**PERINGKASAN TEKS BERITA OTOMATIS DENGAN GRAPH CONVOLUTIONAL NETWORK**

**Laporan Tugas Akhir I**

**Disusun sebagai syarat kelulusan mata kuliah**

**IF4091/Tugas Akhir I dan Seminar**

**Oleh**

**GARMASTEWIRA**

**NIM : 13514068**



**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO & INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**JANUARI 2018**

**PERINGKASAN ARTIKEL BERITA ONLINE OTOMATIS DENGAN GRAPH CONVOLUTIONAL NETWORK**

**Laporan Tugas Akhir I**

**Oleh**

**GARMASTEWIRA**

**NIM : 13514068**

**Program Studi Teknik Informatika**

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Bandung, 8 Januari 2018

Mengetahui,

Pembimbing,

Dr. Masayu Leylia Khodra, S.T. M.T.

NIP. 19760429 200812 2 001

DAFTAR ISI

[DAFTAR ISI iii](#_Toc406869913)

[DAFTAR LAMPIRAN](#_Toc406869914) v

[DAFTAR GAMBAR](#_Toc406869915) vi

[DAFTAR TABEL](#_Toc406869916) vii

[BAB I PENDAHULUAN 1](#_Toc502433472)

[I.1 Latar Belakang 1](#_Toc502433473)

[I.2 Rumusan Masalah 4](#_Toc502433474)

[I.3 Tujuan 4](#_Toc502433475)

[I.4 Batasan Masalah 5](#_Toc502433476)

[I.5 Metodologi 5](#_Toc502433477)

[I.6 Jadwal Pelaksanaan Tugas Akhir 6](#_Toc502433478)

[BAB II STUDI LITERATUR 8](#_Toc502433479)

[II.1 Peringkasan Teks Otomatis 8](#_Toc502433480)

[II.2 Word Embedding 11](#_Toc502433481)

[II.3 Recurrent Neural Network dan Gated Recurrent Unit 13](#_Toc502433482)

[II.4 Relasi *Discourse* 15](#_Toc502433483)

[II.5 Graph Convolutional Neural Network 17](#_Toc502433484)

[II.6 Penelitian Terkait 18](#_Toc502433485)

[II.6.1 Improving the Estimation of Word Importance for News Multi-Document Summarization 18](#_Toc502433486)

[II.6.2 Peringkasan Otomatias Kumpulen Artikel Berita *Online* Berbasiskan Struktur Kalimat 19](#_Toc502433487)

[II.6.3 Towards Coherent Multi-Document Summarization 20](#_Toc502433488)

[II.6.4 Graph-based Neural Multi-Document Summarization 22](#_Toc502433489)

[BAB III ANALISIS DAN RANCANGAN SISTEM PERINGKASAN OTOMATIS 26](#_Toc502433490)

[III.1 Analisis Masalah 26](#_Toc502433491)

[III.2 Analisis Solusi 27](#_Toc502433492)

[III.2.1 Analisis Pembelajaran Fitur dengan *Deep Learning* 27](#_Toc502433493)

[III.2.2 Analisis Representasi Graf *Discourse* dan Pembelajaran dengan GCN 28](#_Toc502433494)

[III.3 Rancangan Sistem 29](#_Toc502433495)

[III.3.1 Komponen Praproses 29](#_Toc502433496)

[III.3.2 Komponen GRU Kalimat 30](#_Toc502433497)

[III.3.3 Komponen Pembentukan Graf 32](#_Toc502433498)

[III.3.4 Komponen Graph Convolutional Network 35](#_Toc502433499)

[III.3.5 Komponen GRU Dokumen 37](#_Toc502433500)

[III.3.6 Komponen Penilaian Kalimat 38](#_Toc502433501)

[III.3.7 Komponen Pemilihan Kalimat 39](#_Toc502433502)

[III.4 Pembelajaran Komponen Deep Learning 40](#_Toc502433503)

[III.5 Implementasi Sistem 41](#_Toc502433504)

[III.6 Sistem *Baseline* 41](#_Toc502433505)

DAFTAR LAMPIRAN

[Lampiran A. Contoh Judul Lampiran 45](#_Toc406869964)

[A.1 Contoh Judul Anak Lampiran 45](#_Toc406869965)

DAFTAR GAMBAR

[Gambar I.1. Diagram Gantt Jadwal Kegiatan Tugas Akhir 6](#_Toc502433347)

[Gambar II.1. Contoh Visualisasi Word Embedding Bahasa Indonesia dengan Model FastText 13](#_Toc502433348)

[Gambar II.2. Topologi RNN Sederhana 13](#_Toc502433349)

[Gambar II.3. Perbedaan Sel RNN (Kiri) dan Sel GRU (Kanan) 14](#_Toc502433350)

[Gambar II.4. Contoh Graf *Discourse* (Christensen, 2013) 17](#_Toc502433351)

[Gambar II.5. Arsitektur Peringkasan Reztaputra & Khodra (2017) 20](#_Toc502433352)

[Gambar II.6. Arsitektur Peringkasan Multi-Dokumen Berbasis Graf Neural (Yasunaga, 2017) 23](#_Toc502433353)

[Gambar III.1. Arsitektur Sistem Peringkasan Otomatis 29](#_Toc502433354)

[Gambar III.2. Ilustrasi Proses Pembentukan *Sentence Embedding* Menggunakan GRU 31](#_Toc502433355)

[Gambar III.3. Ilustrasi GCN dengan Satu Lapisan 36](#_Toc502433356)

[Gambar III.4. Ilustrasi GRU Dokumen dan Perhitungan *Cluster Embedding* 37](#_Toc502433357)

DAFTAR TABEL

[Tabel I.1. Tabel Jadwal Kegiatan Tugas Akhir 6](#_Toc502435574)

[Tabel I.2. Tabel *Milestone* Tugas Akhir 7](#_Toc502435575)

[Tabel I.3. Tabel *Deliverables* Tugas Akhir 7](#_Toc502435576)

[Tabel II.1. Tingkat Fitur Peringkasan Otomatis Beserta Contoh 8](#_Toc502435577)

[Tabel II.2. Contoh Perhitungan ROUGE 10](#_Toc502435578)

[Tabel II.3. Contoh Komponen N-Gram Karakter 12](#_Toc502435579)

[Tabel II.4. Contoh Kalimat yang Memiliki Hubungan *Discourse* 16](#_Toc502435580)

[Tabel II.5. Hasil Evaluasi Peringkasan Hong & Nenkova (2014) 19](#_Toc502435581)

[Tabel II.6. Hasil Evaluasi Peringkasan Reztaputra & Khodra (2017) 20](#_Toc502435582)

[Tabel II.7. Hasil Evaluasi Peringkasan Berbasis Graf Neural pada Data DUC 2004 (Yasunaga, 2017) 25](#_Toc502435583)

[Tabel III.1. Pemetaan Masalah dan Rancangan Solusi 27](#_Toc502435584)

[Tabel III.2. Contoh Hasil Praproses 30](#_Toc502435585)

[Tabel III.3. Contoh Pembangunan Sisi Graf ADG 33](#_Toc502435586)

[Tabel III.4. Fitur Pembelajaran Regresi Linier Skor Personalisasi 35](#_Toc502435587)

# PENDAHULUAN

## Latar Belakang

Portal berita *online* merupakan konsumsi utama masyarakat untuk mengetahui kejadian-kejadian terbaru. Portal berita *online* lebih diminati dibandingkan dengan media berita konvensional seiring dengan meningkatnya penggunaan media sosial. Survei yang dilakukan oleh Lu & Holcomb (2016) menunjukkan bahwa jumlah orang Amerika Serikat yang menggunakan media portal berita *online* untuk membaca berita menduduki peringkat dua setelah media cetak, dengan jumlah 38%.

Umumnya, suatu topik berita dilaporkan beberapa kali oleh banyak portal berita *online*. Setiap artikel suatu topik berita memiliki informasi yang redundan yang dimuat oleh artikel lainnya, namun dapat juga memiliki informasi unik. Oleh karena itu, diperlukan suatu metode peringkasan yang dapat merangkum informasi seluruh artikel topik berita menjadi suatu ringkasan ideal. Peringkasan multi-dokumen ideal menghasilkan ringkasan yang mencakup informasi utama yang terdapat pada seluruh dokumen serta informasi unik dari setiap dokumen (Goldstein, 2000).

Berdasarkan pendekatan metode pembangunan ringkasan, peringkasan terbagi menjadi dua, yaitu peringkasan ekstraktif dan peringkasan abstraktif. Peringkasan ekstraktif memilih kalimat-kalimat penting pada dokumen tanpa mengubah kalimat, sedangkan peringkasan abstraktif mengambil informasi penting pada suatu dokumen yang nantinya dituliskan dalam kalimat-kalimat baru (Jurafsky & Martin, 2007). Ringkasan yang ideal pada dasarnya merupakan peringkasan abstraktif karena peringkasan yang dilakukan oleh manusia juga menggunakan pendekatan abstraktif. Namun, penelitian peringkasan otomatis dengan pendekatan ekstraktif maupun abstraktif memberikan perbedaan kinerja yang tipis, dan peringkasan abstraktif seringkali membutuhkan waktu pemrosesan lebih lama serta *tools* dan sumber daya, misalnya aturan parafrase informsai, yang kualitasnya buruk pada bahasa kurang populer (Galanis dkk., 2012). Oleh karena itu, penelitian ini akan fokus ke peringkasan dengan pendekatan ekstraktif.

Pada bahasa Indonesia, terdapat penelitian peringkasan multi-dokumen dengan pendekatan ekstraktif maupun abstraktif. Untuk pendekatan abstraktif, Christie & Khodra (2016) membangun sistem peringkasan abstraktif dengan memanfaatkan penggabungan kalimat dengan graf kata dan seleksi informasi ringkasan menggunakan ILP. Reztaputra & Khodra (2017) membangun sistem peringkasan yang menafaatkan ekstraksi informasi subjek, predikat, objek, dan keterangan (SPOK) dari suatu artikel dengan memanfaatkan pohon kebergantungan, kemudian melakukan pengelompokan dan penggabungan SPOK. Hasil ekstraksi informasi SPOK kemudian disusun menjadi ringkasan. Untuk pendekatan ekstraktif, Annisa & Khodra (2017) membangun sistem peringkasan berbasis *query*, di mana selain mempertimbangkan kepentingan kalimat, pemilihan kalimat ringkasan juga mempertimbangkan relevansi kalimat dengan *query*.

Pada peringkasan ekstraktif, tahapan yang paling penting adalah menentukan nilai kepentingan kalimat. Kupiec (1998) menentukan nilai kepentingan kalimat dengan menentukan fitur-fitur klasik pada suatu kalimat (posisi kalimat, keberadaan kata kunci, dll.), kemudian melakukan pembelajaran terhadap fitur-fitur tersebut dengan algoritma naïve Bayes. Kalimat yang dijadikan ringkasan adalah kalimat dengan probabilitas keluaran naïve Bayes di atas ambang batas. Tentunya, dengan fitur klasik saja, kualitas ringkasan yang dihasilkan belum cukup baik. Hong & Nenkova (2014) kemudian membangun sistem peringkasan ekstraktif berdasarkan kepentingan kata. Sistem peringkasan ini menghasilkan nilai ROUGE-2 dengan nilai 0.0975 pada korpus Document Understanding Conference (DUC) 2004, namun fitur yang digunakan pada pembelajaran jumlahnya mencapai 9.000 fitur. Kebanyakan fitur merupakan fitur rekayasa manusia yang cukup kompleks, seperti nilai hasil algoritma PageRank terhadap graf keterhubungan kata berdasarkan pohon kebergantungan dan model bahasa yang dibuat berdasarkan korpus selain data latih, yaitu korpus berita New York Times.

Saat ini, banyak penelitian peringkasan otomatis yang mengaplikasikan *deep learning* dengan fitur tingkat rendah dari masukan. Penelitian peringkasan dokumen tunggal menggunakan *deep learning* cukup banyak. Untuk peringkasan abstraktif, Rush (2015) memanfaatkan arsitektur *recurrent neural network* (RNN) dengan variasi *sequence-to-sequence* untuk membangun ringkasan secara abstraktif. Untuk peringkasan ekstraktif, Cheng & Lapata (2012) memanfaatkan arsitektur *convolutional neural network* (CNN) dan *long short-term memory* (LSTM) untuk menentukan nilai kepentingan dari suatu kalimat. Pada peringkasan multi-dokumen, penelitian yang menggunakan *deep learning* masih sedikit. Cao (2016) melakukan peringkasan ekstraktif multi-dokumen berbasis *query* menggunakan *deep learning* dengan memanfaatkan CNN. Lapian konvolusi menerima masukan kalimat artikel setopik dan menghasilkan vektor *sentence embedding* untuk setiap kalimat dan vektor *query embedding* dari kalimat *query*. Lapisan *pooling* kemudian menghasilkan vektor *document embedding* dari kombinasi seluruh *sentence embedding* dengan mempertimbangkan relevansi setiap *sentece embedding* terhadap *query embedding*. Nilai kepentingan kalimat dihitung dari *cosine similarity* antara *sentence embedding* dari kalimat dengan *document embedding*.

Salah satu hal yang sering diabaikan ketika menghasilkan ringkasan adalah keterhubungan antar kalimat. Meskipun kinerja yang dihasilkan sangat baik, Hong & Nenkova (2014) hanya memperhitungkan fitur leksikal dari kata setiap kalimat. Pada peringkasan multi-dokumen ekstraktif dengan *deep learning,* Cao (2016) menghitung nilai *sentence embedding* setiap kalimat secara independen. Sistem peringkasan multi-dokumen yang dibangun Christensen (2013) menunjukkan bahwa keterhubungan antar kalimat perlu diperhitungkan untuk menghasilkan ringkasan yang koheren.

Yasunaga (2017) memanfaatkan *graph convolutional network* (GCN) untuk memperhitungkan hubungan setiap kalimat seluruh dokumen pada suatu topik (Kipf & Welling, 2017). Untuk memperoleh representasi vektor suatu kalimat, Yasunaga menggunakan arsitektur RNN untuk menghasilkan *sentence embedding* dari sekuens token kalimat. Seluruh *sentence embedding* kemudian dimodelkan sebagai sebuah graf, di mana simpul merupakan *sentence embedding*, dan bobot busur antar simpul menyatakan keterhubungan antar kalimat. Hasil propagasi GCN menghasilkan *sentence embedding* yang memperhitungkan keterhubungan antar kalimat. Nilai kepentingan kalimat kemudian diperhitungkan berdasarkan *sentence embedding* tersebut. Hasil evaluasi ROUGE-1 dan ROUGE-2 penelitian Yasunaga (2017) menunjukkan kinerja yang lebih baik dibandingkan dengan penelitian Cao (2016).

Tugas akhir ini akan mengimplementasikan peringkasan otomatis yang memperhatikan keterhubungan antar masukan dengan menggunakan arsitektur GCN. Dengan arsitektur *deep learning* GRU dan GCN, sistem peringkasan dapat memperoleh nilai kepentingan setiap kalimat berdasarkan fitur tingkat rendah hasil pembelajaran arsitektur *deep learning* serta mempertimbangkan keterhubungan antar kalimat agar menghasilkan ringkasan yang koheren*.* Tugas akhir ini diharapkan dapat membangun sistem peringkasan otomatis yang menghasilkan ringkasan dengan kualitas baik.

## Rumusan Masalah

Permasalahan utama dari sistem peringkasan multi-dokumen dengan pendekatan ekstraktif adalah tahap pemilihan kalimat penting. Untuk menghasilkan ringkasan yang baik, pemilihan kalimat penting harus melihat berbagai aspek, salah satunya adalah keterhubungan antar kalimat. Rumusan masalah dari tugas akhir ini adalah bagaimana cara membangun sistem peringkasan multi-dokumen otomatis untuk bahasa Indonesia dengan memanfaatkan *graph convolutional network* dan bagaimana perbandingan hasil peringkasan sistem dengan ringkasan acuan buatan manusia.

## Tujuan

Tujuan tugas akhir ini adalah untuk membangun sistem peringkasan multi-dokumen otomatis untuk bahasa Indonesia dengan *graph convolutional network* yang menghasilkan ringkasan yang mirip dengan ringkasan buatan manusia serta menganalisis karateristik dokumen bahasa Indonesia berdasarkan hasil pembelajaran dan keluaran sistem.

## Batasan Masalah

Berikut adalah batasan masalah dari tugas akhir ini.

1. Artikel dan ringkasan yang digunakan ditulis dalam bahasa Indonesia.
2. Ringkasan yang dihasilkan memiliki batasan jumlah kata, yaitu 100 kata dan 200 kata.

## Metodologi

Berikut adalah metodologi untuk pengerjaan tugas akhir ini.

1. Pengumpulan Data

Data berupa sekumpulan artikel bahasa Indonesia beserta ringkasannya akan dikumpulkan terlebih dahulu. Data yang dikumpulkan akan menjadi data latih untuk pembangunan model sistem peringkasan.

1. Analisis Persoalan dan Rancangan Solusi

Tahap ini mencakup analisis terhadap permasalahan yang akan diselesaikan serta rancangan dari sistem solusi yang akan dibangun berdasarkan analisis permasalahan.

1. Eksperimen dan Pengujian

Tahap eksperimen akan melatih model sistem peringkasan otomatis yang dirancang sesuai hasil analisis solusi. Model hasil pelatihan kemudian diuji dengan data uji yang digunakan pada penelitian peringkasan otomatis bahasa Indonesia lainnya.

1. Evaluasi dan Penarikan Simpulan

Kinerja sistem peringkasan terhadap data uji akan dievaluasi terhadap kinerja sistem peringkasan penelitian lain. Hasil evaluasi dijadikan dasar untuk penarikan simpulan.

## Jadwal Pelaksanaan Tugas Akhir

Gambar I.1. dan Tabel I.1. memperlihatkan jadwal kegiatan tugas akhir untuk setiap minggu. Tabel I.2. dan Tabel I.3 memperlihatkan *milestone* dan *deliverables* dari tugas akhir

Gambar I.1. Diagram Gantt Jadwal Kegiatan Tugas Akhir

Tabel I.1. Tabel Jadwal Kegiatan Tugas Akhir

| No | Tugas | Tanggal Mulai | Tanggal Akhir | Durasi (Hari) |
| --- | --- | --- | --- | --- |
| 1 | Eksplorasi topik | 21-08-2017 | 03-09-2017 | 13 |
| 2 | Pengajuan topik dan pembimbing | 04-09-2017 | 10-09-2017 | 6 |
| 3 | Studi literatur | 11-09-2017 | 29-10-2017 | 48 |
| 4 | Menulis bab I | 30-10-2017 | 26-11-2017 | 27 |
| 5 | Perancangan dan analisis solusi | 27-11-2017 | 07-01-2018 | 41 |
| 6 | Seminar TA I | 08-01-2018 | 21-01-2018 | 13 |
| 7 | Implementasi | 22-01-2018 | 25-02-2018 | 34 |
| 8 | Eksperimen dan pengujian | 26-02-2018 | 01-04-2018 | 34 |
| 9 | Evaluasi | 02-04-2018 | 08-04-2018 | 6 |
| 10 | Finalisasi laporan TA | 09-04-2018 | 06-05-2018 | 27 |
| 11 | Seminar TA II | 07-05-2018 | 20-05-2018 | 13 |
| 11 | Sidang akhir TA | 21-05-2018 | 27-05-2018 | 6 |

Tabel I.2. Tabel *Milestone* Tugas Akhir

| Nomor | Tanggal | Kegiatan |
| --- | --- | --- |
| 1 | 10/09/2017 | Mengajukan topik beserta dosen pembimbing terkait |
| 2 | 29/10/2017 | Mengajukan proposal bab II |
| 3 | 26/11/2017 | Mengajukan proposal bab I |
| 4 | 21/01/2018 | Seminar TA I |
| 5 | 01/04/2018 | Menyelesaikan eksperimen dan pengujian |
| 6 | 06/05/2018 | Mengumpulkan laporan TA |
| 7 | 20/05/2018 | Seminar TA II |
| 8 | 27/05/2018 | Sidang Akhir TA |

Tabel I.3. Tabel *Deliverables* Tugas Akhir

| Nomor | Deliverable | Keterangan |
| --- | --- | --- |
| 1 | Presentasi seminar TA I | - |
| 2 | Presentasi seminar TA II | - |
| 3 | Dokumen laporan TA | - |
| 4 | Dataset artikel dan ringkasan hasil pengumpulan data | Dataset berita *online* yang membahas 50 topik, di mana masing-masing topik terdiri atas 10 artikel |
| 5 | Hasil peringkasan sistem | - |

# STUDI LITERATUR

## Peringkasan Teks Otomatis

Peringkasan teks otomatis merupakan proses untuk menyaring informasi penting pada sebuah teks untuk menghasilkan versi yang lebih singkat secara otomatis (Mani & Maybury, 1999). Berdasarkan pendekatan metode pembangunan ringkasan, peringkasan otomatis terbagi atas dua pendekatan, yaitu pendekatan ekstraktif dan abstraktif (Hahn & Mani, 2000). Pada pendekatan ekstraktif, peringkasan memilih kalimat-kalimat dengan informasi penting. Kalimat-kalimat tersebut kemudian dikonkatenasi menjadi sebuah ringkasan. Pada pendekatan abstraktif, peringkasan mengambil informasi penting pada teks, kemudian membuat kalimat-kalimat baru sesuai informasi penting yang diperoleh.

Berdasarkan jumlah dokumen yang akan diringkas, peringkasan teks otomatis dapat dibagi menjadi peringkasan dokumen tunggal dan peringkasan multi-dokumen. Menurut Jurafsky dan Martin (2007), pada peringkasan dokumen tunggal, masukan peringkasan adalah dokumen tunggal, dan tujuannya adalah untuk menentukan poin-poin penting pada dokumen. Pada peringkasan multi-dokumen, masukan peringkasan adalah sekumpulan dokumen yang menceritakan kejadian yang sama, dan tujuannya adalah untuk menghasilkan konten yang merepresentasikan seluruh kumpulan dokumen. Berdasarkan fitur yang diamati pada peringkasan otomatis, fitur teks dapat dibagi menjadi tiga tingkatan, yaitu fitur klasik, fitur entitas, dan fitur *discourse* (Mani & Maybury, 1999). Tabel II.1 memperlihatkan contoh fitur untuk masing-masing tingkatan.

Tabel II.1. Tingkat Fitur Peringkasan Otomatis Beserta Contoh

| Tingkat Fitur | Contoh Fitur |
| --- | --- |
| Klasik | Posisi kalimat pada teks, keberadaan kata dengan frekuensi besar, keberadaan kata spesifik domain, keberadaan kata penekanan, |
| Entitas | Kesamaan antar kalimat, hubungan kata berdasarkan thesaurus, hubungan sintaksis |
| *Discourse* | Struktur narasi teks |

Secara umum, tahapan pada peringkasan ekstraktif ada dua, yaitu tahap penilaian kalimat dan tahap pemilihan kalimat (Yasunaga, 2017). Pada tahap penilaian kalimat, kepentingan kalimat pada ringkasan dinilai berdasarkan fitur-fitur dari kalimat tersebut. Penilaian fitur umumnya menggunakan algoritma pembelajaran mesin, seperti *naive* Bayes (Kupiec dkk., 1995), *support vector regression* (Li, 2007) dan regresi logistik (Hong & Nenkova, 2014). Setelah setiap kalimat diberikan nilainya, tahap pemilihan kalimat kemudian memilih kalimat-kalimat yang tepat untuk menjadi bagian ringkasan. Pendekatan sederhana adalah dengan memilih kalimat berdasarkan skor terbaik secara *greedy* (Kupiec, 1995). Selain pendekatan *greedy*, terdapat beberapa optimasi pemilihan kalimat untuk meningkatkan kualitas ringkasan, seperti *graph centrality* (Erkan & Radev, 2004) dan *integer linear programming* (Galanis dkk., 2012).

Salah satu evaluasi peringkasan yang paling populer adalah *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE). ROUGE merupakan sekumpulan metode untuk mengevaluasi peringkasan otomatis dan translasi mesin. Metode ROUGE membandingkan hasil peringkasan otomatis terhadap sekumpulan ringkasan acuan yang ditulis oleh manusia (Lin, 2004). Contoh perhitungan ROUGE antar ringkasan hasil sistem dan ringkasan acuan dapat dilihat pada Tabel II.2. ROUGE terbagi menjadi tiga jenis evaluasi dengan rincian sebagai berikut.

1. ROUGE-N

ROUGE-N menghitung jumlah unit n-gram hasil peringkasan otomatis dan ringkasan sebenarnya yang cocok. Pada berbagai macam eksperimen peringkasan, ROUGE-N yang populer adalah ROUGE-1 (unigram) dan ROUGE-2 (bigram). Berikut adalah persamaan untuk ROUGE-N.

|  |  |  |
| --- | --- | --- |
|  |  | (II-1) |

Di mana merupakan himpunan ringkasan acuan, matchn adalah jumlah maksimum n-gram yang ada pada kandidat ringkasan dan ringkasan acuan, dan merupakan jumlah n-gram pada ringkasan acuan.

1. ROUGE-S

ROUGE-S menghitung jumlah unit skip-bigram hasil peringkasan otomatis dan ringkasan acuan yang cocok. Nilai ROUGE-S merupakan nilai skor F1 yang dihitung dengan persamaan (II-4) berdasarkan kombinasi persamaan (II-2) dan (II-3).

|  |  |  |
| --- | --- | --- |
|  |  | (II-2) |
|  |  | (II-3) |
|  |  | (II-4) |

Di mana merupakan ringkasan hasil sistem, Y merupakan ringkasan acuan, dan merupakan himpunan skip-bigram yang ada pada . menyatakan *recall*, sedangkan menyatakan *precision*.

1. ROUGE-SU

ROUGE-SU pada dasarnya merupakan gabungan dari ROUGE-1 dan ROUGE-S. ROUGE-SU menghitung jumlah unit unigram dan jumlah unit skip-bigram ringkasan hasil sistem dan ringkasan acuan yang cocok.

Tabel II.2. Contoh Perhitungan ROUGE

|  |  |  |
| --- | --- | --- |
|  | Ringkasan Hasil Sistem | Ringkasan Acuan |
| Teks | Edward Sirait merupakan Direktur umum Lion Air | Direktur umum Lion Air adalah Edward Sirait |
| Unigram | **Edward**, **Sirait**, merupakan, **Direktur**, **umum**, **Lion**, **Air** | **Direktur**, **umum**, **Lion**, **Air**, adalah, **Edward**, **Sirait** |
| Bigram | **Edward** **Sirait**, Sirait merupakan, merupakan Direktur, **Direktur** **umum**, **umum** **Lion**, **Lion** **Air** | **Direktur** **umum**, **umum Lion**, **Lion** **Air**, Air adalah, adalah Edward, **Edward Sirait** |
| Skip-bigram | **Edward** **Sirait**, Edward merupakan, Edward Direktur, Edward umum, Edward Lion, Edward Air, Sirait merupakan, Sirait Direktur, Sirait Lion, Sirait Air, merupakan Direktur, merupakan umum, merupakan Lion, merupakan Air, **Direktur umum**, **Direktur** **Lion**, **Direktur Air**, **umum Lion**, **umum Air**, **Lion Air** | **Direktur umum**, **Direktur Lion**, **Direktur Air**, Direktur adalah, Direktur Edward, Direktur Sirait, **umum Lion**, **umum Air**, umum adalah, umum Edward, umum Sirait, **Lion Air**, Lion adalah, Lion Edward, Lion Sirait, Air adalah, Air Edward, Air Sirait, adalah Edward, adalah Sirait, **Edward Sirait** |
| ROUGE-1 | 6/7 = 0.857 | |
| ROUGE-2 | 4/6 = 0.667 | |
| ROUGE-S (F1-Measure) | (2 \* 7/21 \* 7/21) / (7/21 + 7/21) = 0.333 | |
| ROUGE-SU (F1-Measure) | (2 \* 13/28 \* 13/28) / (13/28 + 13/28) = 0.464 | |

## Word Embedding

*Word embedding,* atau representasi kata terdistribusi, merupakan teknik populer untuk merepresentasikan kata pada pemrosesan bahasa alami. Ide utama dari *word embedding* adalah merepresentasikan setiap kata sebagai sebuah vektor bilangan riil dengan ukuran pendek, lalu setiap vektor memiliki hubungan, baik secara semantik maupun sintaksis, dengan vektor lainnya (Shi dkk., 2017). Secara umum, teknik *word embedding* terbagi menjadi dua, yaitu teknik berbasis pencacahan dan teknik berbasis prediksi (Baroni, 2014).

Mikolov dkk. (2013) mengembangkan dua buah model *word embedding* berbasis prediksi dengan jaringan saraf buatan dengan satu lapisan tersembunyi, yaitu skip-gram dan *continuous bag of words* (CBOW). Kedua model ini dikenal dengan nama Word2vec. Pada model skip-gram, model mempelajari bobot terbaik dengan masukan kata untuk menghasilkan keluaran kata sebelum dan kata sesudah . dikenal juga sebagai ukuran jendela kata. Sebaliknya, pada model CBOW, masukan model adalah kata sebelum dan sesudah , dan keluarannya adalah . Pada kedua model tersebut, nilai *word embedding* adalah bobot antara lapisan masukan dan lapisan tersembunyi pada jaringan.

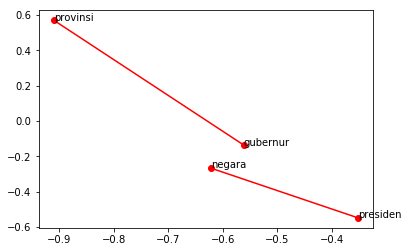
Berdasarkan model skip-gram, Bojanowski dkk. (2017) kemudian membangun model bernama FastText. FastText mampu membuat *word embedding* dengan memperhitungkan struktur morfologi dari kata. Morfologi kata dimodelkan berdasarkan komponen n-gram karakter dari kata, sehingga masukan dari model skip-gramkini bukan hanya berupa kata, melainkan komponen n-gram karakter. Tabel II.3. memperlihatkan contoh komponen n-gram karakter dari suatu kata. Nilai *word embedding* suatu kata adalah jumlah vektor seluruh n-gram karakter dan vektor kata itu sendiri.

Tabel II.3. Contoh Komponen N-Gram Karakter

| Kata | Komponen N-Gram Karakter (N = 3) |
| --- | --- |
| Pelangi | ‘pel’, ‘ela’, ‘lan’, ‘ang’, ‘ngi’ |
| Kota | ‘kot’, ‘ota’ |
| Pesawat | ‘pes’, ‘esa’, ‘saw’, ‘awa’, ‘wat’ |

Pada teknik berbasis pencacahan, masukan dari teknik adalah matriks *co-occurrence* kata dari korpus dengan ukuran jendela kata tertentu. Salah satu teknik sederhana untuk menghasilkan matriks *word embedding* adalah menggunakan transformasi *latent semantic analysis* (LSA) terhadap matriks *co-occurrence* kata (Foltz, 1996).Hingga saat ini, teknik *word embedding* berbasis pencacahan *state-of-the-art* adalah GloVe (Pennington dkk., 2014). GloVe mampu mempelajari hubungan antara dua buah kata dengan menghitung perbandingan probabilitas *co-occurrence* kedua kata terhadap kata-kata konteks.

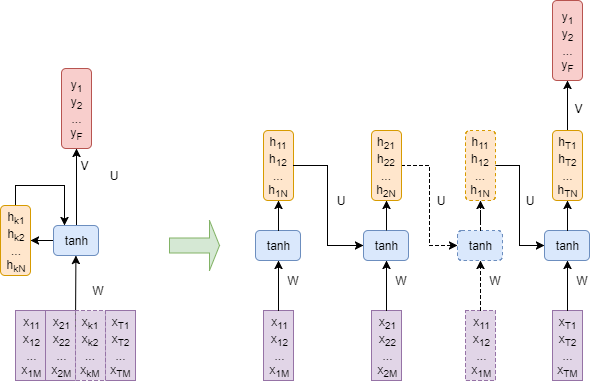
Salah satu evaluasi yang digunakan untuk mengetahui kualitas *word embeddding* adalah prediksi analogi kata, baik secara sintaksis maupun semantik (Mikolov dkk., 2013). Gambar II.1. memperlihatkan bahwa pasangan *word embedding* (provinsi, gubernur) memiliki hubungan semantik terhadap pasangan *word embedding* (negara, presiden). Berdasarkan eksperimen, jika dibandingkan dengan Word2Vec, GloVe memberikan akurasi analogi semantik yang lebih baik (Pennington dkk., 2014), sedangkan FastText memberikan akurasi analogi sintaksis yang lebih baik (Bojanowski dkk., 2017).



Gambar II.1. Contoh Visualisasi Word Embedding Bahasa Indonesia dengan Model FastText

## Recurrent Neural Network dan Gated Recurrent Unit

Recurrent Neural Network (RNN) merupakan sebuah ekstensi dari *feedforward* *neural network* sederhana (Chung dkk., 2014). RNN mampu menerima masukan dengan tipe sekuens. Berbeda dengan *feedforward neural network*, RNN ikut memperhitungkan *hidden state* sebelumnya ketika akan memperbarui *hidden state* suatu waktu. Gambar II.2. mengilustrasikan topologi RNN sederhana dengan masukan matriks sekuens , dan keluaran sebuah vektor .



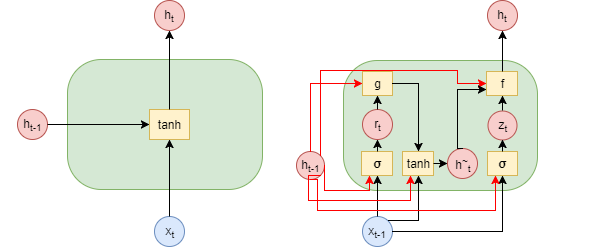
Gambar II.2. Topologi RNN Sederhana

Secara formal, apabila terdapat masukan sekuens , RNN sederhana menghitung nilai *hidden state* dengan persamaan (II-5).

|  |  |  |
| --- | --- | --- |
|  |  | (II-5) |

Di mana merupakan sebuah fungsi non-linier seperti fungsi logistik sigmoid. dan adalah bobot yang akan dipelajari. Kelemahan dari RNN standar sesuai persamaan di atas adalah permasalahan *vanishing gradient* (Hochreiter, 1997). Apabila sekuens masukan RNN terlalu panjang, maka gradien galat yang dihasilkan pada *state* RNN awal nilainya mendekati 0. Solusi yang populer adalah dengan membuat variasi arsitektur RNN yang mampu menyimpan memori lebih baik pada sekuens panjang, seperti LSTM (Hochreiter, 1997) dan GRU (Cho dkk., 2014).

Gated Recurrent Unit (GRU) diajukan oleh Cho dkk. (2014) untuk membuat sebuah unit *recurrent* yang mampu menangkap ketergantungan antar *hidden state* seiring dengan perubahan waktu dalam jangka waktu lebih panjang. GRU memiliki dua buah unit *gate* yang mengatur aliran informasi pada RNN, yaitu *update gate* dan *reset gate*. Ilustrasi perbedaan sel RNN biasa dan sel GRU dapat dilihat pada Gambar II.3.



Gambar II.3. Perbedaan Sel RNN (Kiri) dan Sel GRU (Kanan)

Hidden state GRU pada waktu t adalah interpolasi linier antara *hidden state* sebelumnya dengan kandidat *hidden state* sesuai dengan persamaan (II-6).

|  |  |  |
| --- | --- | --- |
|  |  | (II-6) |

Di mana merupakan *update gate* pada GRU. *Update gate* menentukan seberapa besar memori hasil perhitungan sebelumnya akan diperhitungkan. Persamaan (II-7) merupakan formula untuk menghitung *update gate*.

|  |  |  |
| --- | --- | --- |
|  |  | (II-7) |

Kandidat *hidden state* kemudian dihitung dengan fungsi yang serupa dengan perhitungan *update* pada RNN standar seperti pada persamaan (II-8).

|  |  |  |
| --- | --- | --- |
|  |  | (II-8) |

Terakhir, nilai keluaran dari *reset gate* dihitung dengan formula yang sama seperti *update gate*. *Reset gate* menghitung proporsi antara memori sebelumnya dan masukan saat ini. Persamaan (II-9) merupakan persamaan untuk menghitung keluaran *reset gate*.

|  |  |  |
| --- | --- | --- |
|  |  | (II-9) |

Dibandingkan dengan RNN tradisional yang menggunakan fungsi aktivasi tangen hiperbolik (), eksperimen Chung (2016) membuktikan bahwa GRU selalu mencapai konvergensi lebih cepat dan memberikan hasil lebih baik. Ketika dibandingkan dengan LSTM, GRU melakukan komputasi lebih cepat pada satu epoch dikarenakan jumlah fungsi komputasi yang lebih sedikit dibandingkan LSTM. Secara empirik, kinerja GRU dan LSTM tidak dapat dibandingkan karena kinerja kedua arsitektur memiliki kinerja terbaik pada dataset yang berbeda (Chung, 2016).

## Relasi *Discourse*

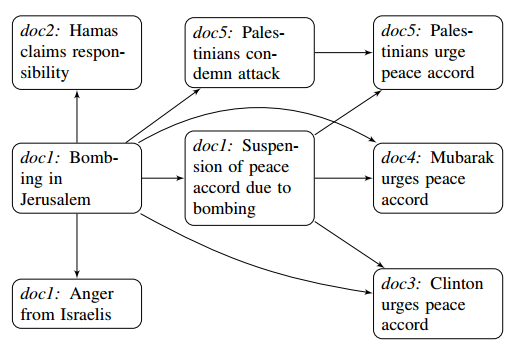
Relasi *discourse* atau relasi retorik antara dua segmen teks merupakan deskripsi bagaimana kedua segmen teks tersebut terhubung satu sama lain secara lojik (Asher & Nicholas, 2003). Mann & Thompson (1988) mencetuskan Rhetorical Structure Theory (RST), yaitu teori yang mendeskripsikan struktur hirarkis pada teks serta relasi *discourse* antar segmen teks. Relasi yang didefinisikan oleh Mann & Thompson memiliki jumlah 23 relasi.

Tabel II.4 memperlihatkan beberapa relasi beserta contoh kalimat masing-masing relasi RST. Pada sistem peringkasan otomatis, beberapa penelitian peringkasan telah mengaplikasikan relasi *discourse* dengan pembangunan pohon atau graf *discourse* kalimat. Marcu (1999) membangun struktur pohon *discourse* sesuai aturan RST yang dicetuskan oleh Mann & Thompson (1988). Pohon *discourse* dibangun dari segmen-segmen kalimat secara *bottom-up*. Berdasarkan pohon *discourse*, Marcu kemudian membangun ringkasan dengan meninjau kepentingan kalimat berdasarkan posisi simpul pada pohon *discourse*.Kalimat yang paling pertama dijadikan ringkasan adalah kalimat yang paling dekat dengan akar *pohon discourse*.

Tabel II.4. Contoh Kalimat yang Memiliki Hubungan *Discourse*

| No | Hubungan *Discourse* | Contoh Kalimat |
| --- | --- | --- |
| 1 | Contrast | Pemerintah memang sudah memberi bantuan kepada korban bencana. **Namun**, jumlah bantuan bencana dinilai masyarakat sedikit. |
| 2 | Cause-Explanation | Jakarta digenang banjir setinggi 2 meter. **Penyebab** dari banjir ini diduga adalah sungai di Jakarta yang mampat. |
| 3 | Elaboration | Sejak kecil, Irma sudah memiliki banyak prestasi**. Contohnya**, Irma pernah memenangkan Olimpiade Sains Nasional (OSN) ketika ia duduk di SD. |
| 4 | Sequence | Presiden Jokowi angkat bicara mengenai *meme* mengenai dirinya. **Beliau menyatakan** bahwa ia tidak mau ambil pusing. |

Christensen (2013) menggunakan relasi *discourse* antar kalimat yang direpresentasikan sebagai graf *discourse* pada penelitian peringkasan multi-dokumen. Graf ini dinamakan *approximate discourse graph* (ADG). Graf ini dinamakan “*approximate*” karena graf tidak mempedulikan jenis relasi antar segmen. Graf hanya memperhatikan urutan parsial antar segmen teks. Pada graf ini, sebuah simpul digambarkan sebagai segmen teks berupa kalimat, dan busur dari simpul ke simpul menyatakan bahwa urutan kalimat harus berada setelah kalimat . Gambar II.4. memperlihatkan contoh *graf discourse* yang dibangun. Tujuan pembangunan ADG adalah untuk menjaga keteraturan urutan kalimat yang dihasilkan pada ringkasan agar ringkasan koheren.



Gambar II.4. Contoh Graf *Discourse* (Christensen, 2013)

## Graph Convolutional Neural Network

Graph Convolutional Neural Network (GCN) merupakan topologi ANN yang mampu memproses data dengan representasi graf dengan operasi konvolusi (Kipf & Welling, 2017). Tujuan dari GCN adalah untuk mempelajari fungsi , di mana fungsi menerima masukan:

* , matriks ketetanggaan dari graf, dan N adalah jumlah simpul pada graf
* , matriks fitur dari setiap simpul, di mana D adalah dimensi dari vektor fitur simpul

Dan mengeluarkan fitur hasil operasi konvolusi dari masukan graf, yaitu matriks , di mana adalah dimensi dari vektor fitur keluaran. Fungsi dimodelkan sebagai propagasi antar lapisan seperti pada jaringan saraf. Misalkan matriks aktivasi pada lapisan ke disebut sebagai , dimulai dari . Keluaran dari GCN dengan L-layer adalah .

Model propagasi tanpa operasi konvolusi dari GCN untuk menghitung nilai matriks lapisan ke didefinisikan dengan persamaan sederhana (II-10).

|  |  |  |
| --- | --- | --- |
|  | ) | (II-10) |

Di mana adalah bobot pada lapisan ke- yang diperoleh dari hasil pembelajaran. Dari model propagasi sederhana tersebut, Kipf & Welling kemudian mengintegrasikan operasi konvolusi graf spektral ke persamaan propagasi, sehingga persamaan (II-10) diubah menjadi persamaan (II-11).

|  |  |  |
| --- | --- | --- |
|  |  | (II-11) |

Di mana merupakan fungsi aktivasi seperti . merupakan matriks ketetanggan dari graf dengan tambahan matriks identitas. merupakan matriks derajat dengan . Kipf & Welling kemudian membandingkan kinerja dari GCN dibandingkan dengan kinerja algoritma pembelajaran *semi-supervised* lainnya untuk masukan graf, dan GCN menghasilkan kinerja terbaik.

## Penelitian Terkait

### Improving the Estimation of Word Importance for News Multi-Document Summarization

Hong & Nenkova (2014) membangun sistem peringkasan otomatis dengan pendekatan ekstraktif dengan memperhatikan kepentingan dari setiap kata pada kalimat. Hong & Nenkova membuat sebuah model regresi logistik yang menerima masukan kata dalam bentuk fitur-fitur dan mengeluarkan nilai kepentingan dari kata tersebut. Berikut adalah rincian dari pembelajaran regresi logistik.

1. Masukan dari regresi logistik merupakan fitur dari suatu kata.
2. Keluaran target dari regresi logistik adalah 1 jika kata tersebut terdapat pada salah satu ringkasan acuan, dan 0 jika tidak.

Dengan rekayasa fitur yang cukup kompleks, Hong & Nenkova mendefinisikan fitur dari suatu kata dengan jumlah 9.261 fitur. Fitur dari suatu tersebut mencakup fitur-fitur standar suatu kata (posisi kata pada kalimat, POS tag, NER tag, huruf kapital, dst.) dan fitur baru yang diusulkan oleh penelitian.

Setelah menghasilkan kepentingan kata-kata, kepentingan suatu kalimat didefinisikan sebagai jumlah kepentingan seluruh kata pada kalimat dibagi dengan jumlah kata pada kalimat. Ketika membangun ringkasan, solusi yang digunakan adalah dengan pendekatan *greedy*. Kalimat dengan skor tertinggi akan diprioritaskan untuk dimasukkan sebagai kalimat ringkasan, kecuali kalimat tersebut merupakan kalimat redundan. Redundansi kalimat dapat dideteksi dengan perbandingan *cosine similarity* antara kalimat kandidat ringkasan dengan kalimat pada ringkasan yang akan dihasilkan.

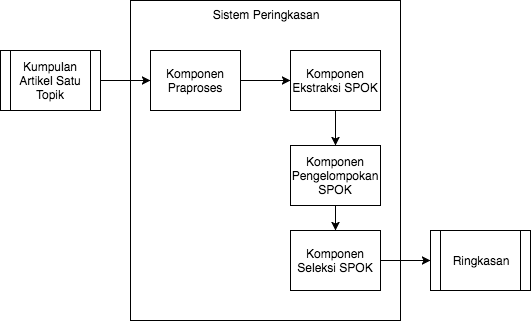
Tabel II.5. memperlihatkan hasil yang dicapai pada peringkasan Hong & Nenkova. REGBASIC merupakan model yang hanya menggunakan fitur-fitur kata standar, sedangkan REGSUM merupakan model yang ikut memperhitungkan fitur kepentingan kata yang diusulkan. Hasilnya, REGBASIC sudah memberikan kinerja yang memuaskan, walaupun REGSUM memberikan hasil yang lebih baik. Hingga saat ini, REGSUM merupakan sistem peringkasan multi-dokumen *state-of-the-art* berdasarkan nilai ROUGE-2 hasil ringkasan.

Tabel II.5. Hasil Evaluasi Peringkasan Hong & Nenkova (2014)

| Model | R-1 | R-2 |
| --- | --- | --- |
| REGBASIC | 37.56% | 9.28% |
| REGSUM | 38.57% | 9.75% |

### Peringkasan Otomatias Kumpulen Artikel Berita *Online* Berbasiskan Struktur Kalimat

Reztaputra & Khodra (2017) membangun sistem peringkasan otomatis dengan pendekatan abstraktif berdasarkan struktur kalimat subjek, predikat, objek, dan keterangan (SPOK) dari dokumen. Gambar II.5. memperlihatkan arsitektur sistem peringkasan. Komponen praproses memproses teks dengan tahapan pemisahan kalimat dan tokenisasi menjadi sekuens token. Dari sekuens token, komponen ekstraksi SPOK menghasilkan sekumpulan struktur SPOK berdasarkan penelusuran pohon kebergantungan. Komponen pengelompokan SPOK kemudian mengelompokkan dan menggabungkan setiap struktur SPOK menjadi beberapa kelompok dengan menggunakan *clustering* dan *string similarity*. Seleksi SPOK kemudian menyusun ringkasan dengan membangun kalimat berdasarkan setiap anggota kelompok SPOK dengan menggunakan algoritma *maximum margin relevance* (MMR) untuk mencegah informasi redundan pada ringkasan (Goldstein, 2000). Tabel II.6. memperlihatkan hasil evaluasi peringkasan dengan metrik ROUGE-2.,



Gambar II.5. Arsitektur Peringkasan Reztaputra & Khodra (2017)

Tabel II.6. Hasil Evaluasi Peringkasan Reztaputra & Khodra (2017)

| Metrik | Nilai |
| --- | --- |
| ROUGE-2 Recall | 0.276 |
| ROUGE-2 F-Score | 0.274 |

### Towards Coherent Multi-Document Summarization

Christensen (2013) membangun sebuah sistem peringkasan multi-dokumen yang dinamai G-Flow. G-Flow memperkenalkan metode pemilihan sekaligus pengurutan kalimat yang mampu memilih kalimat penting dengan baik, namun tetap menjamin ringkasan koheren. Representasi yang digunakan adalah representasi graf *discourse* antar kalimat, yaitu *approximate discourse graph* (ADG). Pada ADG, busur dari kalimat ke-i ke kalimat ke-j menyatakan bahwa dalam penyusunan ringkasan, kalimat tidak boleh ditulis sebelum kalimat .

Pembentukan busur dari ke dibangun dengan menghitung indikator apakah terdapat relasi *discourse* antar kedua kalimat tersebut, seperti *deverbal noun reference*, *event/entity continuation*, *discourse marker*, dan *co-referent*. Bobot positif pada busur tersebut merupakan jumlah indikator yang dipenuhi oleh dan . Misalnya, jika ke memiliki relasi *discourse* karena indikator *discourse marker*, dan acuan *co-referent*, maka bobot busur tersebut adalah dua. Selain bobot positif, graf dapat memiliki bobot negatif, yaitu apabila memiliki *deverbal noun reference*, *discourse marker*, atau acuan *co-referent* yang tidak dapat dipenuhi oleh . Bobot negatif memiliki arti bahwa jika kedua kalimat disusun dalam satu ringkasan, maka ringkasan dapat menjadi tidak koheren.

Untuk membuat ringkasan, akan diperhitungkan terlebih dahulu tiga aspek utama, yaitu koherensi ringkasan, kepentingan seluruh kalimat, dan redundansi. Koherensi ringkasan dihitung dengan persamaan (II-12).

|  |  |  |
| --- | --- | --- |
|  |  | (II-12) |

Di mana merupakan kumpulan busur positif dan merupakan kumpulan busur netgatif. merupakan koefisien untuk memberi bobot seberapa besar akan berpengaruh.

Kepentingan seluruh kalimat kemudian dihitung dengan menggunakan regresi linier dengan fitur-fitur klasik, seperti posisi kalimat, jumlah *proper noun* pada kalimat, dst. Parameter regresi linier dipelajari terhadap data latih dengan masukan kalimat berupa fitur-fiturnya dan keluaran skor ROUGE-1 dari kalimat tersebut terhadap ringkasan acuan. Kepentingan seluruh kalimat merupakan jumlah kepentingan dari setiap kalimat pada ringkasan seperti pada persamaan (II-13).

|  |  |  |
| --- | --- | --- |
|  |  | (II-13) |

Untuk mengatasi redundansi, setiap kalimat awalnya diproses terlebih dahulu menggunakan OLLIE (Mausam dkk., 2012), di mana OLLIE akan mengubah kalimat menjadi komponen *tuple* relasi dengan bentuk (argumen 1, frasa relasi, argumen 2). Kedua kalimat dinyatakan redundan apabila pada *tuple* masing-masing kalimat, terdapat satu argumen yang merupakan sinonim, dan terdapat kata kerja yang merupakan sinonim pada frasa relasinya.

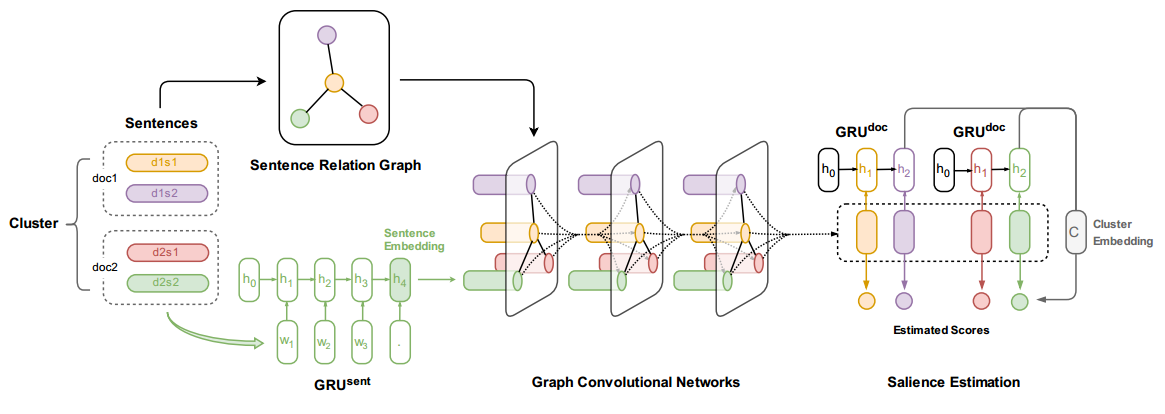
Dari ketiga konsep di atas, maka persamaan (II-14) adalah fungsi objektif yang harus dipenuhi untuk menghasilkan ringkasan terbaik.

|  |  |  |
| --- | --- | --- |
|  |  | (II-14) |

Parameter dan akan dicari nilai optimalnya dari data pengembangan. Untuk memaksimalkan nilai , metode yang dilakukan adalah dengan pencarian lokalmenggunakan algoritma *stochastic hill climbing* dengan *random restart*. Pada setiap langkah, pencarian dapat menambah kalimat, menghapus kalimat, mengganti seuatu kalimat dengan kalimat lain, atau mengganti urutan sepasang kalimat.

### Graph-based Neural Multi-Document Summarization

Yasunaga (2017) membangun sebuah sistem untuk peringkasan multi-dokumen dengan topologi GCN. Yasunaga memanfaatkan GRU untuk menghasil vektor *embedding* dari kalimat dan dokumen, dan memanfaatkan GCN untuk menghasilkan *sentence embedding* yang memperhitungkan keterhubungan antar kalimat.



Gambar II.6. Arsitektur Peringkasan Multi-Dokumen Berbasis Graf Neural (Yasunaga, 2017)

Gambar II.6. menggambarkan arsitektur peringkasan multi-dokumen pada hasil penelitian Yasunaga (2017). Berikut adalah rincian dari tahapan arsitektur tersebut.

1. Menggambarkan hubungan seluruh kalimat pada suatu *cluster*. Misalkan suatu kalimat pada suatu dokumen direpresentasikan sebagai sidj, maka suatu sidj akan memiliki sisi dengan bobot *similarity* dengan sxdy (x ≠ y ∨ i ≠ j) dengan suatu *metric*. Yasunaga menggunakan 3 buah *metric* untuk pengukuran bobot sisi, yaitu TF-IDF *cosine similarity*, *approximate discourse graph* (ADG) (Christensen, 2013), dan *personalized discourse graph* (PDG) yang merupakan modifikasi ADG.

ADG dinilai kurang layak untuk dijadikan sebagai masukan GCN. Hal ini dikarenakan bobot busur pada ADG nilainya hanya bisa bernilai kelipatan satu saja, sehingga bobot antar kalimat nilainya banyak yang sama. Selain relasi *discourse* antar kalimat, bobot antar kalimat seharusnya juga memuat kepentingan dari kedua kalimat antar simpul. Maka dari itu, Yasunaga kemudian mengajukan perhitungan bobot pada PDG dengan persamaan (II-15).

|  |  |  |
| --- | --- | --- |
|  |  | (II-15) |

Di mana merupakan nilai skor personalisasi kalimat yang dihitung dengan regresi linier yang dipelajari dari skor ROUGE-1 masing-masing kalimat terhadap ringkasan acuan. Fitur-fitur yang digunakan merupakan fitur klasik dari kalimat.

1. Membuat representasi *word embedding* untuk seluruh kata pada setiap kalimat menggunakan Word2vec.
2. Membuat *sentence embedding* untuk setiap kalimat dengan memasukkan sekuens *word embedding* kalimat ke GRUsent. *Sentence embedding* merupakan *hidden state* terakhir dari arsitektur GRUsent.
3. Membuat graf hubungan antar *sentence embedding* dengan persamaan II-1, di mana X merupakan matriks berukuran N x D, dengan N adalah jumlah kalimat dan D adalah panjang vector *sentence embedding*, dan A merupakan matriks ketetanggan berukuran N x N.
4. Membuat *document embedding* untuk setiap dokumen dengn sekuens *sentence embedding* dari dokumen ke GRUdoc. *Document embedding* merupakan *hidden state* terakhir dari arsitektur GRUdoc.
5. Menghitung nilai *cluster embedding* C dengan persamaan (II-16).

|  |  |  |
| --- | --- | --- |
|  |  | (II-16) |

Di mana merupakan *document embedding* ke-i dan adalah jumlah dokumen dari suatu kluster.

1. Menghitung estimasi kepentingan kalimat dengan persamaan (II-17) dan (II-18)

|  |  |  |
| --- | --- | --- |
|  |  | (II-17) |
|  |  | (II-18) |

Di mana merupakan parameter bobot yang harus dipelajari, adalah nilai *cluster embedding*, dan adalah nilai *sentence embedding*.

Untuk mempelajari seluruh bobot pada arsitektur *deep learning,* pembelajaran mesin dilakukan dengan masukan kalimat-kalimat artikel setopik dan keluaran target vektor kepentingan kalimat, di mana kepentingan target kalimat merupakan nilai persamaan (II-19).

|  |  |  |
| --- | --- | --- |
|  |  | (II-19) |

Di mana dan merupakan rata-rata skor ROUGE-1 dan skor ROUGE-2 dari kalimat . merupakan nilai konstanta yang digunakan untuk membuat distribusi lebih tajam. Pembaruan bobot kemudian dilakukan dengan teknik *backpropagation* berdasarkan fungsi galat persamaan (II-20).

|  |  |  |
| --- | --- | --- |
|  |  | (II-20) |

Dari struktur graf kalimat yang dibentuk, umumnya, kalimat yang memiliki skor kepentingan tinggi merupakan kalimat yang memiliki busur terberat dan terbanyak. Hal ini menyatakan bahwa kalimat yang memiliki hubungan dengan sebagian besar kalimat dokumen merupakan kalimat penting pada ringkasan. Tabel II.7. memperlihatkan hasil evaluasi arsitektur pada dataset DUC 2004. *Baseline* yang digunakan adalah peringkasan yang hanya menggunakan GRU saja. Dari tiga metrik bobot sisi graf yang digunakan, metrik PDG memberikan hasil terbaik.

Tabel II.7. Hasil Evaluasi Peringkasan Berbasis Graf Neural pada Data DUC 2004 (Yasunaga, 2017)

| Arsitektur | R-1 | R-2 |
| --- | --- | --- |
| GRU | 36.64% | 8.47% |
| GRU+GCN: Cosine Similarity Graph | 37.33% | 8.78% |
| GRU+GCN: ADG from G-Flow | 37.41% | 8.97% |
| GRU+GCN: Personalized Discourse Graph | 38.23% | 9.48% |

# ANALISIS DAN RANCANGAN SISTEM PERINGKASAN OTOMATIS

## Analisis Masalah

Tujuan dari tugas akhir ini adalah membangun sistem peringkasan otomatis multi-dokumen untuk artikel berita dengan bahasa Indonesia yang dapat menghasilkan ringkasan semirip mungkin dengan ringkasan buatan manusia. Ringkasan multi-dokumen yang baik adalah ringkasan yang mengandung informasi utama dari seluruh dokumen serta informasi unik dari setiap dokumen (Goldstein, 2008). Untuk menganalisis masalah, akan dilakukan evaluasi terhadap beberapa penelitian peringkasan multi-dokumen terkait.

Peringkasan berbasis korpusmembutuhkan fitur yang baik untuk menghasilkan ringkasan yang baik. Hong & Nenkova (2014) kemudian berhasil membangun sistem peringkasan multi-dokumen dengan pendekatan ekstraktif *state-of-the-art* berdasarkan kepentingan kata*.* Meskipun tahap penilaian kalimat hanya menggunakan regresi logistik untuk mendapatkan kepentingan kata, fitur yang digunakan jumlahnya adalah 9.261. Sebagian besar fitur tersebut membutuhkan fitur rekayasa manusia yang cukup kompleks. Beberapa fitur memerlukan korpus selain data latih, seperti korpus Gigaword dan New York Times. Beberapa fitur juga membutuhkan *tools* dengan kualitas baik serta spesifik bahasa seperti *named entity recognition* (NER) dan pohon kebergantungan, dan *tools* tersebut belum ada yang memiliki kinerja baik pada bahasa Indonesia.

Ketika membangun ringkasan, aspek keterhubungan antar kalimat merupakan salah satu hal yang penting.Keterhubungan antar kalimat dapat dilihat berdasarkan kemiripan atau relasi *discourse*.Hong & Nenkova (2014) tidak memperhitungkan keterhubungan antar kalimat sama sekali dalam memberikan nilai kepentingan terhadap kalimat. Padahal, penelitian Christensen (2013) menyatakan bahwa untuk menghasilkan ringkasan yang koheren, relasi *discourse* antar kalimat perlu diperhatikan.

Pada bahasa Indonesia, sistem peringkasan Reztaputra & Khodra (2017) memiliki kinerja *recall* ROUGE-2 sebesar 0.276. Ringkasan dengan kualitas buruk umumnya disebabkan oleh kesalahan ekstraksi SPOK. Kesalahan ekstraksi SPOK sendiri merupakan akibat dari kualitas pohon kebergantungan bahasa Indonesia yang digunakan. Sistem peringkasan Christie & Khodra (2016) memiliki kinerja *recall* ROUGE-2 dengan perbedaan tipis, yaitu 0.275. Meskipun kinerjanya sedikit lebih buruk, Christie & Khodra hanya menggunakan *tools* *part-of-speech* (POS) tag dan pembuatan graf kata, dan dua *tools* tersebut memiliki kinerja baik pada bahasa Indonesia. Dalam penyusunan ringkasan, meskipun penggunaan MMR oleh Reztaputra & Khodra (2017) dan ILP oleh Christie & Khodra (2016) menjamin bahwa ringkasan yang dihasilkan tidak redundan, kedua optimasi tersebut tidak menjamin bahwa ringkasan yang dihasilkan pasti koheren.

Dari analisis di atas, diperoleh masalah sebagai berikut.

1. Bagaimana cara mempelajari fitur kepentingan kalimat tanpa bergantung kepada *tools* spesifik bahasa Indonesia?
2. Bagaimana cara memperhitungkan hubungan antar kalimat untuk mengetahui nilai kepentingan kalimat sebagai kalimat ringkasan?

## Analisis Solusi

Dari masalah yang didapatkan, dilakukan perancangan solusi yang merupakan hasil pemetaan dari masalah. Tabel III.1. memperlihatkan pemetaan dari masalah hasil analisis beserta rancangan solusinya. Rincian dari rancangan solusi akan dijelaskan pada subbab III.2.1 dan III.2.2.

Tabel III.1. Pemetaan Masalah dan Rancangan Solusi

| Masalah | Solusi |
| --- | --- |
| Bagaimana cara mempelajari fitur kepentingan kalimat tanpa bergantung kepada *tools* spesifik bahasa Indonesia? | Pembelajaran fitur dengan *deep learning* |
| Bagaimana cara memperhitungkan hubungan antar kalimat untuk mengetahui nilai kepentingan kalimat sebagai kalimat ringkasan? | Representasi graf *discourse* dan pembelajaran dengan GCN |

### Analisis Pembelajaran Fitur dengan *Deep Learning*

Untuk memilih kalimat yang layak menjadi bagian ringkasan, diperlukan proses ekstraksi fitur pada setiap kalimat untuk menghasilkan fitur yang baik. Umumnya, fitur yang baik juga membutuhkan *tools* dengan kualitas baik, seperti NER dan pohon kebergantungan. Sayangnya, pada bahasa Indonesia, kualitas *tools* tersebut masih buruk. Kesalahan ekstraksi SPOK pada sistem peringkasan Reztaputra & Khodra disebabkan oleh kesalahan penguraian struktur pohon kebergantungan, dan kesalahan ekstraksi SPOK mengakibatkan sistem menghasilkan ringkasan yang buruk.

*Deep learning* dikenal dapat mempelajari parameter untuk menghasilkan keluaran yang baik dengan masukan dengan fitur tingkat rendah*. Deep learning* memiliki potensi menggantikan fitur kalimat hasil rekayasa manusia dengan fitur tingkat rendah kalimat hasil pembelajaran. Topologi *deep learning* yang cocok untuk menangani masukan teks adalah RNN yang cocok menangani masukan sekuensial. Topologi GCN juga mampu menangani masukan dengan tipe graf. Yasunaga (2017) telah menggunakan RNN dan GCN untuk membangun sebuah sistem peringkasan otomatis.

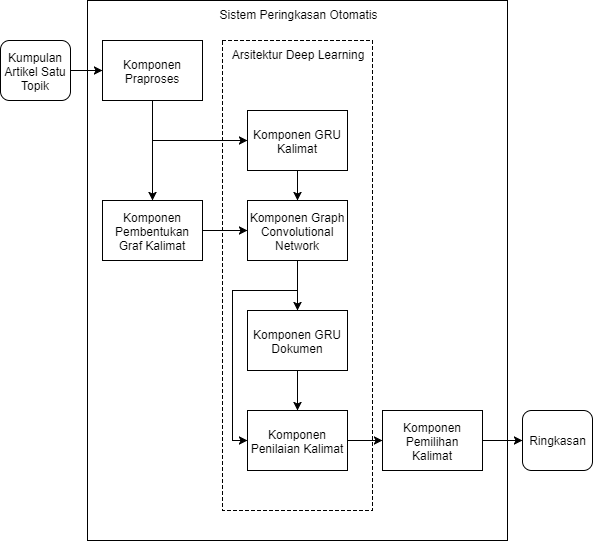
### Analisis Representasi Graf *Discourse* dan Pembelajaran dengan GCN

Pada beberapa sistem peringkasan, ketika mempertimbangkan fitur dari informasi yang pantas untuk dijadikan sebagai ringkasan, fitur tidak mencakup keterhubungan antar informasi, baik dari segi kemiripan maupun relasi *discourse*. Hong & Nenkova (2014) tidak memperhatikan keterhubungan antar kalimat sama sekali dalam memilih kelayakan kalimat untuk dijadikan ringkasan. Meskipun Christie & Khodra (2016) sudah mengelompokkan kalimat berdasarkan kemiripannya, penyusunan kalimat berdasarkan graf kata dengan optimasi menggunakan ILP tidak menjamin ringkasan koheren. Hal serupa juga berlaku untuk sistem peringkasan Reztaptura & Khodra (2017) yang melakukan pengelompokan SPOK berdasarkan kemiripan serta seleksi SPOK menggunakan optimasi MMR.

Penggunaan struktur graf *discourse* kalimat berpotensi meningkatkan kualitas hasil peringkasan. Untuk membangun ringkasan yang koheren, Christensen (2013) membangun sistem peringkasan yang secara bersamaan memperhitungkan kepentingan kalimat sekaligus koherensi ringkasna berdasarkan urutan parsial kalimat sesuai dengan graf ADG. Yasunaga (2017) mengembangkan teknik Christensen dengan memanfaatkan *deep learning* dan GCN. Dengan masukan graf *sentence embedding* beserta keterhubungannya, GCN kemudian menghasilkan *sentence embedding* baru yang mempertimbangkan koherensi ringkasan. Yasunaga kemudian mengusulkan graf *discourse* PDG yang merupakan integrasi graf ADG dengan skor kepentingan kalimat berdasarkan fitur klasik, dan PDG terbukti menghasilkan kinerja yang lebih baik dibandingkan ADG.

## Rancangan Sistem

Secara garis besar, sistem peringkasan otomatis terdiri atas 7 komponen, di mana 4 komponen pada sistem merupakan bagian dari arsitektur *deep learning*.Gambar III.1. memperlihatkan arsitektur sistem peringkasan otomatis secara lengkap.



Gambar III.1. Arsitektur Sistem Peringkasan Otomatis

### Komponen Praproses

Tahap praproses akan mengubah sebuah kalimat menjadi sekuens token. Tahap praproses terdiri atas dua tahapan, yaitu pemisahan kalimat dan tokenisasi. Pemisahan kalimat dilakukan agar teks pada suatu dokumen dibagi menjadi beberapa kalimat berdasarkan tanda baca. Kemudian, proses tokenisasi dilakukan terhadap masing-masing kalimat untuk membagi satu kalimat menjadi beberapa token. Sebuah token dapat berupa sebuah kata atau tanda baca. Keluaran dari tahap praproses adalah sekuens token, di mana urutan sekuens token sama dengan urutan kemunculan token pada kalimat. Tabel III.2. memperlihatkan contoh hasil proses pemisahan kalimat dan tokenisasi. Pada contoh, token ditandai dengan tanda baca kutip satu.

Tabel III.2. Contoh Hasil Praproses

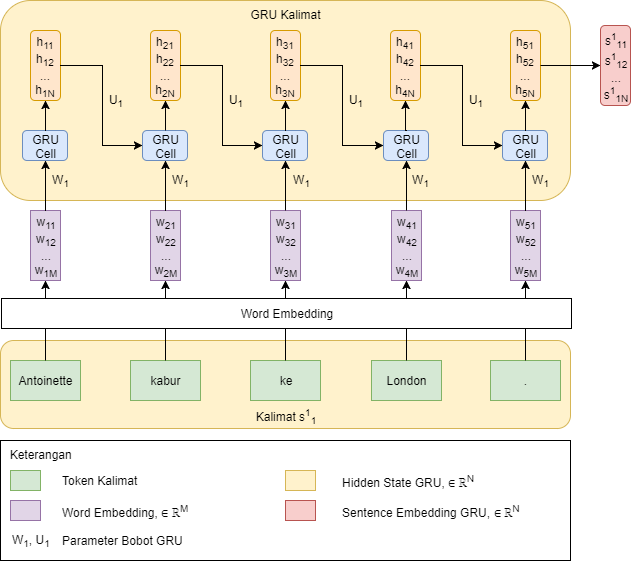
| Masukan | Keluaran Praproses |
| --- | --- |
| Direktur Umum Lion Air, Edward Sirait, angkat bicara terkait insiden yang terjadi di maskapainya. Dia menyatakan kabar adanya kesalahan mendarat pesawat tersebut memang benar adanya. | * [‘Direktur’, ‘Umum’, ‘Lion’, ‘Air’, ‘,’, ‘Edward’, ‘Sirait’, ‘,’, ‘angkat’, ‘bicara’, ‘terkait’, ‘insiden’, ‘yang’, ‘terjadi’, ‘di’, ‘maskapainya’, ‘.’] * [‘Dia’, ‘menyatakan’, ‘kabar’, ‘adanya’, ‘kesalahan’, ‘mendarat’, ‘pesawat’, ‘tersebut’, ‘memang’, ‘benar’, ‘apa’, ‘adanya’, ‘.’] |

### Komponen GRU Kalimat

Komponen GRU kalimat akan menghasilkan vektor *sentence embedding* yang merupakan representasi vektor dari kalimat. Gambar III.2. memperlihatkan ilustrasi pembentukan *sentece embedding* pada GRU kalimat. GRU kalimat memiliki dua buah *hyperparameter* sebagai berikut.

1. , yaitu jumlah neuron pada setiap *hidden state* yang dihasilkan GRU kalimat.
2. , yaitu panjang maksimum sekuens pada GRU kalimat.

Masukan dari GRU adalah sebuah kalimat. Kalimat ke-i dokumen ke-j, , akan direpresentasikan sebagai sekuens token yang dihasilkan tahap praproses, yaitu . Karena masukan GRU pada waktu ke-t harus berupa vektor, maka sebuah token pada akan direpresentasikan menjadi vektor *word embedding* , di mana adalah panjang vektor *word embedding* dan adalah posisi token pada sekuens token. Pada eksperimen ini, akan digunakan empat buah *word embedding* yang dihasilkan dengan teknik berbeda-beda, yaitu LSA (Foltz, 1997), Word2vec (Mikolov dkk., 2013), GloVe (Pennington dkk., 2014), dan FastText (Bojanowski dkk., 2017).



Gambar III.2. Ilustrasi Proses Pembentukan *Sentence Embedding* Menggunakan GRU

Setelah setiap token diubah menjadi *word embedding*, maka sebuah kalimat kini direpresentasikan sebagai matriks , di mana adalah jumlah token pada kalimat dan adalah panjang vektor *word embedding*. Tentunya nilai pada setiap kalimat berbeda. Karena panjang sekuens GRU nilainya tetap, yaitu *,* GRU hanya dapat menerima masukan dengan ukuran . Oleh karena itu, perlu dilakukan proses *padding* dan *pruning* untuk setiap dengan rincian sebagai berikut.

1. Jika , maka akan dilakukan proses *padding* terhadap matriks ,yaitu menambal matriks dengan vektor sebanyak sehingga ukuran matriks menjadi .
2. Jika , maka akan dilakukan proses *pruning* terhadap matriks, yaitu mengecilkan ukuran matriks menjadi dengan membuang elemen-elemen pada kolom .

Pada waktu ke-t (), GRU akan menghitung *hidden state* ke-t dari hasil komputasi , , dan matriks bobot dan seperti pada persamaan (II-3). Nilai *sentence embedding* dari sekuens token adalah nilai vektor *hidden state* terakhir, yaitu . Nilai bobot dan merupakan hasil pembelajaran arsitektur *deep learning*. Keluaran dari komponen GRU kalimat adalah kumpulan *sentence embedding* dalam bentuk matriks , di mana adalah jumlah seluruh kalimat pada artikel satu topik.

### Komponen Pembentukan Graf

Masukan komponen ini adalah sekuens token keluaran komponen praproses. Dari sekuens token, komponen pembentukan graf akan menghasilkan matriks ketetanggaan antar kalimat. Pada eksperimen ini, akan dibangun tiga graf dengan representasi yang berbeda-beda. Rincian dari setiap graf adalah sebagai berikut.

1. *Cosine similarity graph*

Graf ini merupakan graf kemiripan antar kalimat. Kemiripan kalimat dinilai dari nilai *cosine similarity* antar kalimat. Untuk membangun *cosine similarity graph,* akan dibangun matriks TF-IDF kalimat x kata, sehingga setiap kalimat dapat direpresentasikan dengan vektor TF-IDF. Bobot antara dua kalimat nilainya adalah *cosine similarity* dari vektor TF-IDF dua kalimat tersebut. Graf kalimat yang dibangun merupakan graf komplit tidak berarah.

1. *Approximate discourse graph* (ADG)

ADG merupakan graf *discourse* dari seluruh kalimat pada kluster topik. Pada ADG, bobot dari kalimat ke kalimat menunjukkan urutan parsial dari ke kalimat . Jika nilai positif, maka memiliki relasi *discourse* dengan , di mana dalam penyusunan ringkasan, kalimat tidak boleh ditulis sebelum kalimat untuk menjaga ringkasan koheren. Jika nilai yang negatif, maka tidak memiliki relasi *discourse* dengan . Berbeda dengan graf *cosine similarity*¸ busur ADG sifatnya berarah.

Pada sistem peringkasan Christensen (2013), terdapat lima aspek yang digunakan untuk menentukan ADG. Dikarenakan keterbatasan *tools* pada bahasa Indonesia, maka eksperimen ini hanya akan menggunakan satu aspek saja, yaitu *discourse marker*. *Discourse marker* merupakan kata kunci yang menunjukkan keterhubungan antara dua kalimat, seperti kata ‘walaupun’, ‘namun’, dsb. Tabel III.3. memperlihatkan contoh pembangunan sisi graf ADG, di mana kata yang dicetak tebal adalah *discourse marker*. Dengan *discourse marker* saja, maka berikut adalah kemungkinan bobot antara simpul dan .

1. Jika merupakan kalimat dengan posisi persis sebelum kalimat ( = ) dan kata pertama pada merupakan kata *discourse marker*, maka nilainya 1.
2. Jika bukan merupakan kalimat dengan posisi persis sebelum kalimat dan kata pertama pada merupakan kata discourse marker, maka nilainya -1.
3. Jika dan tidak memenuhi kasus (a) dan (b), maka nilai adalah 0.

Tabel III.3. Contoh Pembangunan Sisi Graf ADG

| Contoh Pembangunan Sisi Graf ADG | |
| --- | --- |
| Dokumen | Surono mengatakan bahwa gempa yang terjadi di Dataran Tinggi Dieng pukul 19.00--19.25 WIB terekam sebanyak 86 kali dengan amplituda maksimum 10--100 milimeter (mm) dan lama gempa 10--70 detik. Kemudian, menurutnya gempa tersebut terasa hampir di seluruh wilayah Dataran Tinggi Dieng dengan skala MMI III--V. Sebagian besar warga sudah mengungsi ke tempat yang disediakan oleh pemerintah. |
| Kalimat 1 | Surono mengatakan bahwa gempa yang terjadi di Dataran Tinggi Dieng pukul 19.00--19.25 WIB terekam sebanyak 86 kali dengan amplituda maksimum 10--100 milimeter (mm) dan lama gempa 10--70 detik. |
| Kalimat 2 | **Kemudian**, menurutnya gempa tersebut terasa hampir di seluruh wilayah Dataran Tinggi Dieng dengan skala MMI III--V. |
| Kalimat 3 | Sebagian besar warga sudah mengungsi ke tempat yang disediakan oleh pemerintah. |
| Ilustrasi Graf |  |

1. *Personalized discourse graph* (PDG)

Pada eksperimen Yasunaga (2017), meskipun ADG memberikan hasil yang lebih baik dari graf *cosine similarity*, kebanyakan busur memiliki nilai bobot yang sama, dan setiap bobot hanya dapat bernilai kelipatan satu. Yasunaga memodifikasi ADG dengan mengganti nilai bobot busur ADG dengan nilai bobot yang juga mempertimbangkan skor personalisasi kalimat. Skor personalisasi kalimat adalah skor kepentingan kalimat berdasarkan fitur-fitur klasik yang tertera pada Tabel III.4.

Skor personalisasi diperoleh dari hasil perhitungan regresi linier. Untuk mendapatkan parameter yang tepat, pembelajaran regresi linier dilakukan terhadap korpus data latih, di mana masukan regresi adalah fitur-fitur sebuah kalimat sesuai Tabel III.4., dan keluaran targetnya adalah skor ROUGE-1 dari kalimat masukan terhadap ringkasan acuan. Bobot PDG dari ke kemudian dihitung dengan persamaan (III-1).

|  |  |  |
| --- | --- | --- |
|  |  | (III-1) |

Di mana dan merupakan dua buah kalimat, merupakan skor personalisasi untuk kalimat , dan merupakan bobot antara kalimat dan kalimat pada graf ADG.

Tabel III.4. Fitur Pembelajaran Regresi Linier Skor Personalisasi

| No | Fitur |
| --- | --- |
| 1 | Posisi kalimat |
| 2 | Apakah kalimat merupakan tiga kalimat pertama dokumen |
| 3 | Panjang kalimat |
| 4 | Apakah panjang kalimat lebih dari 20 |
| 5 | Jumlah *proper noun* yang ada pada kalimat |
| 6 | Jumlah kalimat lain yang mengacu kata kerja pada kalimat |
| 7 | Jumlah kalimat lain yang mengacu *proper noun* pada kalimat |
| 8 | Jumlah kalimat lain yang mengacu *common noun* pada kalimat |

### Komponen Graph Convolutional Network

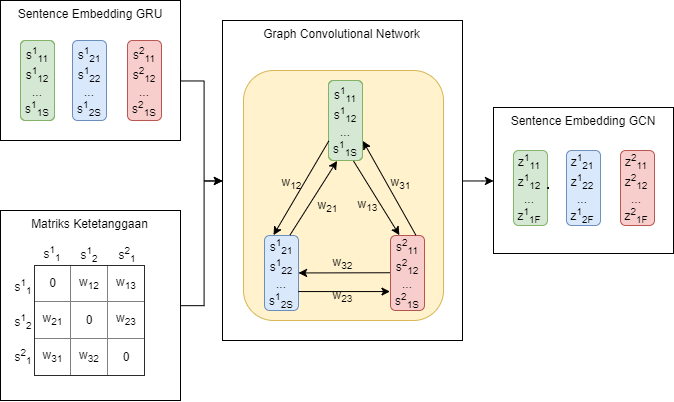
Komponen GCN akan menghasilkan *sentence embedding* yang baru. *Sentence embedding* yang dihasilkan GCN merupakan hasil perhitungan *sentence embedding* dari GRU beserta keterhubungan kalimat dari *sentence embedding* tersebut dengan kalimat lainnya. Gambar III.3 menggambarkan ilustrasi pembuatan *sentence embedding* GCN dengan topologi GCN. Komponen GCN menerima dua buah masukan sebagai berikut.

1. Matriks *sentence embedding* GRU yang dihasilkan oleh GRU kalimat, yaitu , di mana N merupakan jumlah *sentence embedding* GRU dan merupakan panjang setiap vektor *sentence embedding* GRU. merupakan elemen baris matriks berupa *sentence embedding* kalimat ke-i dokumen ke-j.
2. Matriks ketetanggaan antar kalimat, yaitu , di mana merupakan jumlah kalimat pada seluruh dokumen.

Kedua masukan kemudian diproses oleh GCN. Dengan pemrosesan GCN, setiap vektor *sentence embedding* akan ditransformasi menjadi vektor *sentence embedding* baru , di mana merupakan panjang vektor *sentence embedding* keluaran GCN. dibentuk dengan dari dengan perhitungan operasi konvolusi sebanyak kali terhadap vektor dan matriks ketetanggan , di mana merupakan jumlah lapisan pada GCN. Dengan demikian, keluaran dari GCN adalah matriks , di mana adalah jumlah seluruh kalimat.

Pada eksperimen ini, jumlah lapisan GCN yang akan digunakan adalah tiga lapisan sesuai eksperimen Yasunaga (2017). Berdasarkan persamaan (II-7), maka jika dan , hasil keluaran GCN sebanyak tiga lapisan dihasilkan oleh persamaan (III-3).

|  |  |
| --- | --- |
|  | (III-3) |



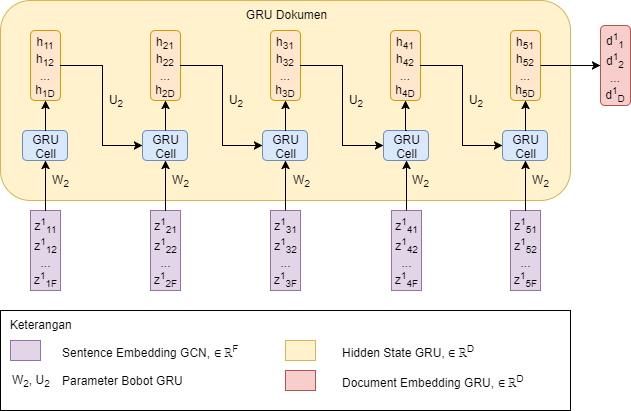
Gambar III.3. Ilustrasi GCN dengan Satu Lapisan

Di mana merupakan bobot pada lapisan ke-*l* yang nilainya merupakan hasil pembelajaran, dan merupakan dimensi fitur dari setiap simpul yang dihasilkan satu lapisan GCN. Khusus, untuk menghasilkan keluaran dengan dimensi , maka nilai Karena terdapat tiga lapisan, maka jumlah bobot yang harus dipelajari pada komponen GCN ada 3 buah, yaitu , , dan .

### Komponen GRU Dokumen

GRU dokumen akan menghasilkan *document embedding* dari setiap dokumen artikel setopik. Gambar III.4. memperlihatkan ilustrasi pembentukan *document embedding* dari sekuens *sentence embedding*.GRU dokumen memiliki *hyperparameter* sebagai berikut.

1. , yaitu jumlah neuron pada setiap *hidden state* yang dihasilkan GRU dokumen.
2. , yaitu panjang maksimum sekuens pada GRU dokumen.



Gambar III.4. Ilustrasi GRU Dokumen dan Perhitungan *Cluster Embedding*

Masukan dari komponen GRU dokumen adalah matriks *sentence embeddding* hasil GCN , dan merupakan elemen baris matriks berupa *sentence embedding* kalimat ke-i dokumen ke-j. Sebelum diproses, setiap *sentence embedding* dikelompokkan terlebih dahulu berdasarkan dokumen kalimat tersebut berada. Hasil pengelompokan adalah buah sekuens *sentence embedding,* di mana merupakan jumlah dokumen pada artikel setopik. Elemen pada sekuens *sentence embedding* terurut sesuai dengan kemunculan kalimat pada dokumen terkait.

Berdasarkan hasil pengelompokan di atas, sebuah dokumen , kini dapat direpresentasikan dengan matriks berukuran , di mana merupakan panjang setiap *sentence embedding*, dan *K* merupakan jumlah kalimat pada . Mirip dengan persoalan pada GRU kalimat, panjang sekuens GRU dokumen nilainya tetap , sedangkan nilai pada setiap dokumen jumlahnya berbeda. Oleh karena itu, perlu dilakukan proses *padding* dan *pruning* dengan rincian sebagai berikut.

1. Jika , maka akan dilakukan proses *padding* terhadap matriks ,yaitu menambal matriks dengan vektor sebanyak sehingga ukuran matriks menjadi .
2. Jika , maka akan dilakukan proses *pruning* terhadap matriks , yaitu mengecilkan ukuran matriks menjadi .dengan membuang elemen-elemen pada kolom .

Pada waktu ke-i (), GRU dokumen akan menghitung *hidden state* ke-i dari hasil komputasi , , dan matriks bobot dan seperti pada persamaan (II-3). Nilai *document embedding* dari sekuens *sentence embedding* adalah nilai vektor *hidden state* terakhir, yaitu. Nilai bobot dan merupakan hasil pembelajaran dari arsitektur *deep learning*. Keluaran dari arsitektur ini adalah kumpulan *document embedding* dalam bentuk matriks , di mana adalah jumlah seluruh dokumen pada artikel satu topik.

### Komponen Penilaian Kalimat

Komponen penilain kalimat akan menghasilkan nilai kepentingan dari kalimat. Semakin tinggi nilai kepentingan, maka semakin layak kalimat dijadikan bagian dari ringkasan. Komponen ini menerima dua buah masukan sebagai berikut.

1. Matriks *document embedding* . merupakan elemen baris matriks berupa *document embedding* dokumen ke-j.
2. Matriks *sentence embedding* keluaran GCN *.* merupakan elemen baris matriks berupa *sentence embedding* kalimat ke-i dokumen ke-j.

Untuk menghitung nilai kalimat, maka akan dihitung terlebih dahulu nilai *cluster embedding.* Nilai *cluster embedding* merupakan hasil perhitungan persamaan (II-16), yaitu rata-rata dari seluruh vektor *document embedding* yang dihasilkan GRU dokumen. Setelah nilai *cluster embedding* didapatkan, akan dihitung nilai dengan mempertimbangkan *cluster embedding .* Nilai dihitung dengan persamaan (III-4).

|  |  |
| --- | --- |
|  | (III-4) |

Di mana , , dan merupakan bobot yang nilainya merupakan hasil pembelajaran, dan merupakan nilai vektor *cluster embedding*. Nilai dimensi merupakan *hyperparameter* dari persamaan (III-4). Kemudian, nilai kepentingan setiap kalimat akan dinilai dengan persamaan (III-5).

|  |  |
| --- | --- |
|  | (III-5) |

Di mana merupakan himpunan seluruh kalimat pada seluruh artikel satu topik. Semakin tinggi nilai , maka semakin layak kalimat untuk menjadi bagian dari ringkasan.

### Komponen Pemilihan Kalimat

Setelah seluruh kalimat memiliki nilai kepentingan, seluruh kalimat akan diurutkan berdasarkan nilai kepentingannya terurut mengecil. Penyusunan ringkasan dilakukan dengan pendekatan *greedy*. Kandidat kalimat dengan skor tertinggi kemudian akan ditambahkan ke ringkasan selama kalimat tersebut bukanlah kalimat redundan. Kalimat redundan adalah kandidat kalimat yang mirip dengan salah satu kalimat yang terdapat pada ringkasan.

Untuk mengetahui kalimat redundan, akan dibangun matriks TF-IDF kalimat x token terhadap dari kalimat. Dari matriks TF-IDF tersebut, setiap kalimat dapat direpresentasikan sebagai sebuah vektor TF-IDF. Jika vektor kandidat kalimat memiliki *cosine similarity* > 0.5 dengan salah satu vektor kalimat ringkasan, maka kandidat kalimat tersebut merupakan kalimat redundan.

## Pembelajaran Komponen Deep Learning

Untuk menemukan bobot-bobot terbaik pada arsitektur *deep learning*, dimulai dari bobot pada komponen GRU kalimat hingga komponen penilaian kalimat, sistem perlu melakukan pembelajaran terhadap data latih. Teknik pembelajaran yang dilakukan adalah teknik *backpropagation*.Rincian dari pembelajaran yang dilakukan adalah sebagai berikut.

1. Tahap *forward pass*

Pada tahap ini, masukan dari tahap *forward pass* adalah kalimat-kalimat pada seluruh artikel satu topik. Keluaran prediksi dari tahap *forward pass* adalah vektor skor kepentingan kalimat seperti pada persamaan (III-4). Keluaran target tahap *forward pass* adalah hasil perhitungan persamaan (III-6).

|  |  |  |
| --- | --- | --- |
|  |  | (III-6) |

Di mana merupakan rata-rata ROUGE-1 dan ROUGE-2 dari kalimat seperti pada persamaan (III-7).

|  |  |  |
| --- | --- | --- |
|  |  | (III-7) |

1. Tahap *backward pass*

Nilai galat dihitung berdasarkan perbedaan dari skor kepentingan kalimat hasil prediksi dengan keluaran target dengan fungsi galat *cross entropy* seperti pada persamaan (III-8).

|  |  |  |
| --- | --- | --- |
|  |  | (III-8) |

Di mana merupakan himpunan kalimat pada seluruh artikel satu topik. Dari nilai galat tersebut, akan dihitung nilai derivasi turunan dari fungsi-fungsi yang ada pada arsitektur *deep learning*. Setiap komponen pada arsitektur *deep learning* memiliki nilai derivasi turunan yang berbeda-beda.

1. Tahap pembaruan bobot

Nilai bobot dari setiap komponen pada arsitektur *deep learning* diperbarui berdasarkan nilai derivasi turunan dari setiap fungsi pada komponen arsitektur *deep learning*.

## Implementasi Sistem

Implementasi sistem akan dilakukan dengan menggunakan bahasa pemrograman Python. *Library* NLTK digunakan untuk melakukan pemisahan kalimat. *Library* Kerasakan digunakan untuk membangun arsitektur *deep learning* dari rancangan solusi, yaitu topologi ANN, GRU, dan GCN.

## Sistem *Baseline*

Selain sistem peringkasan berbasis GCN, akan dibangun juga beberapa sistem peringkasan yang akan dijadikan sebagai *baseline*. Sistem peringkasan yang akan dibangun jumlahnya dua, yaitu sistem B1, B2. Pada sistem B1, sistem peringkasan otomatis mirip dengan sistem rancangan solusi, namun tanpa menggunakan GCN. Pada sistem B2, sistem peringkasan otomatis dengan pendekatan ekstraktif berdasarkan kepentingan kata (Hong & Nenkova, 2014). Selain kedua sistem di atas, hasil sistem peringkasan bahasa Indonesia pada penelitian Christie & Khodra (2016) dan Reztaputra & Khodra (2017) juga dijadikan *baseline*.

DAFTAR PUSTAKA

Asher, N., & Lascarides, A. (2003). *Logics of Conversation*. Cambridge Unversity Press. ISBN 0-521-65058-5.

Baroni, M., dkk. (2014). Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics,* pp. 238-247.

Barzilay, R., & Lapata, M. (2008). Modeling local coherence: An entity-based approach. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*.

Cao, Z., dkk. (2016). AttSum: Joint learning of focusing and summarization with neural attention. *arXiv:1604.00125*.

Cheng, J., & Lapata, M. (2016). Neural summarization by extracting sentences and words. *arXiv:1603.07252*.

Cho, K., dkk. (2014). Learning phrase representation using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*.

Christie, F., & Khodra, M. L. (2016). Multi-document summarization using sentence fusion for Indonesian news article. *2016 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA).* IEEE*.*

Christensen, J., dkk. (2007). Towards coherent multi-document summarization. *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Chung, J., dkk. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*.

Foltz, P. W. (1996). Latent semantic analysis for text-based research. *Behavior Research Methods, Instruments, & Computers*, Vol. 28, No. 2, pp. 197-202

Galanis, D., dkk. (2012). Extractive multi-document summarization with integer linear programming and support vector regression. *Proceeddings of COLING 2012: Technical Papers*, pp. 911-926.

Goldstein, J., dkk. (2000). Multi-document summarization by sentence extraction. *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization,* pp. 40-48*.*

Hahn, U., & Mani, I. (2000). The challenges of automatic summarization. *IEEE* *Computer Journal*, Vol. 33, No. 11, pp. 29–36.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Journal of Neural Computation*, Vol. 9, No. 8, pp. 1735-1780.

Hong, K., & Nenkova, A. (2014). Improving the estimation of word importance for news multi-document summarization. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 712-721.

Jurafsky, D., & Martin, J. H. (2007). *Speech and Language Processing*. Englewood Cliffs, NJ: Prentince-Hall, Inc.

Kipf, T. N., & Welling, M. (2017). Semi supervised classfication with graph convolutional networks. *International Conference on Learning Representations (ICLR) 2017*.

Kupiec, J., dkk. (1995). A trainable document summarizer. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 68-73.

Li, S., dkk. (2007). Multi-document summarization using support vector regression. *Proceedings of DUC*. Citeseer.

Lin, C. (2004). ROUGE: A package for automatic evaluation of summaries. *Proceedings of the Workshop on Text Summarization Branches (WAS 2004)*.

Ma, X., & Hovy, E. (2016). End-to-end sequence modeling via bi-directional lstm-cnns-crf. *arXiv:1603.01354*.

Mani, I., & Maybury, M. (1999). *Advances in Automatic Text Summarization*. Cambridge, Massachusetts: The MIT Press.

Mann, W., & Thompson, S. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Interdisciplinary Journal for the Study of Discourse*, Vol. 8, No. 3.

Marcu, D. (1999). Discourse trees are good indicators of importance in text. *Advances in Automatic Text Summarization*.

Marcu, D., & Echihabi, A. (2002). An unsupervised approach to recognizing discrouse relations. In *Proceedings of the 40th Annual Meeting of the Association for Computional Linguistics (ACL)*, pp. 368-375.

Mausam. (2012). Open language learning for information extraction. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CONLL)*, pp. 523-534.

Mikolov, T., dkk. (2013). Distributed representations of words and phrases and their compositionality. *arXiv:1310.4546*.

Pennington, J., dkk. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Reztaputra, R., & Khodra, M. L. (2017). Sentence structure-based summarization for Indonesian news article. *2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA).* IEEE*.*

Rush, A., dkk. (2015). A neural attention model for abstractive sentence summarization. *arXiv:1509.00685*.

Shi, Bei. (2017). Jointly learning word embeddings and latent topics. *arXiv:1706.07276*.

Yasunaga, M., dkk. (2017). Graph-based neural multi-document summarization. *arXiv:1706.06681*.

1. Contoh Judul Lampiran
   1. Contoh Judul Anak Lampiran

Contoh anak lampiran