BAB 2
TINJAUAN PUSTAKA

2.1 Peringkasan Teks Otomatis

Peringkasan sebuah teks adalah sebuah proses untuk menghasilkan ringkasan dari sebuah dokumen berbentuk berita, ulasan, artikel, atau makalah dari internet. Ringkasan adalah satu cara yang dapat membantu untuk mendapatkan sebuah informasi dari suatu dokumen yang panjang dalam waktu yang singkat. Sementara peringkasan teks otomatis adalah proses penyulingan informasi untuk menghasilkan versi yang lebih singkat dengan menggunakan sebuah sistem yang dijalankan dengan komputer. Terdapat dua tipe peringkasan teks yaitu [1]:

1. Ekstraktif

   Tipe peringkasan ekstraktif menghasilkan ringkasan dengan cara memilih bagian yang penting dalam teks dan menampilkannya kembali dengan tidak merubah intisari dan teks yang aslinya.
2. Abstraktif

   Tipe peringkasan teks abstraktif menghasilkan ringkasan dengan cara menginterpretasikan dan menguji dengan menggunakan bahasa alami tingkat lanjut untuk memilih teks singkat yang baru namun tidak merubah intisari dari teks yang aslinya.

2.2 Berita

Berita adalah informasi tentang kejadian yang baru dengan penuturan secara benar dan sesuai dengan fakta yang tidak dibuat-buat [9]. Berita harus memiliki unsur-unsur penting seperti *what, who, where, when, why,* dan *how*, atau dikenal sebagai 5W+1H. Bahasa yang digunakan dalam sebuah berita harus menggunakan bahasa yang baku agar mudah dipahami.

2.3 Text Preprocessing

Text Preprocessing adalah proses data awal untuk merubah data yang awalnya tidak terstruktur menjadi suatu data baru yang terstruktur [10]. Data baru yang sudah terstruktur dapat mempermudah sistem untuk memproses algoritma agar menjadi lebih mudah. Tahapan *text preprocessing* yang digunakan dalam penelitian ini terdiri dari *case folding, filtering, tokenizing, stemming, stopword,* dan vektorisasi.

2.3.1 Case Folding

Tahap ini bertujuan sebagai menyamakan setiap karakter dengan cara merubah setiap huruf kapital (*uppercase*) dalam kalimat menjadi huruf kecil (*lowercase*) [11]. Tahap ini diperlukan karena tidak setiap data yang digunakan merupakan karakter yang sama sehingga diperlukan tahap ini agar semua karakter yang dimasukan memiliki nilai yang sama.

2.3.2 Tokenizing

Proses ini digunakan sebagai pemisah kata dari teks yang disebut sebagai token. Proses ini dilakukan dengan cara memisahkan setiap kata yang terdapat dalam teks berdasarkan spasi (“ ”) sebagai pemisah di setiap kata [11].

2.3.3 Stopword

Tahap ini merupakan proses untuk menghilangkan kata yang tidak memiliki arti dalam suatu teks [12]. Agar mengetahui kata apa saja yang merupakan *stopword* dapat menggunakan kamus *stopword* yang sudah ada.

2.3.4 Word Embedding

Word embedding merupakan teknik untuk merubah sebuah kata menjadi vektor atau array yang terdiri dari kumpulan angka. Untuk memproses data tekstual algoritma ML perlu mengkonversi data tekstual menjadi bentuk numerik, dikarenakan mesin tidak dapat memahami data tekstual yang sebagaimana dipahami oleh manusia [13]. Salah satu cara yang biasanya digunakan untuk membaca data tekstual tersebut adalah dengan merubah setiap kata menjadi

integer dengan memberinya nomor. Angka-angka tersebut diubah lagi menjadi vektor yang memiliki panjang sebanyak data yang ada di kamus. Array tersebut hanya bernilai 1 atau 0 yang di sebut juga *one hot encoding*. Nilai 1 diposisikan pada indeks yang merupakan nomor kata sedangkan yang lainnya bernilai 0. Dalam kasus ini jika kamus memiliki ukuran yang sangat banyak, maka untuk setiap katanya akan diubah menjadi vektor ukuran tersebut yang hampir setiap elemennya bernilai 0. Oleh karena itu metode *one hot encoding* ini kurang efisien dalam memori dan tidak memberikan banyak informasi.

Dengan metode fasttext dapat merubah kata menjadi sebuah vektor yang berisi angka dengan ukuran yang cukup kecil untuk mendapatkan informasi yang lebih banyak dibandingkan dengan menggunakan *one hot encoding*.

2.3.6.1 Vektorisasi

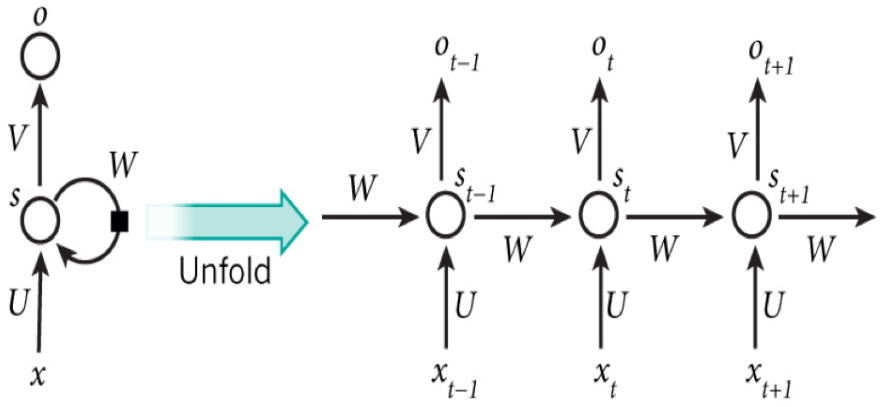
Pada tahap ini, metode yang digunakan untuk membuat *word embedding* adalah *FastText.* Metode ini dapat mempelajari vektor kata dengan baik, *FastText* dikembangkan oleh Mikolov et al (2013) [14]. *FastText* itu sendiri merupakan pengembangan *library* dari *Word2Vec* yang juga merupakan *library* untuk *word embedding*. Tidak seperti *Word2Vec*, model *FastText* menghasilkan kata vektor menggunakan sub-word (terdapat karakter pada setiap katanya), contohnya kata yang mau dirubah menjadi vektor akan dibagi menjadi n-gram kata sehingga kata akan menjadi pecahan kata. Dalam *FastText* vektor dari kata akan merupakan sebuah bilangan antara 0 sampai 1 dan memiliki besar dimensi yang dibutuhkan.

Model *FastText* ini memiliki kemampuan untuk mendapatkan vektor untuk kata apapun meskipun kata tersebut merupakan kata yang salah eja (*typo*) atau gabungan dari kata [15].

2.4 *Recurrent Neural Network* (RNN)

Recurrent Neural Network merupakan jenis neural network yang memiliki koneksi berulang pada setiap selnya, biasa digunakan sebagai ekstraksi fitur pada sebuah data yang bersifat *sequential*. RNN ini merupakan perkembangan dari

jaringan syaraf tiruan dan arsitekturnya hampir menyerupai Multi Layer Perceptron (MLP) [16]. Gambar 2.1 Merupakan gambaran dari arsitektur Recurrent Neural Network.

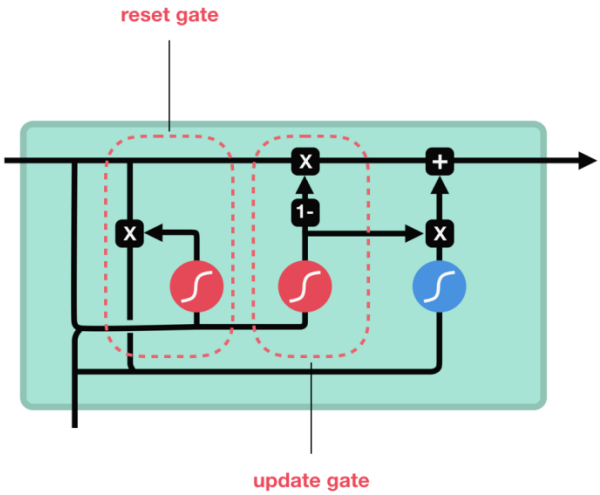


Gambar 2. 1 Arsitektur Recurrent Neural Network (RNN)

Dapat dilihat pada gambar diatas bahwa RNN akan mengulangi setiap proses pada setiap cellnya, yang dimana hasil pada cell sebelumnya akan mempengaruhi hasil pada cell setelahnya begitu juga seterusnya.

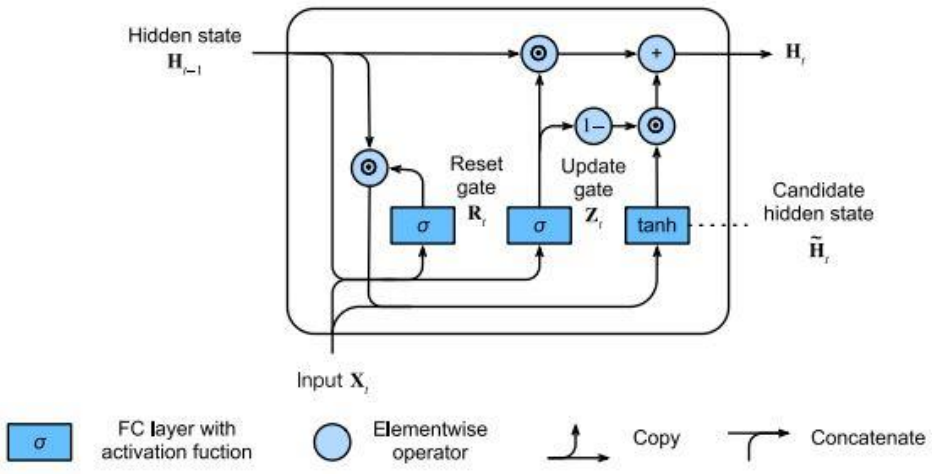
2.4.1 *Gated Recurrent Unit* (GRU)

Gated Recurrent Unit (GRU) dikembangkan oleh Cho et all (2014) [5]. Pada dasarnya *Gated Recurrent Unit* (GRU) merupakan dua vektor yang memutuskan informasi yang harus tetap diteruskan ke output. Dasar *workflow* pada jaringan *Gated Recurrent Unit* mirip dengan dasar dari Recurrent Neural Network ketika diilustrasikan, perbedaan utama antara keduanya adalah dalam setiap unit berulang sebagai jaringan unit gated berulang terdiri dari gerbang yang memodulasi arus input dan keadaan tersembunyi sebelumnya. GRU memiliki dua gerbang (*gate*) untuk memilih informasi yang harus dibuang dan disimpan. Berikut pada Gambar 2.2 merupakan gambaran dari *cell memory* beserta *gates* dalam arsitektur GRU.



Gambar 2. 2 Cell Memory dan Gates GRU

Seperti yang ditunjukan dengan gambar 2.2, *cell memory* pada GRU memiliki dua buah gates antara lain *reset gate* yang berfungsi sebagai yang memutuskan informasi masa lalu yang akan dilupakan, dan *update gate* yang berfungsi sebagai yang memutuskan informasi apa yang akan dibuang dan informasi baru apa yang akan disimpan. Gambar 2.3 mengilustrasikan *cell memory* pada GRU [17].



Gambar 2. 3 Ilustrasi Cell Memory pada GRU

Berikut Persamaan dari Ilustrasi *Cell Memory* pada GRU [12]:

$$R_t = \sigma\ (Wr\ [Ht{-}1,\ X_t\ ] + b) \quad (2.1)$$

$$Z_t = \sigma\ (Wu\ [Ht{-}1,\ X_t\ ] + b) \quad (2.2)$$

$$\tilde{H}\ t = \tanh\ (Wc\ [Rt * Ht{-}1, Xt\ ] + b \quad (2.3)$$

$$Ht = Zt * Ht{-}1 + (1 - Zt) * \tilde{H}\ t \quad (2.4)$$

Keterangan :

R_t = *Reset Gate* X_t = *Input saat ini*

Z_t = *Update Gate* H_t = *New state*

$\tilde{H}\ t$ = *Candidate hidden state*

H_{t-1} = *Hidden state sebelumnya*

Berikut persamaan dari *sigmoid* ($\boldsymbol{\sigma}$) dan ***tanh*** :

$$\boldsymbol{\sigma} = \frac{1}{1 + e^{-x}} \quad (2.5)$$

$$\textbf{tanh} = \frac{2}{1 + e^{-2x}} - 1 \quad (2.6)$$

Keterangan :

x = *Nilai masukan*

2.5 Metode Evaluasi Rouge

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) merupakan *toolkit* yang banyak digunakan oleh DUC untuk evaluasi ringkasan otomatis. *ROUGE* mengukur akurasi ringkasan dengan menghitung unit yang menyerupai n-gram, kata urutan, dan pasangan kata pada ringkasan sistem dan ringkasan

manual. Pada evaluasi *ROUGE-N* ini yang merupakan penarikan n-gram adalah ringkasan kandidat dan kumpulan ringkasan manual. Sehingga *ROUGE-N* memiliki persamaan sebagai berikut :

$$ROUGE - N = \frac{\sum_{S \in Ref} \sum_{Gram_n \in S} Count_{match}(Gram_n)}{\sum_{S \in Ref} \sum_{Gram_n \in S} Count(Gram_n)} \quad (2.7)$$

Keterangan :

N = *Panjang n-gram*

Ref = *Ringkasan Referensi (Manual)*

$Count_{match}$ = Jumlah n-Gram yang sama dengan Ref

ROUGE-N, dan *ROUGE-L* dianggap sebagai perincian teks yang dibandingkan antara ringkasan sistem dan ringkasan manual.

1. *ROUGE-N* merupakan pengukuran tingkat n-gram seperti *ROUGE-1* (unigram), dan *ROUGE-2 (bigram).*
2. *ROUGE-L* merupakan pencocokan kata terpanjang dari peringkasan sistem, *ROUGE-L* ini tidak memerlukan kecocokan berurutan tetapi mencocokan yang mencerminkan urutan kata dalam kalimat.

Dengan begitu *ROUGE* melakukan pengukuran dengan melihat kesamaan antara ringkasan hasil model yang diringkas dan yang dibuat oleh manusia[18].

2.6 Python

Python adalah bahasa pemrograman interpretatif yang mudah dipelajari, karena python dianggap memiliki kode-kode pemrograman yang lengkap dan mudah untuk dipahami. Python pertama kali dikembangkan oleh Guido van Rossum pada tahun 1990 pada CWI, dan versi terakhir yang dikembangkan oleh CWI adalah 1.2. pada tahun 1995 [19].

Python dapat dijalankan dengan berbagai macam *platform* sistem operasi, sehingga aplikasi yang dibuat dengan menggunakan python sangat luas. Beberapa

platform yang mendukung python adalah Linux/Unix, Windows, Mac OS, Java Virtual Machine, OS/2.

Python sangat digemari karena dapat mengolah data dengan kecepatan tinggi dan kepraktisan dalam penulisan kode, dan python memiliki sifat *cross platform* sehingga kode yang serupa dapat dijalankan pada berbagai macam sistem operasi.

2.6 Jupyter Notebook

Jupyter Notebook sebelumnya dikenal sebagai *IPython Notebook* yang berbasis bahasa pemrograman Python [20], *jupyter notebook* pertama kali dipisahkan dari *IPython* pada tahun 2014. *Jupyter notebook* adalah komputasi interaktif berbasis web, dokumen notebook jupyter adalah dokumen JSON yang biasanya diakhiri dengan ekstensi ".ipynb".