisten dalam penggunaan huruf kapital. Dengan *case folding* semua karakter dalam teks dikonversi menjadi suatu bentuk standar (biasanya huruf kecil atau *lowercase*). Sebagai contoh dalam suatu artikel berita terdapat kata “KOMPUTER”,“Komputer”, “KomPUter”, atau “komputer”. Semuanya memiliki makna yang sama yaitu “komputer”. Namun

karena terdapat perbedaan *case* dalam karakter tersebut, komputer menganggap beberapa kata tersebut adalah kata yang berbeda walaupun sebenarnya memiliki makna yang sama. Akhirnya dengan fungsi *case folding* dapat menstandarkan kata yang sama namun memiliki *case* yang berbeda dengan menggunakan satu standar atau dalam penelitian ini menggunakan huruf kecil. Hasil proses dari tahap *case folding* dapat dilihat pada Tabel 4.2.

Table 4.2. Perbandingan *dataset* asli dengan hasil *case folding*

Input *dataset* asli	**Output *Case Folding***
Jakarta , CNN Indonesia - - Dokter Ryan Thamrin , yang terkenal lewat acara Dokter Oz Indonesia , meninggal dunia pada Jumat (4 / 8) dini hari . **Dokter Lula Kamal yang merupakan selebriti sekaligus rekan kerja Ryan menyebut kawannya itu sudah sakit sejak setahun yang lalu .** **Lula menuturkan , sakit itu membuat Ryan mesti vakum dari semua kegiatannya , termasuk menjadi pembawa acara Dokter Oz Indonesia .** **Kondisi itu membuat Ryan harus kembali ke kampung halamannya di Pekanbaru , Riau untuk menjalani istirahat .** **" Setahu saya dia orangnya sehat , tapi tahun lalu saya dengar dia sakit .** **(Karena) sakitnya , ia langsung pulang ke Pekanbaru , jadi kami yang mau jenguk juga susah .** **Barangkali mau istirahat , ya betul juga , kalau di Jakarta susah isirahatnya , " kata Lula kepada**	jakarta , cnn indonesia - - dokter ryan thamrin , yang terkenal lewat acara dokter oz indonesia , meninggal dunia pada jumat (4 / 8) dini hari . dokter lula kamal yang merupakan selebriti sekaligus rekan kerja ryan menyebut kawannya itu sudah sakit sejak setahun yang lalu . lula menuturkan , sakit itu membuat ryan mesti vakum dari semua kegiatannya , termasuk menjadi pembawa acara dokter oz indonesia . kondisi itu membuat ryan harus kembali ke kampung halamannya di pekanbaru , riau untuk menjalani istirahat . " setahu saya dia orangnya sehat , tapi tahun lalu saya dengar dia sakit . (karena) sakitnya , ia langsung pulang ke pekanbaru , jadi kami yang mau jenguk juga susah . barangkali mau istirahat , ya betul juga , kalau di jakarta susah isirahatnya , " kata lula kepada cnnindonesia.com , jumat (4 / 8) .

CNNIndonesia.com , Jumat (4 / 8) .	

b) ***Removing Punctuation***

Dalam pemrosesan kata terkadang terdapat tanda baca dalam kalimat yang sebenarnya tidak terlalu berpengaruh dalam arti kalimat tersebut. Apalagi dalam penelitian berbasis *abstractive*, setiap tanda baca dapat diartikan dalam satu suku kata. Dalam proses *generate* prediksi teks dalam *abstractive text summarization* karena setiap tanda baca berarti satu suku kata, maka akan ada kemungkinan mesin menempatkan suku kata tanda baca dalam posisi yang salah sehingga akan menimbulkan makna yang berbeda. Oleh karena itu dalam tahap ini, dilakukan proses filter terhadap tanda baca yang tidak terlalu berpengaruh kepada kalimat dan juga tanda baca yang dapat merusak makna apabila salah dalam penempatannya. Dalam penelitian ini tanda baca yang peneliti filter adalah sebagai berikut "@#$%^&*();:!-_=+[]{\/}|"`<>?". Berikut adalah hasil proses dari tahap *removing punctuation* dapat dilihat pada tabel berikut.

Table 4.3. Perbandingan *dataset* asli dengan hasil *removing punctuation*

Input *dataset* asli	**Output *Removing Punctuation***
Jakarta , CNN Indonesia - - Dokter Ryan Thamrin , yang terkenal lewat acara Dokter Oz Indonesia , meninggal dunia pada Jumat (4 / 8) dini hari . **Dokter Lula Kamal yang merupakan selebriti sekaligus rekan kerja Ryan menyebut kawannya itu sudah sakit sejak setahun yang lalu .**	jakarta , cnn indonesia dokter ryan thamrin , yang terkenal lewat acara dokter oz indonesia , meninggal dunia pada jumat 4 8 dini hari . dokter lula kamal yang merupakan selebriti sekaligus rekan kerja ryan menyebut kawannya itu sudah sakit sejak setahun yang lalu . lula menuturkan , sakit itu membuat

Lula menuturkan , sakit itu membuat Ryan mesti vakum dari semua kegiatannya , termasuk menjadi pembawa acara Dokter Oz Indonesia . **Kondisi itu membuat Ryan harus kembali ke kampung halamannya di Pekanbaru , Riau untuk menjalani istirahat .** **" Setahu saya dia orangnya sehat , tapi tahun lalu saya dengar dia sakit .** **(Karena) sakitnya , ia langsung pulang ke Pekanbaru , jadi kami yang mau jenguk juga susah .** **Barangkali mau istirahat , ya betul juga , kalau di Jakarta susah isirahatnya , " kata Lula kepada CNNIndonesia.com , Jumat (4 / 8) .**	ryan mesti vakum dari semua kegiatannya , termasuk menjadi pembawa acara dokter oz indonesia . kondisi itu membuat ryan harus kembali ke kampung halamannya di pekanbaru , riau untuk menjalani istirahat . setahu saya dia orangnya sehat , tapi tahun lalu saya dengar dia sakit . karena sakitnya , ia langsung pulang ke pekanbaru , jadi kami yang mau jenguk juga susah . barangkali mau istirahat , ya betul juga , kalau di jakarta susah isirahatnya , kata lula kepada cnnindonesia.com , jumat 4 8 .

c) *Word Tokenization*

Tokenization adalah salah satu proses dalam *preprocessing* dimana proses ini memecah teks yang pada umumnya memiliki ukuran yang besar menjadi bagian yang lebih kecil yang biasa disebut token. Dalam tokenisasi artikel kita bisa menganggap terdapat dua jenis tokenisasi, yaitu *sentence tokenization* dan *word tokenization*. *Sentence tokenization* adalah proses mambagi teks menjadi ukuran yang lebih kecil yaitu dalam bentuk kalimat, sementara *word tokenization* memecah teks dalam bentuk kata per kata. Dalam penelitian ini, dalam penelitian yang bertipe *abstractive* digunakan tokenisasi yang bertipe *word tokenization*. Tabel 4.4 adalah hasil dari proses *word tokenization*.

Table 4.4. Perbandingan *dataset* asli dengan hasil *word tokenization*

Input dataset asli	**Output *word tokenization***
Jakarta , CNN Indonesia - - Dokter Ryan Thamrin , yang terkenal lewat acara Dokter Oz Indonesia , meninggal dunia pada Jumat (4 / 8) dini hari . **Dokter Lula Kamal yang merupakan selebriti sekaligus rekan kerja Ryan menyebut kawannya itu sudah sakit sejak setahun yang lalu .** **Lula menuturkan , sakit itu membuat Ryan mesti vakum dari semua kegiatannya , termasuk menjadi pembawa acara Dokter Oz Indonesia .** **Kondisi itu membuat Ryan harus kembali ke kampung halamannya di Pekanbaru , Riau untuk menjalani istirahat .** **" Setahu saya dia orangnya sehat , tapi tahun lalu saya dengar dia sakit .** **(Karena) sakitnya , ia langsung pulang ke Pekanbaru , jadi kami yang mau jenguk juga susah .** **Barangkali mau istirahat , ya betul juga , kalau di Jakarta susah isirahatnya , " kata Lula kepada CNNIndonesia.com , Jumat (4 / 8) .**	['jakarta', ',', 'cnn', 'indonesia', 'dokter', 'ryan', 'thamrin', ',', 'yang', 'terkenal', 'lewat', 'acara', 'dokter', 'oz', 'indonesia', ',', 'meninggal', 'dunia', 'pada', 'jumat', '4', '8', 'dini', 'hari', '.', 'dokter', 'lula', 'kamal', 'yang', 'merupakan', 'selebriti', 'sekaligus', 'rekan', 'kerja', 'ryan', 'menyebut', 'kawannya', 'itu', 'sudah', 'sakit', 'sejak', 'setahun', 'yang', 'lalu', '.', 'lula', 'menuturkan', ',', 'sakit', 'itu', 'membuat', 'ryan', 'mesti', 'vakum', 'dari', 'semua', 'kegiatannya', ',', 'termasuk', 'menjadi', 'pembawa', 'acara', 'dokter', 'oz', 'indonesia', '.', 'kondisi', 'itu', 'membuat', 'ryan', 'harus', 'kembali', 'ke', 'kampung', 'halamannya', 'di', 'pekanbaru', ',', 'riau', 'untuk', 'menjalani', 'istirahat', '.', 'setahu', 'saya', 'dia', 'orangnya', 'sehat', ',', 'tapi', 'tahun', 'lalu', 'saya', 'dengar', 'dia', 'sakit', '.', 'karena', 'sakitnya', ',', 'ia', 'langsung', 'pulang', 'ke', 'pekanbaru', ',', 'jadi', 'kami', 'yang', 'mau', 'jenguk', 'juga', 'susah', '.', 'barangkali', 'mau', 'istirahat', ',', 'ya', 'betul', 'juga', ',', 'kalau', 'di', 'jakarta', 'susah', 'isirahatnya', ',', 'kata', 'lula', 'kepada', 'cnnindonesia.com', ',', 'jumat', '4', '8']

d) Pembangunan Dataset

Dataset yang sudah melalui proses *pre-processing* sebelumnya yaitu dari tahap *case folding* untuk menstandarkan ukuran, *removing punctuation* untuk pengahapusan karakter yang kurang berguna dan terakhir *word tokenization* untuk memecah teks menjadi struktur kata. Hasil output dari tahap tersebut masih belum bisa diolah oleh algoritma mesin karena data yang bisa diolah oleh mesin adalah data berbentuk angka. Oleh karena itu, pada proses pembangunan *dataset* ini data yang sudah dibersihkan dikonversi menjadi angka melalui tahap pembangunan *dataset*. Cara kerjanya adalah pertama mesin membuat kamus kata yang memuat semua kata unik yang ada di *dataset* dengan indeks berupa angka. Kemudian ada juga beberapa kata bawaan yang penting untuk algoritma seq2seq seperti:

1. <padding>: digunakan sebagai *padding* yang membuat urutan data memiliki panjang yang sama.
2. <unk>: bisa disebut *unknown*, digunakan untuk mengidentifikasi kata-kata yang tidak terdefinisi dalam *dictionary*.
3. <s>: digunakan untuk mengidentifikasi awal dari suatu kalimat.
4. </s>: digunakan untuk mengidentifikasi akhir dari kalimat.

Selain kata-kata standar tersebut didefinisikan sebagai kata unik dengan indeks. Tahap kedua setelah dibuat kamus kata dari *dataset* adalah menerjemahkan setiap kata dari input berdasarkan kamus kata yang telah dibuat. Dalam penerjemahan artikel data distandarkan berdasarkan artikel

dengan jumlah kata paling banyak, jadi apabila terdapat artikel dengan jumlah kata yang kurang dari jumlah standar artikel maka akan ditambahkan *padding* untuk membuat jumlah kata menjadi standar. Dalam hal ini *padding* didefinisikan sebagai angka 0. semakin banyak kata dalam dataset semakin kompleks kamus data yang dibuat. Tabel 4.5 merupakan hasil konversi dataset berdasarkan kamus kata dalam proses pembangunan dataset.

Table 4.5. Perbandingan *dataset* asli dengan hasil pembangunan *dataset*

Input dataset asli	**Output *Build Dataset***
Jakarta , CNN Indonesia - - Dokter Ryan Thamrin , yang terkenal lewat acara Dokter Oz Indonesia , meninggal dunia pada Jumat (4 / 8) dini hari . **Dokter Lula Kamal yang merupakan selebriti sekaligus rekan kerja Ryan menyebut kawannya itu sudah sakit sejak setahun yang lalu .** **Lula menuturkan , sakit itu membuat Ryan mesti vakum dari semua kegiatannya , termasuk menjadi pembawa acara Dokter Oz Indonesia .** **Kondisi itu membuat Ryan harus kembali ke kampung halamannya di Pekanbaru , Riau untuk menjalani istirahat .** **" Setahu saya dia orangnya sehat , tapi tahun lalu saya dengar dia sakit .** **(Karena) sakitnya , ia langsung pulang ke Pekanbaru , jadi kami yang mau jenguk juga susah .** **Barangkali mau istirahat , ya betul juga , kalau di Jakarta susah isirahatnya , " kata Lula kepada**	[[51, 4, 58, 13, 7, 6, 28, 4, 8, 59, 60, 16, 7, 17, 13, 4, 61, 62, 63, 52, 14, 14, 64, 65, 5, 7, 10, 22, 8, 23, 24, 25, 26, 27, 6, 29, 30, 9, 31, 11, 32, 33, 8, 15, 5, 10, 34, 4, 11, 9, 12, 6, 35, 36, 37, 38, 39, 4, 40, 41, 42, 16, 7, 17, 13, 5, 43, 9, 12, 6, 44, 45, 18, 46, 47, 19, 20, 4, 48, 49, 50, 21, 5, 66, 53, 54, 67, 68, 4, 69, 70, 15, 53, 71, 54, 11, 5, 72, 73, 4, 74, 75, 76, 18, 20, 4, 77, 78, 8, 55, 79, 56, 57, 5, 80, 55, 21, 4, 81, 82, 56, 4, 83, 19, 51, 57, 84, 4, 85, 10, 86, 87, 4, 52, 14, 14, 5]]

CNNIndonesia.com , Jumat (4 / 8) .	

4.1.3. Vektorisasi Dataset

Setelah *pre-processing*, hasil *pre-process* diubah menjadi vektor. vektorisasi *dataset* adalah suatu metode untuk merepresentasikan setiap kata di dalam konteks sebagai vektor dengan N demensi. Hal ini supaya teks dapat dihitung dalam proses *training* selain teks harus diubah menjadi kamus angka, teks tersebut juga harus diubah menjadi bentuk vektor. Metode vektorisasi yang digunakan adalah *word embedding vector*. *word embedding* vektor memetakan konteks kata ke dalam vektor. Hasil dari *embedding* kontekstual vektor ini dapat merepresentasikan relasi suatu kata dengan kata lainnya, misalnya relasi antara 'Pria — Wanita', relasi pada 'Kata Kerja', dan bahkan relasi pada 'Indonesia — Negara'

Metode *word embedding* yang digunakan adalah Word2Vec. Model vektorisasi Word2Vec yang peneliti gunakan dilatih dengan 403950 artikel dari Wikipedia berbahasa Indonesia, dimensi vektor (ukuran *embedding*) adalah 50 dimensi. Tabel 4.6 merupakan contoh representasi vektor dari beberapa kata berdasarkan hasil train dari Wikipedia berbahasa Indonesia.

Table 4.6. Contoh representasi vektor

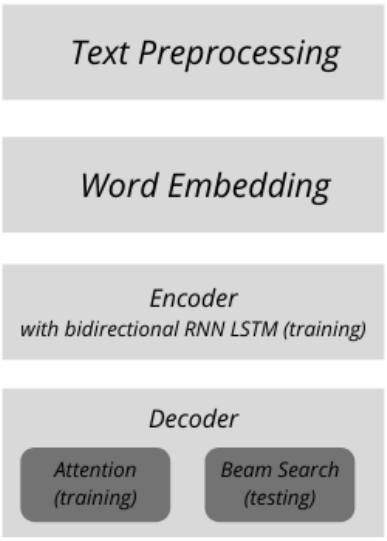
Kata	**Vektor**
Rumah	[2.2080677 5.9375963 3.0163329 -0.13347664 -0.6860644 -3.2368624 2.088639 1.6504346 -0.45981097 -2.5707371 -4.844986 -4.378362

	-4.4189863 -3.482012 -1.7284296 -2.2496796 0.98841006 -3.2315104 -0.8973775 -4.877335 -2.04818 5.4241524 -3.8995316 -2.0226285 2.725587 -2.5911293 -1.5482998 -1.273208 1.6526736 -3.6218865 -0.08282666 0.5218348 1.4562945 0.4869153 -2.7536728 3.8568883 -3.0713623 2.0960982 5.737881 2.160936 0.81659746 1.9070693 3.4118695 5.55899 -0.15873995 0.67162186 -0.04977261 2.8539495 2.5592878 0.72608334]
jakarta	[-2.2774374 -1.2183187 -2.748263 -0.6970468 -3.7465017 2.1984076 -2.789526 2.628822 -6.2247024 -1.9940684 -1.5774819 -1.9201723 -1.5976936 -0.46484843 2.0923617 3.2037735 -1.8428062 -3.7260659 -3.5887117 -2.0001674 -0.29422575 3.0571246 1.4943291 -0.12267653 5.495486 -2.2410698 4.2619886 1.5997388 0.944478 -3.5597596 -0.7232965 2.3565588 -2.526716 -1.6067371 0.90881 2.2456794 3.0516498 0.3505161 6.033811 -0.17425568 -1.8031099 1.0186446 3.7994573 -0.9442943 -2.7385268 -0.7830452 3.100348 2.0142965 4.9273367 7.0368404]
kejahatan	[1.603156 4.018308 -3.289746 0.30599886 -2.8163593 -1.5907574 1.3292934 -1.2934152 -3.1686685 -0.2643346 -1.2859949 -1.0873827 1.5501179 -0.03327138 0.26203543 3.0684319 1.9282238 1.8473293 3.6663547 -1.4522858 -0.55322963 -2.3691757 1.2062808 -1.7761946 0.48265177 3.7315207 -0.79523045 -5.026902 1.2098378 -2.8374023 1.5549101 -2.2385414 -2.4040155 3.58892 0.21941297 3.1324666 -2.8109906 3.9464583 3.9624128 -4.964247 -3.2801576 -1.2435125 0.8063947 -0.58228344 0.12501957 4.928639 -0.04115057 -0.06649492 2.4687917 0.2886443]
cantik	[2.5384598 -1.5758759 1.8332925 1.4674206 1.0479904 -1.532768

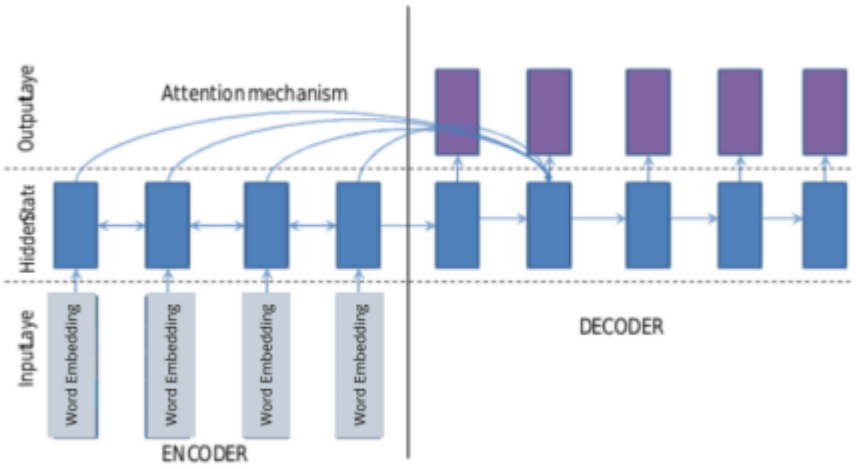
```
 0.6177877  0.74130464 -3.2464516  1.9923546 -
2.2411015 -0.59712845
 -4.222497  -1.3206335 -6.811005  4.98674
0.9436768  0.8309835
 -1.8828291 -3.8073003 -0.08299697 5.623486
1.4260517 -2.2019346
 -1.4167784 -1.195744  -3.1232233 -0.8235017
1.014537  0.67176104
 -1.6829811  2.9967372  0.02289255 3.8838105
1.3626992  1.8969612
 0.87771213 -0.5098602  2.1533728  0.792993
3.2023077  0.14339748
 1.7490079 -2.962115  2.291612  4.1231713 -
0.8818835  1.9443401
 0.5394802 -2.1112773 ]
```

4.1.4. *Training Model*

Metode yang digunakan adalah metode berbasis *sequence-to-sequence* yaitu RNN. RNN pada umumnya menghasilkan keluaran dengan panjang yang sama dengan masukan. Namun, karena hasil ringkasan harus memiliki panjang yang berbeda dari input, maka digunakan mekanisme *encoder-decoder*. Gambar 4.3 adalah ilustrasi model yang digunakan dalam penelitian ini.



Gambar 4.3. Struktur Model



Gambar 4.4. Ilustrasi model *sequence-to-sequnce*

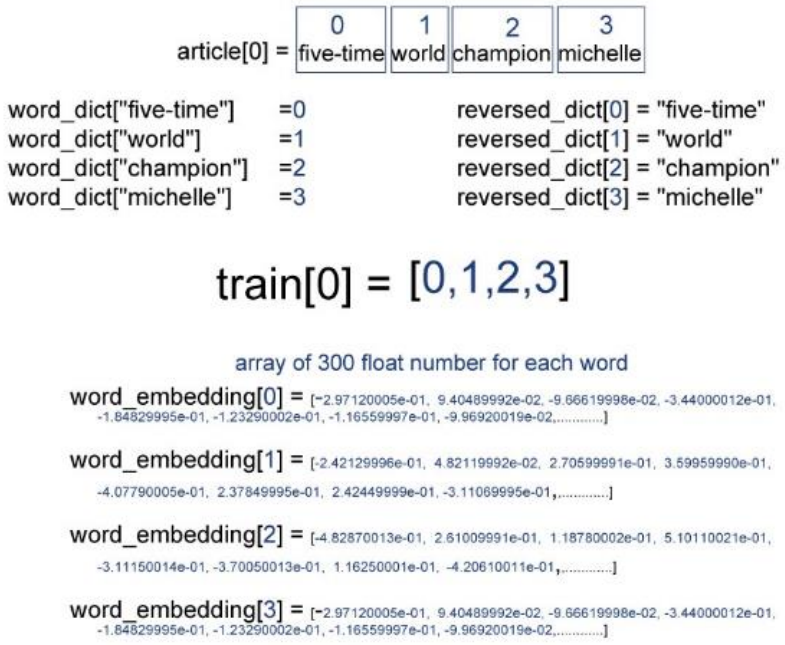
Berikut adalah penjelasan dari ilustrasi model pada Gambar 4.4.

❑ ***Input embedding***

Metode dimulai dari persiapan *dataset* yang akan digunakan sebagai

dataset training. Berdasarkan langkah sebelumnya *dataset* terlebih dahulu melalui beberapa langkah *preprocessing* sebelum dapat diproses dalam proses *training model*. Tahap *preprocessing* berfungsi untuk menstandarkan input data agar bisa diproses oleh mesin. *preprocessing* data dalam penelitian ini meliputi*: case folding, removing punctuation, word tokenization.*

Setelah data dibersihkan dan melalui proses penstandaran tahap selanjutnya adalah vektorisasi *dataset*. Melalui proses vektorisasi *dataset* kata diubah menjadi vektor yang bisa dihitung dan bahkan diketahui bagaimana hubungan antar kata-kata. Berikut adalah contoh hasil dari output *text preprocessing* dan vektorisasi *dataset*.

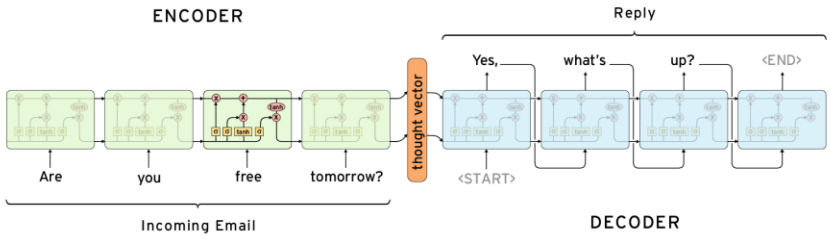


Gambar 4.5. Contoh hasil *preprocessing dataset*

❑ ***Sequence-to-Sequence Network***

Dalam penelitian ini menggunakan metode berbasis *sequence-to-sequence*. Basis dari *Sequence-to-Sequence* sebenarnya adalah RNN. RNN berbeda dengan *neural network* biasa, RNN bisa mengenali entitas karena RNN adalah *neural network* yang memperhatikan urutan dan waktu. selain itu RNN juga memperhatikan output dari jaringan sebelumnya.

Alasan menggunakan metode *sequence-to-sequence* adalah karena jaringan *sequence-to-sequence* dapat menerima teks dengan panjang input (Tx) dan mengasilkan output dengan panjang yang berbeda (Ty), atau biasa disebut *encoder-decoder*. Kedua *encoder decoder* disini merupakan jaringan RNN, *encoder* menggunakan input dan menghasilkan state output yang kemudian dijadikan input ke tahap *decoder*. Berikut adalah ilustrasi dari metode *encoder-decoder*.

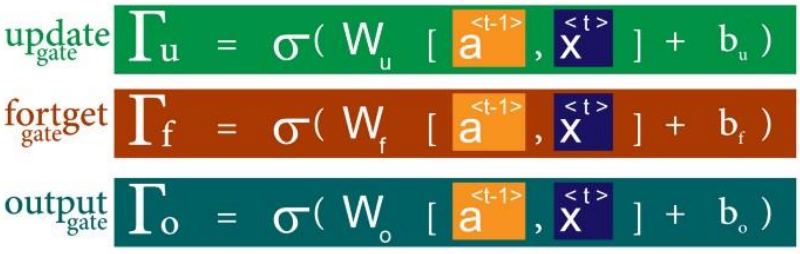


Gambar 4.6. Ilustrasi *encoder-decoder*

❑ **RNN *Modifications* (LSTM)**

Ada dua masalah utama dalam RNN:

1. *Exploding Gradients:* sering terjadi pada jaringan dengan banyak *layer* yaitu ketika menerapkan *backpropragation*, gradient jaringan akan menjadi terlalu besar.
2. *Vanishing Gradients:* masalah yang sering ditemukan pada jaringan dengan banyak *layer*, namun pada *vanishing gradient* masalah utama lebih sering terjadi karena jaringan yang terlalu panjang sehingga akan ada memori yang hilang pada unit RNN normal karena panjangnya jaringan. Masalah di atas cukup penting dalam permasalahan NLP. Untuk mengatasi masalah tersebut dibuatlah arsitektur modifikasi jaringan RNN dengan LSTM. LSTM dibangun dengan konsep jaringan yang dapat mengingat urutan yang panjang. dalam LSTM memori dikontrol dalam tiga gerbang berbeda, Gambar 4.7 adalah ilustrasi LSTM.



Gambar 4.7. Ilustrasi LSTM

Ketika kita menghubungkan beberapa LSTM bersama-

sama, kita dapat melihat bahwa jika jaringan dapat mempelajari parameter gerbang dengan benar, kita dapat meneruskan nilai dari awal urutan hingga ke akhir urutan, sehingga kita dapat memodelkan urutan yang panjang dengan akurasi tinggi

❏ ***Bidirectional Network***

Bidirectional network adalah salah satu modifikasi RNN dimana jaringan memiliki dua input jaringan yaitu *input forward* dan *input backward*. Karena dalam pengaplikasiannya dalam memahami kata atau kalimat kita tidak hanya harus memahami kata sebelumnya tapi juga harus memahami kata yang akan datang. Berikut adalah ilustrasi *bidirectional network* (lihat Gambar 4.8):

Gambar 4.8. Ilustrasi *bidirectional network*

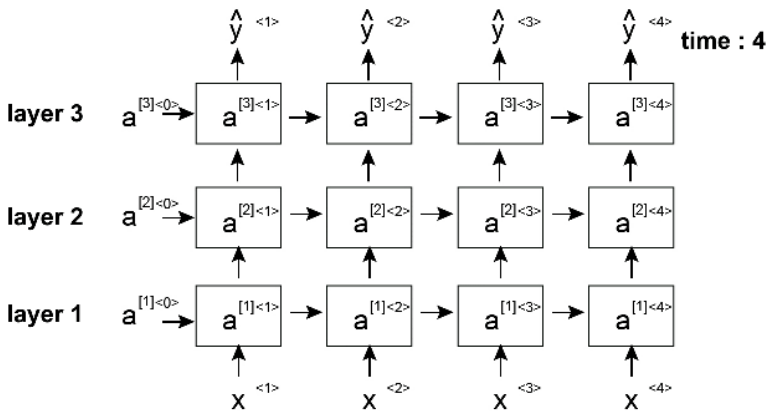
Dari dua kalimat di atas ada dua perbedaan makna, pada kalimat pertama kata tedy bukan berarti nama orang akan tetapi merupakan

bagian dari Teddy bears, sedangkan pada kalimat kedua kata teddy merupakan bagian dari nama orang. Hal ini tergantung dari kata apa yang terdapat di sebelum dan sesudah kata tersebut. hal ini menjadikan *bidirectional RNN* menjadi penting untuk memahami kalimat secara lebih kompleks. Dalam penelitian ini *bidirectional network* digunakan pada *encoder* khususnya pada proses *training*.

❑ ***Multilayer Networks***

Untuk mencapai hasil yang lebih maksimal, kita dapat menumpuk beberapa RNN (LSTM atau RNN normal) di atas satu sama lain, tapi kita tetap harus memperhatikan tiap layer berkerja dengan waktu yang beriringan.

semakin banyak layer yang ada dalam RNN maka akan semakin kompleks kombinasi learning yang bisa dibuat, namun hal tersebut berbanding terbalik dengan biaya komputasi. semakin banyak layer maka akan semakin mahal komputasinya. Gambar 4.9 merupakan ilustrasi *multilayer network* pada RNN.



Gambar 4.9. Ilustrasi *multilayer network*

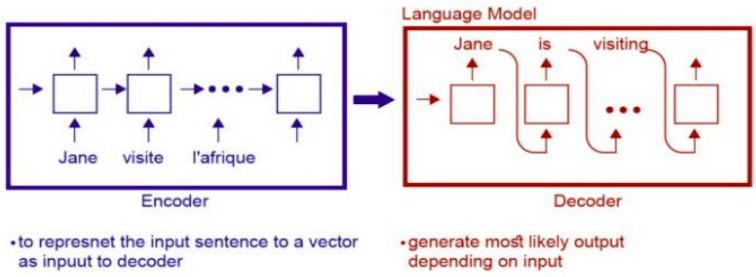
❑ ***Beam Search & Attention Model***

Dalam tahap *beam search & attention mechanism* kita melakukan modifikasi pada tahap inti model *encoder-decoder*. Dengan adanya modifikasi ini dapat membantu jaringan memilih hasil terbaik dari beberapa kemungkinan yang berbeda yang dikenal dengan *beam search.* Dengan adanya *attention mechanism* dapat memfilter hasil ringkasan untuk menghasilkan hasil ringkasan yang lebih baik melalui jaringan sederhana yang ditambahkan. Berikut adalah penjelasan dari *beam search & attention mechanism*:

1. *Beam Search*

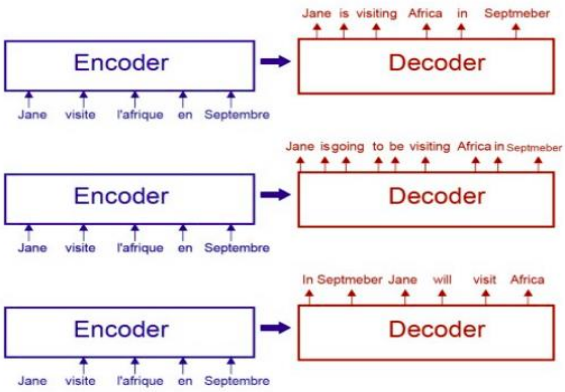
   Dalam menghasilkan output dari peringkasan teks kita menggunakan model bahasa bersyarat, yang artinya output kata hasil keluaran merupakan kondisional dari kata input. Model

bahasa yang dikeluarkan bersifat probabilitas dari inputan. Jadi arsitektur sebenarnya dapat dibagi menjadi dua bagian yaitu *encoder* dan *decoder*. *Encoder* merepresentasikan kalimat masukan sebagai vektor, dan meneruskannya ke bagian arsitektur berikutnya (*decoder*). Gambar 4.10 ilustrasi urutan kata-kata dalam *encoder-decoder*.



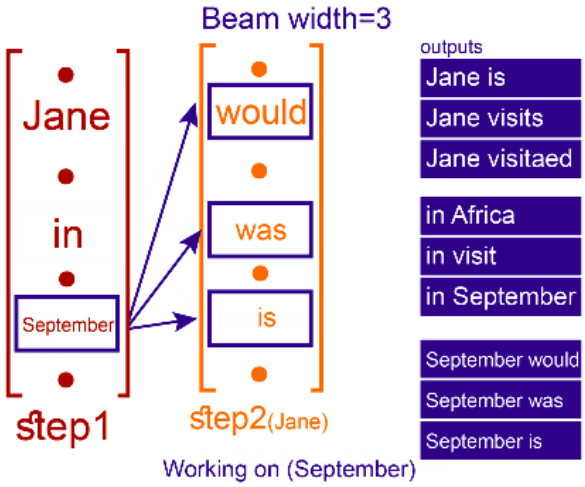
Gambar 4.10. Ilustrasi kata-kata dalam *encoder-decoder*

Masalah sebenarnya ada pada *decoder*, yaitu tentang bagaimana menentukan kalimat mana yang harus dipilih. Karena akan ada output yang besar sekali dari suatu input tertentu atau bisa diilustrasikan pada Gambar 4.11.



Gambar 4.11. Ilustrasi output berdasarkan kondisional input

Solusinya adalah dengan menghasilkan satu kata pada satu waktu, dengan cara memilih kata dengan presentase terbaik, atau bisa disebut dengan *greedy search*. Tapi disisi lain *greedy search* masih mempunyai kelemahan yaitu belum bisa mempertimbangkan kemungkinan kata dalam *database*. Untuk mengatasi masalah tersebut kita menggunakan *beam search*. Konsep *beam search* serupa dengan *greedy search* namun *beam* dalam *beam search* dalam menghasilkan output tidak hanya memilih satu kata terbaik sebagai output tetapi memperhatikan beberapa opsi kata yang disebut *beam width*. Gambar 4.12 adalah ilustrasi dari metode *beam search*.



Gambar 4.12. Ilustrasi *beam search* dengan *width* 3

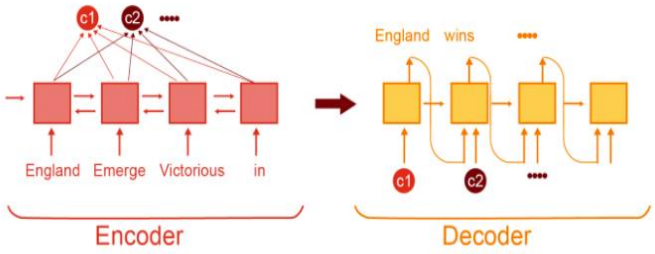
semakin banyak dimensi dari *beam width* maka akan semakin kompleks kombinasi kata yang bisa dihasilkan. Namun semakin tinggi dimensi dari *beam width* akan semakin memperberat komputasi. Dalam penelitian ini peneliti menggunakan *beam width* 3 kata karena mempertimbangkan kecepatan komputasi mesin. Pada penelitian ini *beam search* digunakan pada bagian *decoder* khususnya pada proses *testing*.

2. *Attention Model*

   Ketika kita sebagai manusia meringkas teks, cara kita meringkas sebenarnya melihat beberapa kata pada satu waktu, bukan keseluruhan teks untuk diringkas pada waktu tertentu, inilah yang dinamakan *attention model*. *Attention model* mencoba

untuk mengajarkan model untuk hanya memperhatikan kata-kata yang berkaitan bukan seluruh teks.

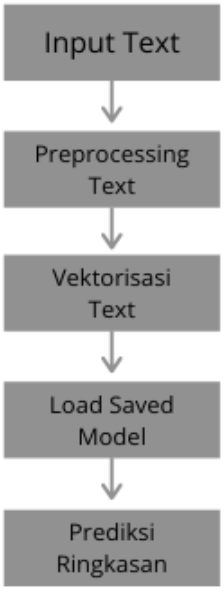
Dalam struktur *encoder-decoder*, proses *attention model* menjadi *interface* baru antara *encoder* dan *decoder*, *interface* ini disebut vektor konteks, yang sebenarnya mewakili jumlah *attention* yang diberikan pada kata-kata. Pada penelitian ini *attention mechanism* digunakan pada bagian *decoder* khususnya ketika proses *training*. Gambar 4.13 adalah ilustrasi *attention model* pada struktur *encoder-decoder*.



Gambar 4.13. *Sequence-to-sequence* dengan *attention mechanism*

4.1.5. Prediksi Ringkasan

Tahap selanjutnya setelah dibuat model dan melalui proses *training* adalah tahap prediksi ringkasan. Tahap prediksi ringkasan adalah tahap dimana algoritma menghasilkan output berupa hasil prediksi dari perhitungan input yang diproses oleh model yang sudah disimpan. Gambar 4.14 adalah ilustrasi dari alur proses prediksi ringkasan pada penelitian ini.



Gambar 4.14. *Flowchart* alur prediksi model

Gambar 4.14 adalah alur proses prediksi ringkasan pada penelitian ini. Dalam Gambar 4.14 dijelaskan bahwa dalam prediksi ringkasan dimulai dari input berupa teks yang akan diringkas, atau dalam penelitian ini adalah data teks berita. Tahap selanjutnya adalah teks melalui proses penstandaran input agar dapat di proses oleh mesin atau dalam dalam hal ini proses tersebut dapat dinamakan proses *pre-processing*. Setelah teks dibersihkan dan difilter dari data-data yang tidak diperlukan, tahap selanjutnya adalah tahap vektorisasi. tahap vektorisasi adalah tahap mengubah teks yang sudah dibersihkan menjadi angka vektor. dengan mengubah teks menjadi vektor angka dapat menjadikan

teks tersebut dapat diolah oleh mesin. Setelah teks berubah menjadi vektor angka tahap berikutnya adalah proses *load* model. Dalam proses *training* data model disimpan dalam sebuah direktori sehingga bisa menyimpan pola yang sudah dihasilkan melalui proses *training*.

Tahap terakhir dari proses prediksi adalah tahap prediksi itu sendiri. tahap prediksi adalah tahap dimana model yang sudah dimuat diberikan input berupa teks yang sudah melalui proses *pre-processing* dan vektorisasi. Teks yang sudah divektorisasi akan dihitung dan diolah berdasarkan pola di dalam model yang sudah di simpan. Tabel 4.7 adalah contoh input berita dan perbandingan teks hasil ringkasan yang dibuat oleh mesin dengan teks hasil ringkasan asli dari manusia.

Table 4.7. Contoh data input

Judul Berita	**Ketua MPR: Mustahil Transportasi Tradisional dan Online Bersaing Bebas**
Isi Berita	Ketua MPR Zulkifli Hasan menyesalkan kisruh yang terjadi antara pelaku sarana transportasi online dan tradisional. Zulkifli menyarankan adanya pertemuan bersama antara pemerintah, pelaku transportasi online dan transportasi tradisional demi meredam kisruh yang masih belum terselesaikan. Zulkifli menilai aturan yang dikeluarkan pemerintah seharusnya tidak hanya membahas tarif tapi juga mekanisme yang dapat menguntungkan semua pihak, baik pelaku transportasi online maupun tradisional. "Tidak hanya tarif tapi apa saja harus diatur. Dipanggil keduanya untuk berbicara masing-masing, musyawarah, duduk bareng kemudian dibuat aturan yang saling menguntungkan. Kan bisa saling melengkapi, negara lain bisa masa kita enggak bisa," ucap Zulkifli di Gedung DPR, Senayan, Jakarta Pusat, Senin (27/3). Ketua Umum PAN menambahkan bahwa hal ini harus

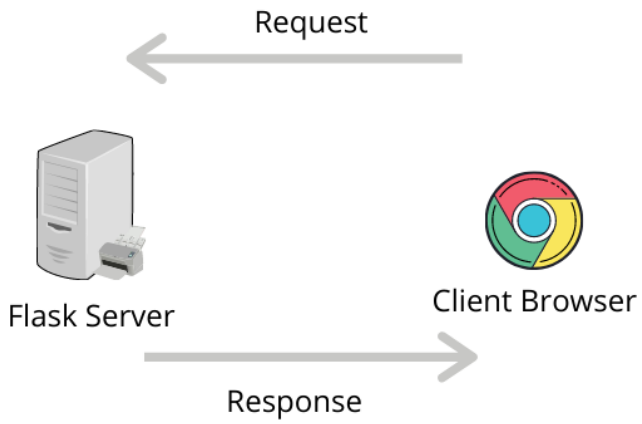
	diatur karena menyangkut mata pencaharian dari masyarakat itu sendiri. Menurut dia, titik temu masih bisa dicari. Sebab, transportasi online tradisional masih memiliki pasar sendiri dan transportasi tradisional juga memiliki pasarnya sendiri. "Yang tradisional tentu akan ada pasar sendiri, yang online punya pasar sendiri. Nah itulah segera ditata kalau tidak tentu terjadi bentrokan-bentrokan. Itu namanya urusan perut, urusan rezeki tentu kalau tidak ditata dengan baik akan menimbulkan keresahan," lanjut Zulkifli. Zulkifli menilai bahwa tidak mungkin bila keduanya bertarung bebas. Diperlukan adanya aturan yang membahas hal tersebut. "Itu mesti diatur kan kalau semua tarung bebas kan enggak mungkin. Kalau semua dibiarkan tarung bebas atau disebut pasar bebas itu tentu nanti yang tradisional akan teriak, akan menjerit," ujar Zulkifli.

Table 4.8. Perbandingan hasil output mesin dengan ringkasan asli manusia

Ringkasan Asli Manusia	**Hasil Prediksi Mesin**
Ketua MPR Zulkifli Hasan menyesalkan kisruh yang terjadi antara pelaku sarana transportasi online dan tradisional. Zulkifli menyarankan adanya pertemuan bersama antara pemerintah, pelaku transportasi online dan transportasi tradisional demi meredam kisruh yang masih belum terselesaikan. Zulkifli menilai aturan yang dikeluarkan pemerintah seharusnya tidak hanya membahas tarif tapi juga mekanisme yang dapat menguntungkan semua pihak, baik pelaku transportasi online maupun tradisional.	Ketua mpr zulkifli hasan menyesalkan kisruh yang terjadi antara pelaku sarana transportasi online dan tradisional . menyarankan adanya pertemuan bersama pemerintah , demi meredam gejolak masih belum terselesaikan sebab aktivitas memiliki pasar sendiri maupun juga pasarnya menilai bahwa tidak mungkin bila keduanya bertarung bebas di gedung dpr senayan jakarta.

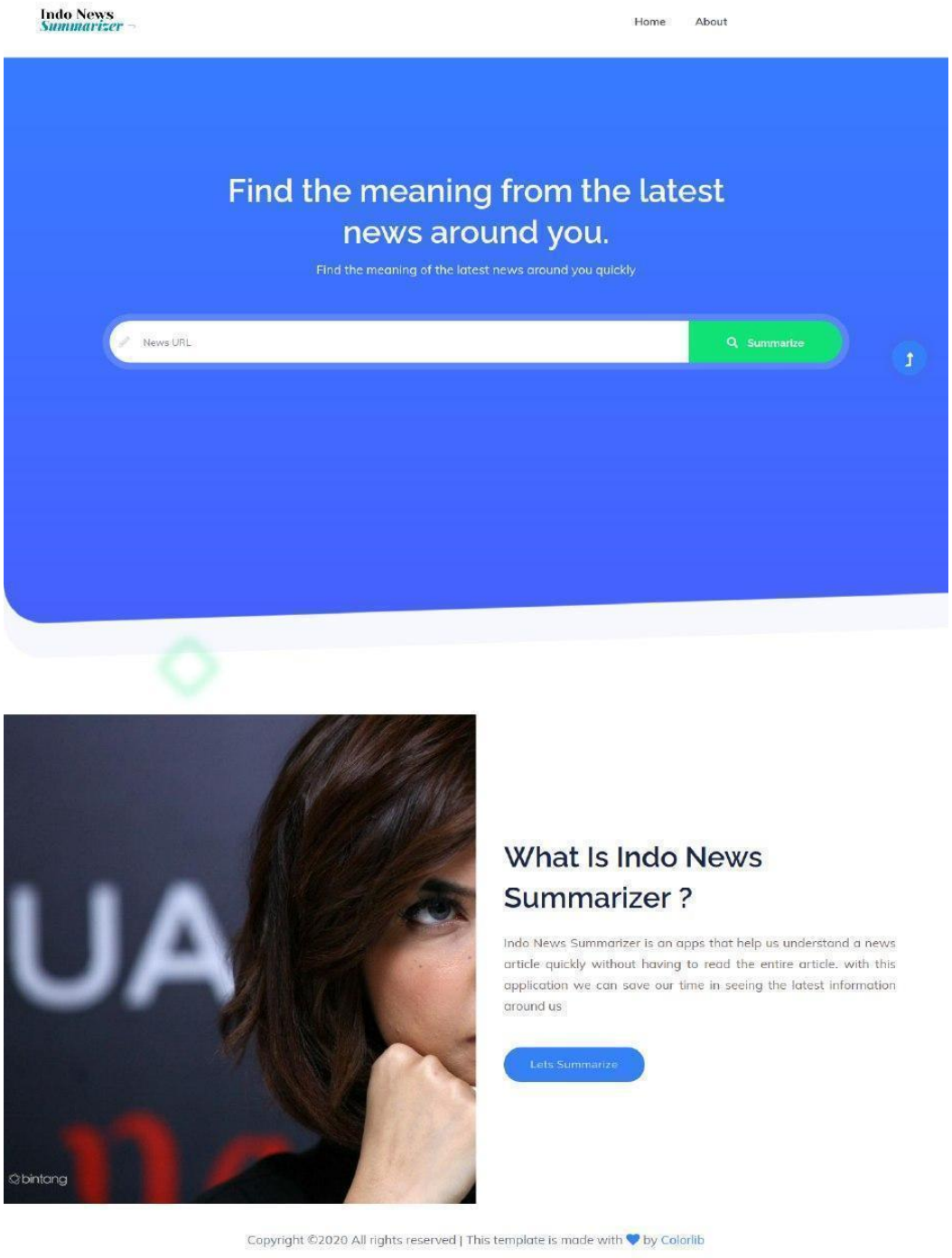
4.1.6. Implementasi Sistem

Langkah selanjutnya adalah tahap implementasi sistem secara kesuluruhan. Dalam implementasinya peneliti menggunakan bahasa *Python* dengan Framework Flask. Arsitektur sistem kali ini dibagi menjadi *client* dan server, server berfungsi sebagai penyedia layanan dalam aplikasi kemudian client bertugas melakukan *request* dan menerima *response* dari server. untuk komunikasi antara *client* dan server menggunakan metode REST API, cara kerja REST API adalah *client* mengirimkan *request* kepada server melalui suatu url yang disediakan oleh server kemudian server mengirimkan *response* kepada client berdasarkan apa yang client minta. Pemilihan Flask menjadi *tools* dalam penelitian ini adalah karena Flask merupakan *micro framework* berbasis *python* yang tepat untuk arsitektur REST API. Gambar 4.15 adalah ilustrasi arsitektur aplikasi yang digunakan dalam penelitian ini.



Gambar 4.15. Arsitektur aplikasi yang akan dibuat

Sisi *client* bertugas untuk mempermudah *user* melakukan *request* pada aplikasi, oleh karena itu diperlukan *design interface* yang menarik dan mudah dipahami oleh *user* supaya *client* dapat dengan mudah untuk mendapatkan apa yang mereka inginkan. Dalam penelitian ini aplikasi dibuat dengan *single page aplication* sederhana karena memang fitur dalam aplikasi kali ini adalah hanya untuk mencari makna atau meringkas artikel berita, sehingga diperlukan *user interface* yang sesederhana mungkin. Gambar 4.16 adalah *user interface* aplikasi dalam penelitian ini.



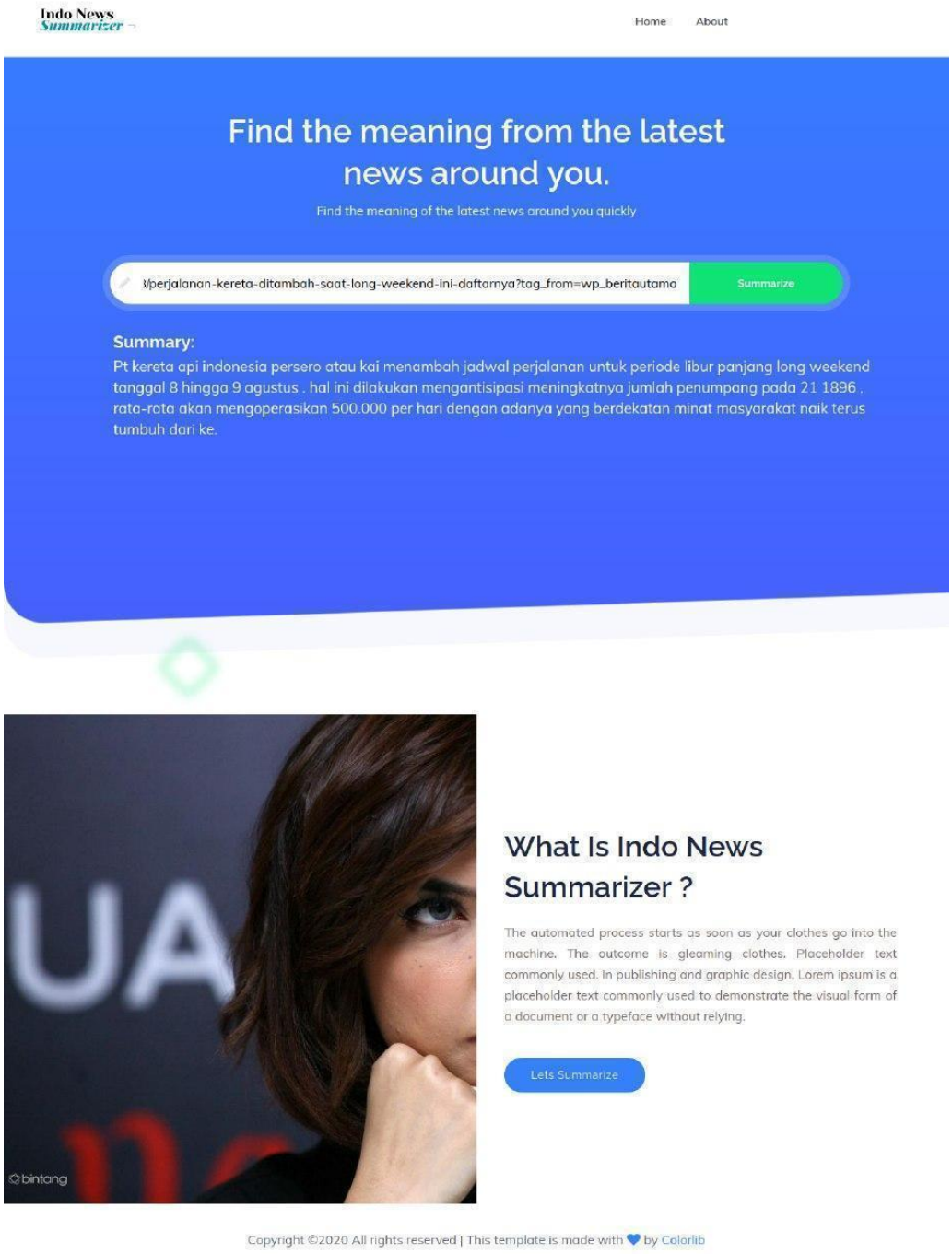
Gambar 4.16. *Homepage* aplikasi peringkas berita

Penjelasan dari *user inteface* seperti ditunjukkan pada Gambar 4.16

adalah aplikasi *client* adalah berupa *one page application* atau aplikasi yang terdiri dari hanya satu halaman. Dalam halaman tersebut terbagi menjadi dua bagian penting yaitu *search section* dan *about section*, berikut penjelasan kedua bagian tersebut.

1. *Search section*: atau bagian pencarian merupakan bagian penting yang berisi form untuk memasukkan url berita yang akan diringkas. Pada input teks tersebut kita bisa memasukkan url kemudian jika sudah bisa kita tekan tombol *summarize* untuk mengetahui hasil dari peringkasan berita yang ada di input form tersebut.
2. *About section:* merupakan bagian yang menjelaskan tentang detail aplikasi yaitu penjelasan tentang profil dan fungsi utama dari aplikasi itu sendiri.

Setelah *user* memasukkan url berita kedalam kotak input *user* bisa menekan tombol *summarize* untuk melakukan proses peringkasan berita. Ketika proses peringkasan berjalan akan ada proses tampilan loading dari aplikasi yang menandakan proses sedang berjalan. Kemudian setelah proses selesai output dari ringkasan tersebut akan langsung ditampilkan tepat di bawah form input. Gambar 4.17 adalah ilustrasi hasil dari output ringkasan.



Gambar 4.17. Hasil output aplikasi peringkas berita

4.2. Pembahasan

Penelitian ini mengambil topik *automatic text summarization* berbahasa Indonesia dengan berbasis *abstractive*. Untuk metode yang digunakan dalam penelitian ini menggunakan pendekatan *deep learning.* Algoritma *deep learning* yang digunakan dalam penelitian ini adalah *sequence-to-sequence* dengan RNN LSTM. *Dataset* yang digunakan adalah *dataset* publik dengan jumlah cukup besar dan sebaran data yang cukup bervariasi sehingga hasil dari penelitian ini dapat dijadikan *benchmarking* untuk penelitian-penelitian sejenis.

Metode evaluasi dalam penelitian ini adalah ROUGE. ROUGE sendiri adalah standar evaluasi penelitian berbasis *text summarization.* Cara kerja ROUGE adalah dengan cara membandingkan hasil output ringkasan dari mesin dengan buatan asli manusia. Dalam penelitian ini metric ROUGE yang peneliti gunakan adalah ROUGE-N. Tabel 4.9 adalah hasil perbandingan evaluasi penelitian terkait dengan hasil dari peneliti menggunakan *metric* ROUGE. Untuk *range output* dari ROUGE sendiri adalah antara 0 sampai 1. Semakin hasil evaluasi mendekati angka satu maka semakin mirip hasil ringkasan mesin dengan buatan asli manusia.

Table 4.9. perbandingan hasil evaluasi model peneliti dengan penelitian sejenis

Metode	Peneliti	Bahasa	Dataset	R-1	R-2
Extractive	Cai et al., (2019)	Indonesia	Publik (IndoSum)	0.29	0.071
	Garmastewira., (2020)	Indonesia	Tidak disebutkan	-	0.370
	Kurniawan & Louvan., (2018)	Indonesia	Publik (IndoSum)	0.46	0.48
Abstractive	Adelia et al., (2019)	Indonesia	Tidak Publik	0.1197	0.0119
	Ivanedra & Mustikasari., (2019)	Inggris	Tidak Publik	0.44	-
	Nallapati et al., (2016)	Inggris	Publik (CNN/Dailymail Corpus)	0.3546	0.1664
	Model Peneliti	**Indonesia**	**Publik (IndoSum)**	**0.5371**	**0.3757**

Penjelasan dari hasil perbandingan penelitian terkait adalah untuk penelitian *automatic text summarization* berbahasa Indonesia masih belum terlalu banyak, rata-rata penelitian di dominasi oleh penelitian yang berbasis *extractive*. Untuk penelitian berbasis *abstractive* yang berbahasa Indonesia sendiri peneliti menggunakan model sequence-to-sequence dari Adelia et al (2019) sebagai referensi karena Adelia sama-

sama menggunakan metode berbasis *deep learning*. Untuk *dataset* sendiri masih sedikit yang menggunakan *dataset* publik untuk *automatic text summarization* berbahasa Indonesia, sehingga sulit untuk dijadikan *benchmarking* pada penelitian yang akan datang.

Berdasarkan hasil evaluasi peneliti menggunakan R-1 dan R-2 yang mana R-1 mengambarkan kemiripan *output* dengan cara membandingkan kata perkata sedangkan R-2 menggunakan perbandingan kombinasi dua kata sehingga lebih meningkatkan keterbacaan kalimat. Untuk hasil evaluasi penelitian yang peneliti gunakan untuk perbandingan adalah penelitian yang mempunyai beberapa kesamaan antara lain. sama-sama *automatic text summarization* berbahasa Indonesia. Sama-sama menggunakan metode *deep learning*. Berikut adalah perbandingannya:

1. Perbandingan *text summarization* berbahasa Indonesia: untuk *automatic text summarization* berbahasa Indonesia peneliti menggunakan sampel penelitian berbasis *abstractive* dan *extractive* yaitu Adelia et al., (2019), Kurniawan & Louvan., (2018), Cai et al., (2019) dan Garmastewira., (2020). Untuk hasilnya evaluasi penelitian peneliti memiliki hasil yang paling tinggi pada metric R-1 namun untuk metric R-2 hasil peneliti masih kalah dengan penelitian Kurniawan & Louvan., (2018).
2. Perbandingan *text summarization* berbasis *deep learning*: untuk perbandingan ini peneliti mengambil sampel penelitian yang menggunakan metode serupa yaitu berbasis *deep learning* dengan

metode *sequence-to-sequence*. Untuk hasilnya output peneliti memiliki hasil evaluasi yang yang cukup tinggi.

BAB V

PENUTUP

5.1 Simpulan

Berdasarkan hasil penelitian dan pembahasan, dapat ditarik kesimpulan sebagai berikut:

1. Dalam penelitian ini peneliti meneliti mengusulkan metode berbasis *deep learning* bernama *sequence-to-sequence* untuk menyelesaikan masalah *abstractive text summarization* berbahasa Indonesia. Cara kerja *sequence-to-sequence* atau biasa disebut *encoder decoder* adalah mesin membaca input berupa urutan/*sequence* kemudian mesin mengasilkan output berupa urutan/*sequence*. Dengan jaringan model yang sudah disusun sebelumnya mesin mempelajari pola antara urutan input dan output yang dihasilkan melalui data *training*.

   Neural network dalam *sequence-to-sequence* adalah yang bertipe RNN karena RNN adalah *neural network* yang paling cocok untuk jaringan yang berupa urutan. Kemudian untuk mengatasi masalah dasar pada RNN yang memiliki jaringan yang panjang seperti *exploding gradient* dan *vanishing gradient* dalam RNN ini juga ditambahkan LSTM sebagai cellnya untuk mengatasi masalah tersebut. Untuk *encoder* jaringan dibuat *bidirectional* atau bisa memiliki dua arah yaitu kedepan dan kebelakang, hal ini digunakan untuk meningkatkan

kualitas *learning* jaringan RNN.

Kemudian karena panjang urutan antara *encoder* dan *decoder* memiliki perbedaan yang cukup besar digunakan mekanisme *attention mechanism* atau dalam menghasilkan output mesin memilih kata-kata yang berkaitan untuk memfilter kata-kata. Untuk menghasilkan prediksi hasil ringkasan yang baik digunakan mekanisme *beam search* atau dalam memprediksi kata mesin menggunakan n kemungkinan kata atau sebanyak lebar dari *beam width*. Semakin lebar maka output akan semakin baik namun berbanding terbalik dengan kecepatan komputasi.

2. Peneliti membandingkan dua skenario yang mana metode evaluasi yang kami gunakan adalah ROUGE. Skenario pertama dievaluasi menggunakan ROUGE-1 dan skenario kedua menggunakan ROUGE-2. ROUGE-1 mengukur ketepatan hasil ringkasan yang dihasilkan berdasarkan *unigram* (satu kata) dan ROUGE-2 mengukur berdasarkan *bigram* (pasangan dua kata). Skenario pertama memiliki hasil yang lebih baik daripada skenario kedua dengan ROUGE-1 0.5371 dan ROUGE-2 0.3757. Hal tersebut membuktikan bahwa ringkasan yang dihasilkan memiliki hasil yang bagus pada kata individual bukan kata berpasangan sehingga tingkat keterbacaan kalimat masih sangat rendah.

   Dibandingkan dengan penelitian-penelitian sejenis hasil evaluasi peneliti menunjukkan hasil yang bagus khususnya pada kategori *abstractive text summarization* berbahasa Indonesia. Meskipun masih memiliki hasil yang

sangat rendah yaitu masih di bawah 0.6. hal ini membuktikan bahwa penelitian ini masih bisa dikembangkan lagi untuk mendapatkan hasil yang lebih optimal, dan penelitian ini bisa juga dijadikan sebagai metric untuk dibandingkan dengan penelitian-penelitian selanjutnya.

5.2 Saran

Adapun saran untuk penelitian yang akan datang adalah sebagai berikut:

1. Melakukan percobaan mengganti *cell* RNN dari LSTM ke GRU karena berdasarkan referensi dari penelitian-penelitian terbaru GRU dapat menghasilkan output yang lebih baik daripada LSTM.
2. Mencoba menaikkan dimensi vektorisasi dari yang sebelumnya hanya 50 dimensi. Karena walaupun berbanding terbalik dengan kecepatan komputasi berdasarkan beberapa referensi dengan menambah dimensi vektor dapat meningkatkan keterkaitan antar kata yang dapat meningkatkan kualitas output ringkasan.
3. Mencoba melakukan percobaan dengan berbagai jenis kategori teks, kerena pada penelitian ini peneliti mengambil subjek yaitu artikel berita sehingga ringkasan belum tentu memiliki hasil yang baik pada teks bertipe artikel penelitian, draft hukum, laporan bisnis, maupun artikel-artikel lainnya.

DAFTAR PUSTAKA

Adelia, R., Suyanto, S., & Wisesty, U. N. (2019). Indonesian abstractive text summarization using bidirectional gated recurrent unit. *Procedia Computer Science*, *157*, 581–588. https://doi.org/10.1016/j.procs.2019.09.017

Alami, N., Meknassi, M., & En-nahnahi, N. (2019). Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning. *Expert Systems with Applications*, *123*, 195–211. https://doi.org/10.1016/j.eswa.2019.01.037

Annisa, D., & Khodra, M. L. (2017). Query-based summarization for Indonesian news articles. *Proceedings - 2017 International Conference on Advanced Informatics: Concepts, Theory and Applications, ICAICTA 2017*. https://doi.org/10.1109/ICAICTA.2017.8090959

Bahdanau, D., Cho, K., & Bengio, Y. (2015). *NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE*.

Cai, Z., Lin, N., Ma, C., & Jiang, S. (2019). Indonesian automatic text summarization based on a new clustering method in sentence level. *ACM International Conference Proceeding Series*, 30–35. https://doi.org/10.1145/3341620.3341626

Garmastewira, G. (2020). SUMMARIZING INDONESIAN NEWS ARTICLES USING GRAPH CONVOLUTIONAL NETWORK. *Journal of Information and Communication Technology*, *18*(3), 345–365.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. In *MIT Press*. http://files.sig2d.org/sig2d14.pdf#page=5

Hochreiter, S., & Schmidhuber, J. (1997). LSTM CAN SOLVE HARD LONG TIME LAG PROBLEMS. *Advances in Neural Information Processing Systems*, 473–479.

Hutama, R. B., Barakbah, A. R., & Helen, A. (2017). Indonesian news auto summarization in infrastructure development topic using 5W+1H consideration. *International Electronics Symposium on Knowledge Creation and Intelligent Computing, IES-KCIC 2017*, 258–264. https://doi.org/10.1109/KCIC.2017.8228596

Indurkhya, N., & Damerau, F. J. (2010). Handbook of Natural Language Processing, Second Edition. In *HLT-NAACL 2006 - Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings of the Main Conference*. CRC Press. https://doi.org/10.3115/1220835.1220877

Ivanedra, K., & Mustikasari, M. (2019). *Implementasi Metode Recurrent Neural Network Pada Text Summarization Dengan Teknik Abstraktif the Implementation of Text Summarization With Abstractive Techniques Using Recurrent Neural Network Method*. *6*(4), 377–382. https://doi.org/10.25126/jtiik.201961067

Jivani, A. G. (2012). A Comparative Study of Stemming Algorithms. *J. Comp. Tech. Appl.*, *2*(6), 1930–1938. https://doi.org/10.1093/oxfordhb/9780195396430.013.0038

Joshi, A., Fidalgo, E., Alegre, E., & Fernández-Robles, L. (2019a). SummCoder: An unsupervised framework for extractive text summarization based on deep auto-encoders. *Expert Systems with Applications*, *129*, 200–215. https://doi.org/10.1016/j.eswa.2019.03.045

Joshi, A., Fidalgo, E., Alegre, E., & Fernández-Robles, L. (2019b). SummCoder: An unsupervised framework for extractive text summarization based on deep auto-encoders. *Expert Systems with Applications*, *129*, 200–215. https://doi.org/10.1016/j.eswa.2019.03.045

Kostadinov, simeon. (2019). *Understanding Encoder-Decoder Sequence to Sequence Model*. Towards Data Science.

Kurniawan, K., & Louvan, S. (2018). I NDO S UM : A New Benchmark Dataset for Indonesian Text Summarization. *2018 International Conference on Asian Language Processing (IALP)*, 215–220.

Kurniawan, K., & Louvan, S. (2019). IndoSum: A New Benchmark Dataset for Indonesian Text Summarization. *Proceedings of the 2018 International Conference on Asian Language Processing, IALP 2018*, 215–220. https://doi.org/10.1109/IALP.2018.8629109

Lin, C.-Y. (2004). Looking for a Few Good Metrics: ROUGE and its Evaluation. *NTCIR Workshop*, *June*, 1–8.

Méndez, J. R., Iglesias, E. L., Fdez-Riverola, F., Díaz, F., & Corchado, J. M. (2006). Tokenising, stemming and stopword removal on anti-spam filtering domain. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4177 LNAI*, 449–458. https://doi.org/10.1007/11881216_47

Moreno, L., & Marcus, A. (2017). Automatic software summarization: The state of the art. *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, 511–512. https://doi.org/10.1109/ICSE-C.2017.169

Nallapati, R., Zhou, B., Santos, C. dos, & Xiang, B. (2016). Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond Ramesh. *The*

SIGNLL Conference on Computational Natural Language Learning (CoNLL).

Puspitasari, D. Z. E., Barakbah, A. R., & Winarno, I. (2012). Automatic Representative News Generation using On-Line Clustering. *EMITTER International Journal of Engineering Technology*, *1*(1). https://doi.org/10.24003/emitter.v1i1.11

Rahimi, S. R., Mozhdehi, A. T., & Abdolahi, M. (2018). An overview on extractive text summarization. *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation, KBEI 2017*, *2018-Janua*, 0054–0062. https://doi.org/10.1109/KBEI.2017.8324874

Ran, X., Shan, Z., Fang, Y., & Lin, C. (2019). An LSTM-based method with attention mechanism for travel time prediction. Sensors, 19(4), 861.

Ravi, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B., & Yang, G. Z. (2017). Deep Learning for Health Informatics. *IEEE Journal of Biomedical and Health Informatics*, *21*(1), 4–21. https://doi.org/10.1109/JBHI.2016.2636665

Reztaputra, R., & Khodra, M. L. (2017). Sentence structure-based summarization for Indonesian news articles. *Proceedings - 2017 International Conference on Advanced Informatics: Concepts, Theory and Applications, ICAICTA 2017*, 0–5. https://doi.org/10.1109/ICAICTA.2017.8090983

Sharma, O. (2019). Deep Challenges Associated with Deep Learning. *Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel Computing: Trends, Prespectives and Prospects, COMITCon 2019*, 72–75. https://doi.org/10.1109/COMITCon.2019.8862453

Shirwandkar, N. S., & Kulkarni, S. (2018). Extractive Text Summarization using Neural Networks. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. http://arxiv.org/abs/1802.10137

Sintoris, K., & Vergidis, K. (2017). Extracting business process models using natural language processing (NLP) techniques. *Proceedings - 2017 IEEE 19th Conference on Business Informatics, CBI 2017*, *1*, 135–139. https://doi.org/10.1109/CBI.2017.41

Sorin, V., Barash, Y., Konen, E., & Klang, E. (2020). Deep Learning for Natural Language Processing in Radiology—Fundamentals and a Systematic Review. *Journal of the American College of Radiology*, *17*(5), 639–648. https://doi.org/10.1016/j.jacr.2019.12.026

Steinberger, J., & Jezek, K. (2004). Using latent semantic analysis in text summarization and summary evaluation. *Proc. ISIM'04*, 93–100. https://doi.org/10.1177/0165551511408848

Sun, S., Luo, C., & Chen, J. (2017). A review of natural language processing techniques for opinion mining systems. *Information Fusion*, *36*, 10–25. https://doi.org/10.1016/j.inffus.2016.10.004

Tsuchiya, G. (1971). Postmortem Angiographic Studies on the Intercoronary Arterial Anastomoses.: Report I. Studies on Intercoronary Arterial Anastomoses in Adult Human Hearts and the Influence on the Anastomoses of Strictures of the Coronary Arteries. *Japanese Circulation Journal*, *34*(12), 1213–1220. https://doi.org/10.1253/jcj.34.1213

Vanderwende, L., Suzuki, H., Brockett, C., & Nenkova, A. (2007). Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing and Management*, *43*(6), 1606–1618. https://doi.org/10.1016/j.ipm.2007.01.023

Zajic, D. M., Dorr, B. J., & Lin, J. (2008). Single-document and multi-document summarization techniques for email threads using sentence compression. *Information Processing and Management*, *44*(4), 1600–1610. https://doi.org/10.1016/j.ipm.2007.09.007

LAMPIRAN

Lampiran 1. *Source code* model *sequence-to-sequence*

```
#models
import tensorflow as tf
from tensorflow.contrib import rnn

class Model(object):
   def __init__(self, reversed_dict, article_max_len, summary_max_len,forward_only,keep_prob,e
mbedding_size,num_hidden,num_layers,learning_rate,beam_width,glove,path):
      self.vocabulary_size = len(reversed_dict)
      self.embedding_size = embedding_size
      self.num_hidden = num_hidden
      self.num_layers = num_layers
      self.learning_rate = learning_rate
      self.beam_width = beam_width
      if not forward_only:
         self.keep_prob = keep_prob
      else:
         self.keep_prob = 1.0
      self.cell = tf.nn.rnn_cell.BasicLSTMCell
      with tf.variable_scope("decoder/projection"):
         self.projection_layer = tf.layers.Dense(self.vocabulary_size, use_bias=False)
      self.batch_size = tf.placeholder(tf.int32, (), name="batch_size")
      self.X = tf.placeholder(tf.int32, [None, article_max_len])
      self.X_len = tf.placeholder(tf.int32, [None])
      self.decoder_input = tf.placeholder(tf.int32, [None, summary_max_len])
      self.decoder_len = tf.placeholder(tf.int32, [None])
      self.decoder_target = tf.placeholder(tf.int32, [None, summary_max_len])
      self.global_step = tf.Variable(0, trainable=False,name='step')

      with tf.name_scope("embedding"):
         if not forward_only and glove:
            init_embeddings = tf.constant(get_init_embedding(path,reversed_dict, self.embedding_siz
e), dtype=tf.float32)
         else:
            init_embeddings = tf.random_uniform([self.vocabulary_size, self.embedding_size], -
1.0, 1.0)
         self.embeddings = tf.get_variable("embeddings", initializer=init_embeddings)
         self.encoder_emb_inp = tf.transpose(tf.nn.embedding_lookup(self.embeddings, self.X), per
m=[1, 0, 2])
         self.decoder_emb_inp = tf.transpose(tf.nn.embedding_lookup(self.embeddings, self.decoder
_input), perm=[1, 0, 2])

      with tf.name_scope("encoder"):
         fw_cells = [self.cell(self.num_hidden) for _ in range(self.num_layers)]
         bw_cells = [self.cell(self.num_hidden) for _ in range(self.num_layers)]
         fw_cells = [rnn.DropoutWrapper(cell) for cell in fw_cells]
         bw_cells = [rnn.DropoutWrapper(cell) for cell in bw_cells]

         encoder_outputs, encoder_state_fw, encoder_state_bw = tf.contrib.rnn.stack_bidirectional_d
```

```
ynamic_rnn(
            fw_cells, bw_cells, self.encoder_emb_inp,
            sequence_length=self.X_len, time_major=True, dtype=tf.float32)
        self.encoder_output = tf.concat(encoder_outputs, 2)
        encoder_state_c = tf.concat((encoder_state_fw[0].c, encoder_state_bw[0].c), 1)
        encoder_state_h = tf.concat((encoder_state_fw[0].h, encoder_state_bw[0].h), 1)
        self.encoder_state = rnn.LSTMStateTuple(c=encoder_state_c, h=encoder_state_h)

    with tf.name_scope("decoder"), tf.variable_scope("decoder") as decoder_scope:
        decoder_cell = self.cell(self.num_hidden * 2)
        if not forward_only:
            attention_states = tf.transpose(self.encoder_output, [1, 0, 2])
            attention_mechanism = tf.contrib.seq2seq.BahdanauAttention(
                self.num_hidden * 2, attention_states, memory_sequence_length=self.X_len, normalize
=True)
            decoder_cell = tf.contrib.seq2seq.AttentionWrapper(decoder_cell, attention_mechanism,
                                               attention_layer_size=self.num_hidden * 2)
            initial_state = decoder_cell.zero_state(dtype=tf.float32, batch_size=self.batch_size)
            initial_state = initial_state.clone(cell_state=self.encoder_state)
            helper = tf.contrib.seq2seq.TrainingHelper(self.decoder_emb_inp, self.decoder_len, time_
major=True)
            decoder = tf.contrib.seq2seq.BasicDecoder(decoder_cell, helper, initial_state)
            outputs, _, _ = tf.contrib.seq2seq.dynamic_decode(decoder, output_time_major=True, sco
pe=decoder_scope)
            self.decoder_output = outputs.rnn_output
            self.logits = tf.transpose(
                self.projection_layer(self.decoder_output), perm=[1, 0, 2])
            self.logits_reshape = tf.concat(
                [self.logits, tf.zeros([self.batch_size, summary_max_len - tf.shape(self.logits)[1], self.v
ocabulary_size])], axis=1)
        else:
            tiled_encoder_output = tf.contrib.seq2seq.tile_batch(
                tf.transpose(self.encoder_output, perm=[1, 0, 2]), multiplier=self.beam_width)
            tiled_encoder_final_state = tf.contrib.seq2seq.tile_batch(self.encoder_state, multiplier=sel
f.beam_width)
            tiled_seq_len = tf.contrib.seq2seq.tile_batch(self.X_len, multiplier=self.beam_width)
            attention_mechanism = tf.contrib.seq2seq.BahdanauAttention(
                self.num_hidden * 2, tiled_encoder_output, memory_sequence_length=tiled_seq_len, n
ormalize=True)
            decoder_cell = tf.contrib.seq2seq.AttentionWrapper(decoder_cell, attention_mechanism,
                                               attention_layer_size=self.num_hidden * 2)
            initial_state = decoder_cell.zero_state(dtype=tf.float32, batch_size=self.batch_size * self.
beam_width)
            initial_state = initial_state.clone(cell_state=tiled_encoder_final_state)
            decoder = tf.contrib.seq2seq.BeamSearchDecoder(
                cell=decoder_cell,
                embedding=self.embeddings,
                start_tokens=tf.fill([self.batch_size], tf.constant(2)),
                end_token=tf.constant(3),
                initial_state=initial_state,
                beam_width=self.beam_width,
```

```
                output_layer=self.projection_layer
            )
            outputs, _, _ = tf.contrib.seq2seq.dynamic_decode(
                decoder, output_time_major=True, maximum_iterations=summary_max_len, scope=de
coder_scope)
            self.prediction = tf.transpose(outputs.predicted_ids, perm=[1, 2, 0])
        with tf.name_scope("loss"):
            if not forward_only:
                crossent = tf.nn.sparse_softmax_cross_entropy_with_logits(
                    logits=self.logits_reshape, labels=self.decoder_target)
                weights = tf.sequence_mask(self.decoder_len, summary_max_len, dtype=tf.float32)
                self.loss = tf.reduce_sum(crossent * weights / tf.to_float(self.batch_size))
                params = tf.trainable_variables()
                gradients = tf.gradients(self.loss, params)
                clipped_gradients, _ = tf.clip_by_global_norm(gradients, 5.0)
                optimizer = tf.train.AdamOptimizer(self.learning_rate)
                self.update = optimizer.apply_gradients(zip(clipped_gradients, params), global_step=self.
global_step)
```