E-ISSN: 2655-867X
P-ISSN: 2655-8661
Volume 3 No. 01 Febuari 2021

PEMBANGKITAN KALIMAT ILMIAH MENGGUNAKAN RECURRENT NEURAL NETWORK

Reza Dwi Putra[1], Ridwan Ilyas[2], Fatan Kasyidi[3]
[1,3]Jurusan Informatika, Universitas Jenderal Achmad Yani[2]Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung,
e-mail : rdp1497@gmail.com[1], rdwnilyas@gmail.com[2], fatan.kasyidi@lecture.unjani.ac.id[3]

Abstrak

Text Generation merupakan pekerjaan dasar Natural Language Processing (NLP), yang memainkan peran penting dalam sistem dialog dan terjemahan cerdas. Text Generation merupakan sistem yang dapat membangkitkan text berupa kalimat secara otomatis dari teks atau dokumen dengan menggunakan metode atau beberapa pola tertentu. Recurrent Neural Network (RNN) merupakan arsitektur jaringan saraf tiruan yang telah terbukti berkinerja baik karena pemrosesannya disebut berulang kali untuk memproses input data sekuensial. Penelitian ini telah berhasil membuat model komputasi pembuatan teks menggunakan RNN, dengan fitur yang telah diekstraksi menggunakan fungsi Word2Vec untuk menghasilkan satu set vektor. Dalam melakukan proses pembuatan teks penelitian ini menggunakan total 1000 data kalimat ilmiah. Penelitian ini melakukan perbandingkan dengan tiga optimasi yaitu Adam, Nadam, dan Adamax untuk menemukan tingkat pembelajaran terbaik dan cocok untuk pembuatan teks. Hasil tingkat pembelajaran terbaik diperoleh dengan pengoptimalan Adamax dengan nilai skor BLEU yang dihasilkan mencapai 28. Hal ini menunjukan bahwa kualitas hasil terjemahan dari sistem cukup baik dalam menghasilkan kalimat yang direkomendasikan.

Kata Kunci*:* Text Generation, Natural Language Processing, Recurrent Neural Networks, Word2vec

Abstract

Text Generation is the basic work of Natural Language Processing (NLP), which plays an important role in intelligent dialogue and translation systems. Text generation is a system that can generate text in the form of sentences automatically from text or documents using certain methods or patterns. Recurrent Neural Network (RNN) is a proven neural network architecture that performs well because the processing is repeated to process sequential data input. This research has succeeded in creating a computational model for creating text using RNN, with features that have been extracted using the Word2Vec function to generate a set of vectors. In carrying out the process of making the text of this study using a total of 1000 scientific sentence data. This study made a comparison with three optimizations namely Adam, Nadam, and Adamax to find the best and suitable level of learning for text creation. The best learning level results were obtained by optimizing Adamax with the resulting BLEU score reaching 25. This shows that the quality of the translation results from the system is quite good at producing recommended sentences.

Keywords: *Text Generation, Natural Language Processing, Recurrent Neural Networks, Word2vec*

1. PENDAHULUAN

Setiap tahunnya karya tulis ilmah akan terus berkembang sesuai dengan semakin banyaknya beberapa hasil penelitian yang ditemukan, beberapa kalimat ilmiah yang dihasilkan juga akan semakin beragam [1]. Keberagaman tersebut diperngaruhi oleh penggunaan rujukan yang digunakan untuk mendukung, menguatkan, atau memperbaharui penelitian yang telah dilakukan sebelumnya, dalam rujukan sendiri berarti membuat kalimat baru dengan rangkaian kata yang berbeda atau mengurangi kata yang telah dikutip untuk mendukung penelitian yang akan dilakukan.

Sitasi merupakan salah satu penanda yang mencerminkan bagaimana para peneliti merangcang penelitian mereka dan juga ikut serta mempengaruhi pengambilan argumen untuk para peneliti di masa depan [2].

E-ISSN: 2655-867X
P-ISSN: 2655-8661
Volume 3 No. 01 Febuari 2021

Menuliskan sitasi merupakan bentuk pengakuan terhadap pengarang, ide, pendapat, gagasan atau bahkan teorinya telah kita gunakan.

Kalimat ilmiah yang terdapat dalam sebuah makalah ilmiah dapat dibangkitkan, karena dalam membangkitkan kalimat ilmiah harus mempertimbangkan kesamaan kata, urutan, kesamaan makna dari setiap kalimat, tata bahasa yang baik dan banyaknya kemungkinan pilihan kata dengan pendekatan Natural Language Processing (NLP).

Natural Language Processing (NLP) merupakan cabang ilmu komputer yang mendalami interaksi antara komputer dengan bahasa alami. Beberapa penelitian yang menggunakan pendekatan NLP salah satunya untuk peringkasan [3], deteksi plagiat [4], klasifikasi analisis sentimen ulasan film, respon crisis pada twitter [5], relasi kata dan kalimat [6], pembuatan teks cerita [7], pembuatan teks puisi [8] dan translasi bahasa [9]. NLP Dalam penelitian ini membahas mengenai Text Generation pada kalimat ilmiah yang merupakan salah satu contoh dari beberapa peneltian terkait pemrosesan NLP.

Text Generation beberapa tahun terakhir semakin berkembang, karena kebutuhan pembuatan dalam Text Generation terus meningkat dan menjadi semakin penting selain itu Text Generation merupakan bagian dari NLP yang memanfaatkan pengetahuan dalam linguistik komputasi dan kecerdasan buatan untuk secara otomatis menghasilkan teks bahasa alami, yang dapat memenuhi persyaratan komunikatif tertentu.

Hasil penelitian sebelumnya memperkenalkan pembuatan teks kutipan, lalu diberi sepasang dokumen ilmiah, yang menjelaskan hubungan dalam teks bahasa alami dengan cara mengutip dari satu teks ke teks lainnya. Hasil ini mendorong sistem untuk mempelajari hubungan antara teks ilmiah dan untuk mengungkapkannya secara konkret dalam bahasa alami. Model untuk generasi teks kutipan akan membutuhkan pemahaman dokumen yang kuat termasuk kapasitas, cepat beradaptasi dengan kosakata baru dan untuk bernalar tentang konten dokumen dengan menggunakan model GPT-2 [10]. Hasil yang didapatkan pada penelitian tersebut menghasilkan Skor BLEU tertinggi dengan hasil 10.23, dengan menggunakan *subset* dari 154.000 artikel mengenai korpus bidang ilmu komputer dan mengekstrak 622.000 kalimat ilmiah yang dihubungkan kembali ke dokumen lain dalam korpus tersebut.

Recurrent Neural Network (RNN) dipilih dikarenakan jenis arsitektur jaringan saraf tiruan yang dapat memproses data secara sekuensial. RNN merupakan model arsitektur yang sering digunakan dalam beberapa kasus NLP [11].

Pada penelitian lain Text Generation menggunakan RNN dan Long Short Term Memory (LSTM) menghasilkan akurasi mencapai 97%, dengan pengaturan konfigurasi yaitu Bidirectional LSTM, 256 *hidden units*, lapisan dalam menggunakan fungsi aktivasi *relu*, lapisan luar menggunakan aktivasi *softmax*, ektraksi fitur menggunakan Word2vec. Data yang digunakan pada pelatihan penelitian menggunakan The Lord of the Rings (LOR) dataset dan dibatasi 100.000 kalimat pada dataset tersebut [12].

Pada penelitian sebelumnya Text Generation menggunakan RNN menghasilkan akurasi sebesar 63% yang di evaluasi dengan survey responden. Hasil eksperimen yang dilakukan dengan *train loss* minimal 0,01, ukuran RNN 512 dengan 3 lapisan dan *batch size* 100 dengan panjang urutan 50, pada set pelatihan ada sekitar 30 cerita yang dimana setiap cerita terdiri dari 5000 kata, akurasi sistem dapat ditingkatkan lebih lanjut dengan mempertimbangkan makna kontekstual dari kata-kata tersebut. Selain itu, sinonim dapat digunakan untuk lebih meningkatkan akurasi sistem [13].

Pada penelitian lain mengenai Text Generation menggunakan LSTM mendapatkan akurasi sebesar 30%. Hasil eksperimen yang dilakukan dengan sequence *length* 60, *bacth size* 80, menggunakan optimasi Adam dengan *learning rate* 0,002, dengan fungsi aktivasi *relu* dan beberapa *dropout* yang diset berbeda *size* pada setiap *layer* dalam membangun percakapan secara otomatis [14].

Penelitian ini telah membangun sistem yang dapat membangkitkan kalimat kutipan

E-ISSN: 2655-867X
P-ISSN: 2655-8661
Volume 3 No. 01 Febuari 2021

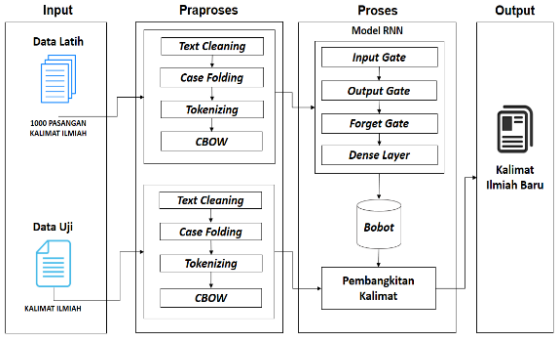
dari karya ilmiah secara otomatis dengan menggunakan metode pemrosesan kalimat Word2vec dengan konfigurasi menggunakan Continous Bag-Of-Word (CBOW) untuk merubah kedalam fitur kata menjadi sebuah vektor serta metode pelatihan yang digunakan untuk pembangkitan kalimat ilmiah yaitu RNN dengan konfigurasi LSTM dan GRU. Hasil penelitian ini memberikan rekomendasi kalimat ilmiah yang baru untuk dapat dimanfaatkan dalam penelitian yang akan dibangun, serta mendukung dan memenuhi penelitian terutama dalam pembuatan kalimat ilmiah.

2. KAJIAN PUSTAKA

Kajian pustaka dilakukan pengumpulan semua referensi dan memahami konsep dari sistem yang akan dibangun tentang studi kasus yang diambil, yaitu mengenai pembangkitan kalimat dan algoritma Recurrent Neural Network dengan konfigurasi Long Short Term Memory. Kajian pustaka diperoleh dari berbagai sumber referensi seperti jurnal, prosiding, website resmi dan laporan Tugas Akhir (TA).

3. METODE PENELITIAN

Tahap penelitian dimulai dengan mendapatkan korpus dataset kalimat ilmiah sebagai subjek yang digunakan untuk melatih data. Data pada sistem pembangkitan kalimat ilmiah akan diproses dengan praproses yaitu Text Cleaning, Case Folding, Tokenizing, kemudian data harus diekstraksi oleh Word2Vec untuk mendapatkan fitur vektor sehingga kalimat yang dihasilkan dapat dilakukan pembelajaran mesin menggunakan RNN untuk mendapatkan nilai bobot dan hasil tersebut akan dilakukan pembangkitan kalimat ilmiah menggunakan LSTM. Sistem pembangkitan kalimat ilmiah dapat dilihat pada Gambar 1.



Gambar 1. Sistem Pembangkitan Kalimat Ilmiah

3.1 Perolehan Data

Perolehan data pada penelitian ini didapatkan dari kalimat ilmiah berbahasa inggris melalui makalah bidang komputasi ilmu komputer yang sudah tersedia dalam korpus. Kalimat yang diambil merupakan kalimat rujukan atau kalimat yang memiliki sitasi dan minimal teridiri dari tiga kata yaitu subjek, predikat dan objek atau keterangan. Jumlah dataset yang digunakan berjumlah 1000 kalimat ilmiah.

3.2 Praproses

Praproses merupakan tahapan awal dalam mengolah dataset maupun data uji yang akan digunakan untuk proses ekstraksi fitur dan pembangkitan. Tujuan dari praproses pada penelitian ini agar mempermudah dalam mencari nilai vektor dalam suatu kata yang nantinya akan menghasilkan bobot untuk diproses pada model RNN dan LSTM. Tahapan praproses tediri dari beberapa tahapan yaitu Text Cleaning, Case Folding, dan Tokeninzing.

3.2.1 Text Cleaning

Text Cleaning merupakan proses dalam pembersihan atau penghapusan teks atau karakter yang bersifat non-alfabetis (tidak sesuai dengan abjad) untuk mengurangi noise. Proses ini akan membersihkan sekumpulan simbol-simbol seperti titik (.), koma (,), kurung siku ([]), kurung buka (()) dan simbol atau karakter lainnya.

3.2.2 Case Folding

Case Folding merupakan proses Text Preprocessing untuk mengubah semua huruf kapital dalam dokumen menjadi huruf kecil. Hanya huruf ‘a’ sampai ‘z’ yang diterima untuk meningkatkan komputasi dalam pembelajaran.

3.2.3 Tokenizing

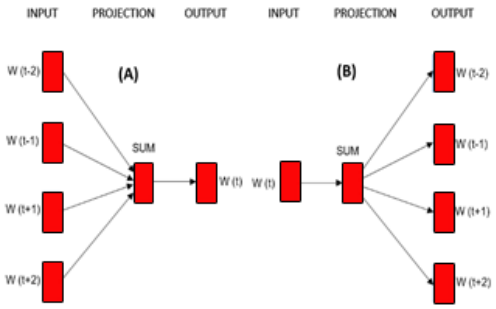
Tokenizing merupakan kalimat hasil dari Case Folding yang dipecah menjadi beberapa bagian kata. Pemecahan kalimat berdasarkan tanda spasi antar kalimat, sehingga dibuatlah

E-ISSN: 2655-867X
P-ISSN: 2655-8661
Volume 3 No. 01 Febuari 2021

list yang terdiri dari kumpulan kata yang disebut token.

3.3 Word2vec

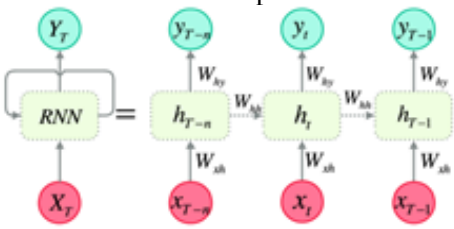
Word2vec merupakan metode yang dapat mengolah kata-kata dari korpus menjadi sebuah vektor. Word2vec juga merupakan *neural networks* yang digunakan untuk memproses teks sebelum teks tersebut diproses lebih lanjut oleh algoritma *deep-learnin* [15]. Metode ini bekerja dengan mengambil korpus teks sebagai input dengan melalui tahapan praproses dan *one hot-encoding*, lalu menghasilkan representasi kata yang ada pada korpus menjadi sebuah vektor output [16]. Word2vec memiliki dua jenis arsitektur pemodelan yang dapat digunakan untuk merepresentasikan vektor kata, arsitektur tersebut diantaranya yaitu CBOW dan Skip-gram seperti yang ditunjukan pada Gambar 2.



Gambar 1. (A) Model CBOW dan (B) Model Skip-Gram

3.4 Recurrent Neural Network

Recurrent Neural Network (RNN) merupakan jenis arsitektur jaringan saraf tiruan yang telah terbukti kinerjanya dengan baik. RNN merupakan sebuah model arsitektur jaringan saraf tiruan yang dimana pada prosesnya dipanggil secara berulang-ulang untuk memproses input data sekuensial. RNN berisi memori *internal* yang dapat digunakan untuk mengingat masukan sebelumnya serta masukan saat ini yang membuat tugas pemodelan urutan jauh lebih mudah [17]. Output pada setiap *time step* tidak hanya bergantung pada *input* saat ini tetapi juga pada *output* yang dihasilkan pada *time step* sebelumnya, yang membuatnya sangat mampu melakukan tugas-tugas seperti pembuatan teks. Dalam JST semua input tidak bergantung pada satu sama lain, tetapi dalam RNN ada ketergantungan antar *input*. Berikut merupakan arsitektur RNN pada Gambar 3.



Gambar 3. Arsitektur Recurrent Neural Network

RNN dapat menyimpan informasi dari masa lalu dengan melakukan *looping* di dalam arsitekturnya, yang secara otomatis membuat informasi dari masa lalu tetap tersimpan. Pada jaringan RNN menggunakan fungsi aktivasi *sigmoid* untuk *hidden layer*. Fungsi *sigmoid* memiliki output dengan rentang 0 sampai 1. Fungsi *sigmoid* dapat dilihat pada Persamaan 1. Dengan turunan fungsi terlihat pada Persamaan 2.

$$f(x) = \frac{1}{1+e^{-x}} \quad (1)$$

$$f'(x) = f(x)(1 - f(x)) \quad (2)$$

Terdapat inti perulangan pada RNN yang mengambil nilai input *x* kemudian dimasukkan ke dalam RNN yang berisi nilai dari *hidden layer* yang akan diperbarui setiap kali RNN membaca *input* baru sehingga menghasilkan *output* pada setiap waktu. Konfigurasi RNN memiliki hubungan perulangan dengan fungsi aktivasi yang disimbolkan oleh f sehingga fungsi tersebut akan bergantung pada bobot *w*. Bobot *w* akan menerima nilai state baru sebelumnya dari *hidden layer* dikurangi 1 yang menjadi masukan pada saat keadaan x_t dan disimpan ke dalam hidden layer berikutnya (h_t) atau dengan kata lain pada saat hidden layer diperbarui. Proses ini dilakukan menggunakan Persamaan 3.

$$h_t = f_w(h_{t-1}, x_t) \quad (3)$$

Nilai *x* dimasukan ke dalam fungsi aktivasi f dan bobot *w* yang sama pada setiap kali perhitungan. Secara sederhana terdapat matriks bobot W_{xh} yang dikalikan dengan input x_t serta matriks bobot lain W_{hh} yang dikalikan terhadap nilai dari *hidden layer* sebelumnya atau h_{t-1}. Kedua matriks tersebut ditambahkan. Jika terdapat data *non-linear* maka hasil

E-ISSN: 2655-867X
P-ISSN: 2655-8661
Volume 3 No. 01 Febuari 2021

penjumlahkan kedua matriks dikalikan dengan *tanh*, seperti pada Persamaan 4.

$$. h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \quad (4)$$

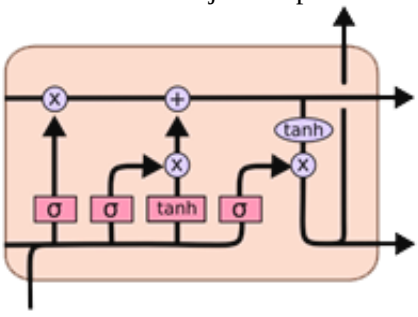
Nilai Pada arsitektur RNN jika ingin menghasilkan beberapa y_t di setiap waktu, dikarenakan terdapat matriks bobot lain *W* dari *hidden layer* W_h sehingga mengubah beberapa nilai *y* yang terlihat pada Persamaan 5.

$$y_t = W_{hy}h_t \quad (5)$$

Hasil pada setiap data yang telah dilakukan oleh praproses, maka tahapan pemodelan RNN dapat dilakukan. Semua data kalimat yang telah dilakukan proses *embedding* akan dijadikan sebagai *input* terhadap *neuron*. Jumlah *neuron* dihasilkan dari ukuran *window size* sebanyak 300 dan jumlah kata unik sebanyak 3102, sehingga menghasilkan jumlah *neuron* sebanyak 930.600.

3.5 Long Short Term Memory

Long Short Term Memory (LSTM) merupakan jenis modul pemrosesan lain untuk RNN. Saat ini LSTM telah menjadi salah satu model yang banyak digunakan pada Deep Learning untuk NLP. Pada penelitian sebelumnya LSTM mampu mengatasi masalah vanishing gradient pada RNN [18] menggunakan mekanisme gerbang (*gate*). LSTM merupakan cara lain untuk menghitung hidden state. Arsitektur RNN dengan konfigurasi LSTM ditunjukkan pada Gambar 4.



Gambar 4. Arsitektur RNN dengan konfigurasi LSTM

Pada Gambar 4 menjelaskan bagaimana alur kerja memory cells pada setiap *neuron* LSTM bekerja. Terdapat empat proses fungsi aktivasi pada setiap masukan pada *neuron* yang selanjutnya disebut sebagai *gates units*. *Gates units* tersebut ialah *forget gates*, *input gates*, *cell gates*, dan *output gates*. Pada *forget gates* informasi pada setiap data masukan akan diolah dan dipilih data mana saja yang akan disimpan atau dibuang pada *memory cells*. Fungsi aktivasi yang digunakan pada *forget gates* ini adalah fungsi aktivasi *sigmoid*. Dimana hasil keluarannya antara 0 dan 1. Jika keluarannya adalah 1 maka semua data akan disimpan dan sebaliknya jika keluarannya 0 maka semua data akan dibuang. Proses ini dilakukan menggunakan persamaan 6.

$$f = \sigma(x_t U_i + s_{t-1}W_f) \quad (6)$$

Pada *input gates* terdapat dua *gates* yang akan dilalukan, pertama akan diputuskan nilai mana yang akan diperbarui menggunakan fungsi aktivasi *sigmoid*. Selanjutnya fungsi aktivasi *tanh* akan membuat vektor nilai baru yang akan disimpan pada *memory cell*. Dengan persamaan 7 dan persamaan 8.

$$i = \sigma(x_t U_i + s_{t-1}W_i) \quad (7)$$

$$g = \tanh(x_t U_g + s_t W_g) \quad (8)$$

Pada *cell gates* akan mengganti nilai pada *memory cell* sebelumnya dengan nilai *memory cell* yang baru. Dimana nilai ini didapatkan dari menggabungkan nilai yang terdapat pada *forget gate* dan *input gate*. Dengan persamaan 9.

$$C_t = C_{t-1} \circ f + g \circ i \quad (9)$$

Pada *output gates* terdapat dua *gates* yang akan dilakukan, pertama akan diputuskan nilai pada bagian *memory cell* mana yang akan dikeluarkan dengan menggunakan fungsi aktivasi *sigmoid*. Selanjutnya ditempatkan nilai pada *memory cell* dengan menggunakan fungsi aktivasi tanh. Terakhir kedua *gates* tersebut dikalikan sehingga menghasilkan nilai yang dikeluarkan. Dengan persamaan 10, dan persamaan 11.

$$O = \sigma(x_t U_o + s_{t-1}W_o) \quad (10)$$

$$S_t = \tanh(C_t) \circ O \quad (11)$$

Setelah mendapatkan nilai output S_t, selanjutnya diberi fungsi aktivasi *softmax* untuk menghitung probabilitas, untuk fungsi aktivasi *softmax* dapat menggunakan Persamaan 12.

$$Softmax(y_j) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}} \quad (12)$$

Dimana:

y_j = Nilai output ke-j
K = Jumlah neuron lapisan output
k = Indeks neuron output

E-ISSN: 2655-867X
P-ISSN: 2655-8661
Volume 3 No. 01 Febuari 2021

$e^{x^T w_j}$ = nilai unit yang ke-j

$e^{x^T w_k}$ = nilai neuron yang ke-k

Selanjutnya adalah menghitung nilai *error* dari masing-masing output *softmax*, dilanjut dengan *backpropagation* dan *update* bobot. Proses tersebut dilakukan secara berulang sampai terpenuhi. Untuk pembangkitan kalimat proses yang dilakukan hanya sampai *feed forward* menggunakan bobot dari hasil pelatihan.

3.6 Gate Recurrent Unit

Gate Recurrent Unit (GRU) mirip dengan unit LSTM, GRU memiliki unit gerbang yang memodulasi aliran informasi di dalam unit, namun tanpa memiliki sel memori terpisah [19]. Model GRU terdiri dari dua gerbang, yaitu gerbang pembaruan z_t dan gerbang *reset* U_t.

Gerbang pembaruan menentukan berapa banyak memori sebelumnya yang harus disimpan, ini dilambangkan dalam persamaan 13. Gerbang *reset* menentukan bagaimana menggabungkan input baru dengan memori sebelumnya, dihitung dengan persamaan 14. Lapisan tersembunyi dihitung dengan persamaan 15 menggunakan H_t yang dihitung dengan persamaan 16.

$$z_t = \sigma\ (W_z x_t + U_z h_{t-1} \quad (13)$$

$$r_t = \sigma\ (W_r x_r + U_r h_{t-1} \quad (14)$$

$$H_t = tanh(W_H x_t + U_H(r_t h_{t-1})) \quad (15)$$

$$h_t = (1 - z_t) h_{t-1} + z_t H_t \quad (16)$$

Parameter x_t adalah input pada satu waktu, dan matriks bobot dilambangkan dengan $W_t, W_t, W_h, U_z, U_r, U_h$.

Nilai yang diperoleh dari gerbang keluaran diubah menjadi probabilitas vocab menggunakan fungsi aktivasi Softmax, yang dilambangkan dalam persamaan 17.

$$y_k = \frac{e^{x^T w_k}}{\sum_{i=1}^{m} e^{x^T w_i}} \quad (17)$$

Nilai Langkah selanjutnya adalah menghitung kesalahan pada lapisan keluaran menggunakan cross-entropy, yang ditunjukkan pada persamaan 18.

$$D(S, L) = -\sum_i L_i log(S_i) \quad (18)$$

Dimana D adalah jarak, S adalah hasil dari nilai Softmax, dan L adalah *vocabularry*.

3.7 Bilingual Evaluation Understudy

Bilingual Evaluation Understudy (BLEU) merupakan sebuah algoritma yang berfungsi untuk mengevaluasi kualitas dari sebuah hasil terjemahan yang telah ditejemahkan oleh mesin dari satu bahasa alami ke bahasa lain. BLEU mengukur modified N-Gram *precission score* antara hasil *candidate* dengan kalimat rujukan dan menggunakan konstanta yang dinamakan *brevity penalty*.

Brevity Penalty digunakan untuk mencegah kalimat pendek memperoleh nilai yang tinggi. Dengan kata lain, BLEU diperoleh dari hasil perlkalian antara *brevity penalty* dengan rata-rata geometri dari *modified precission score*. Jadi agar dapat menghasilkan nilai yang tinggi dalam BLEU, panjang kalimat hasil terjemhan harus dapat menghasilkan beberapa kata serta urutan yang sama dengan kalimat referensi. Nilai dari BLEU berada dalam rentang 0 sampai 1. Semakin tinggi nilai BLEU, maka semakin baik akurat dengan rujukan. Berikut adalah Persamaan 19, Persamaan 20 dan Persamaan 21 dari BLEU.

$$BP_{BLEU} \begin{cases} 1, & if\ c > r \\ e^{\left(1-\frac{r}{c}\right)}, & if\ c \leq r \end{cases} \quad (19)$$

$$P_n = \frac{\sum_{Ce\ corpus\ n-gram\ eC} \sum count_{clip(n-gram)}}{\sum_{Ce\ corpus\ n-gram\ eC} \sum count_{clip(n-gram)}} \quad (20)$$

$$BLEU = BP_{BLEU} . e^{\sum_{n-1}^{N} W_n log P_n} \quad (21)$$

Keterangan:

BP : *brevity penalty*

c : jumlah kata dari hasil terjemahan

r : jumlah rujukan

P_N : *modified precission score*

W_N : 1/N (standar nilai N untuk BLEU adalah 4)

p_N : Jumlah n-gram hasil terjemahan yang sesuai dengan rujukan dibagi jumlah n-gram hasil terjemahan

Pada penelitian terdahulu yaitu berisi tentang analisis terhadap nilai akurasi mesin penerjemah statistik terhadap korpus paralel bahasa Arab dan bahasa Indonesia yang bersumber dari alquran dan hadits nabi, untuk menguji tingkat akurasi pada terjemahan menggunakan metode BLEU [20]. Pada

E-ISSN: 2655-867X
P-ISSN: 2655-8661
Volume 3 No. 01 Febuari 2021

penelitian tersebut menghasilkan nilai BLEU terhadap korpus alquran sebesar 10,56, korpus hadits sebesar 27,65, dan korpus gabungan sebesar 15,41.

4. HASIL DAN PEMBAHASAN

Penelitian ini menggunakan data pelatihan yang telah dilakukan praproses pada kalimat ilmiah. Pelatihan dan pengujian menggunakan *library* Keras dengan Python 3.0. Korpus *dataset* akan diekstraksi dengan praproses untuk mendapatkan nilai vektor menggunakan Word2vec. Vektor tersebut digunakan dalam proses pelatihan RNN untuk mendapatkan nilai bobot sebagai lapisan *input* dengan kofigurasi LSTM dan GRU. Pengujian terhadap sistem yang dibangun menggunakan algoritma BLEU.

Pada pengujian ini kalimat rujukan sebagai *input* dan kalimat kandidat sebagai *output* hasil pengujian pembangkitan kalimat ilmiah. Hasil pengujin dapat dilihat pada Tabel 1 dan Tabel 2, kalimat yang dihasilkan oleh sistem yang telah dibangun.

Tabel 1. Hasil Pembangkitan Kalimat Terbaik pada Sistem.

No.	Kalimat Rujukan	Kalimat Kandidat
1	koehn et al 2007 we use the moses software package 5 to train a pbmt model koehn et al 2007	we build a state of the art phrasebased smt system using moses koehn et al 2007 we use the moses
2	koehn et al 2007 we build a state of the art phrasebased smt system using moses koehn et al 2007	we use the moses software package 5 to train a pbmt model koehn et al 2007 we build a state
…	…	…
1000	of the art phrasebased smt system using moses koehn et al 2007 we use the moses software package 5 to	train a pbmt model koehn et al 2007 we build a state of the art phrasebased smt system using mose

Tabel 2. Hasil Pembangkitan Kalimat Terburuk pada Sistem.

No.	Kalimat Rujukan	Kalimat Kandidat
1	use the moses toolkit koehn et al 2007 we evaluate our method on the following data sets bullet ontonotesdev development	set of the ontonotes data provided by the conll2012 shared task pradhan et al 2012 bullet ontonotestest test set of
2	the ontonotes data provided by the conll2012 shared task pradhan et al 2012 we use the moses phrasebased translation system	koehn et al 2007 to implement our modelswe use the stateoftheart phrasebased machine translation system and scikitlearn pedregosa et al
…	…	…
1000	moses toolkit koehn et al 2007 for the phrasebased smt system we adopted the moses toolkit koehn et al 2007	5gram language models of turkish and english were trained using kenlm heafield 2011 we built a 5gram language model on

Pada kalimat rujukan dan kalimat kandidat akan dievaluasi menggunakan score BLEU yang merangkum seberapa dekat antara kalimat rujukan dengan kalimat kandidat yang dihasilkan dengan hasil yang diharapkan. Untuk masing masing skor BLEU ditambahkan bobot untuk setiap skor BLEU dan kemudian diproses menggunakan library Python NLTK. Berikut merupakan hasil skor BLEU yang dihasilkan dari sistem yang telah dibangun menggunakan model optimasi yaitu Adaptive Moment Estimation (Adam), Nesterov-accelerated Adaptive Moment Estimation (Nadam) dan AdaMax yang dapat dilihat pada Tabel 3 dan Tabel 4.

E-ISSN: 2655-867X
P-ISSN: 2655-8661
Volume 3 No. 01 Febuari 2021

Tabel 3. Hasil Skor BLEU dengan GRU

optimization	Epoch	Hasil Skor BLEU
Adam	100	21
	200	19
	300	18
Adamax	100	19
	200	**28**
	300	17
Nadam	100	20
	200	19
	300	18

Tabel 4. Hasil Skor BLEU dengan LSTM

optimization	Epoch	Hasil Skor BLEU
Adam	100	20
	200	20
	300	24
Adamax	100	21
	200	**25**
	300	17
Nadam	100	18
	200	15
	300	18

Pada Hasil skor BLEU menggunakan model GRU yang terlihat pada Tabel 3 dapat dilihat bahwa untuk optimasi AdaMax skor BLEU mendapatkan skor mencapai 28, sedangakan untuk optimasi Adam skor BLEU mendapatkan skor mencapai 21, sedangkan untuk optimasi Nadam skor BLEU mendapatkan skor mencapai 20. Pada hasil skor BLEU menggunakan model LSTM yang terlihat pada Tabel 4 dapat dilihat bahwa untuk optimasi AdaMax skor BLEU mendapatkan skor mencapai 25, sedangakan untuk optimasi Adam skor BLEU mendapatkan skor mencapai 24, sedangkan untuk optimasi Nadam skor BLEU mendapatkan skor mencapai 18. Pengujian terhadap *unigram* cenderung memberikan nilai BLEU yang lebih besar dibandingkan dengan *bigram*, hal ini menunjukan bahwa pendekatan ini bekerja dengan menghitung pencocokan unigram dalam terjemahan kandidat ke *unigram* dalam teks referensi. Di mana unigram akan menjadi setiap token dan perbandingan *bigram* akan menjadi setiap pasangan kata, perbandingan ini terlepas dari urutan kata.

5. KESIMPULAN

Kesimpulan Penelitian ini menghasilkan sistem pembangkitan kalimat ilmiah menggunakan Recurrent Neural Network. Dataset yang digunakan pada penelitian ini adalah korpus kalimat ilmiah dari beberapa kalimat kutipan dibidang ilmu komputer. Sebelum melakukan pelatihan, dilakukan praproses untuk mengurangi *noise*, tahapannya mulai dari pembersihan karakter non-*alphanumeric*, lalu dilanjut dengan mengubah setiap *upper case* menjadi *lower case*, selanjutnya masuk ke dalam proses tokenizing yang memisahkan setiap kata unik menjadi token yang nantinya akan digunakan sebagai masukan dalam ekstraksi fitur Word2vec dengan model CBOW. Hasil *embbeding* dengan Word2vec menghasilkan 300 *feature* vektor dari setiap kata yang ada dalam korpus. Vektor kata tersebut yang nanti akan digunakan dalam proses pelatihan menggunakan Recurrent Neural Network kofigurasi LSTM dan GRU, dengan pengaturan yang sama yaitu *learning rate* sebesar 0,001, *bacth size* sebesar 128, *dropout* sebesar 0.2.

Pengujian selanjutnya menggunakan BLEU skor pada ketiga optimasi yaitu Adaptive Moment Estimation (Adam), Nesterov-accelerated Adaptive Moment Estimation (Nadam) dan AdaMax. Untuk optimasi AdaMax mendapatkan skor pertama tertinggi dengan BLEU skor mencapai 28. Sedangkan untuk optimasi Adam mendapatkan skor kedua tertinggi setalah AdaMax dengan BLEU skor mencapai 24. Sedangkan untuk optimasi Nadam mendapatkan skor ketiga setelah Adam dengan BLEU skor mencapai 20. Hasil untuk optimasi Nadam menunjukan bahwa kualitas hasil terjemahan dari sistem belum mampu dalam menghasilkan kalimat yang direkomendasikan. Sedangkan untuk optimasi Adam dan Adamax menunjukan bahwa kualitas hasil terjemahan dari sistem sudah mampu baik dalam menghasilkan kalimat yang direkomendasikan.

E-ISSN: 2655-867X
P-ISSN: 2655-8661
Volume 3 No. 01 Febuari 2021

6. REFERENSI

[1] D. Jurgens, S. Kumar, R. Hoover, D. McFarl and, and D. Jurafsky, "Measuring the Evolution of a Scientific Field through Citation Frames," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 391–406, 2018.

[2] Y. Ding, G. Zhang, T. Chambers, M. Song, X. Wang, and C. Zhai, "Content-based citation analysis: The next generation of citation analysis," *Journal of the Association for Information Science and Technology*, vol. 65, no. 9, pp. 1820–1833, 2014.

[3] A. Cohan and N. Goharian, "Scientific article summarization using citation-context and article's discourse structure," *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, pp. 390–400, 2015.

[4] B. Gipp, N. Meuschke, and C. Breitinger, "Citation-based plagiarism detection: Practicability on a large-scale scientific corpus," *Journal of the Association for Information Science and Technology*, vol. 65, no. 8, pp. 1527–1540, 2014.

[5] A. Hassan and A. Mahmood, "Deep learning for sentence classification," *2017 IEEE Long Island Systems, Applications and Technology Conference, LISAT 2017*, no. May, 2017.

[6] X. Guo, H. Zhang, H. Yang, L. Xu, and Z. Ye, "A Single Attention-Based Combination of CNN and RNN for Relation Classification," *IEEE Access*, vol. 7, no. c, pp. 12467–12475, 2019.

[7] E. Clark, Y. Ji, N. A. Smith, P. G. Allen, and C. Science, "Neural Text Generation in Stories Using Entity Representations as Context," pp. 2250–2260, 2018.

[8] Y. Luo and Y. Huang, "Text Steganography with High Embedding Rate," pp. 99–104, 2017.

[9] R. Sunil, V. Jayan, and V. K. Bhadran, "Preprocessors in NLP applications: In the context of English to Malayalam machine translation," *2012 Annual IEEE India Conference, INDICON 2012*, pp. 221–226, 2012.

[10] K. Luu, R. Koncel-kedziorski, K. Lo, I. Cachola, and N. A. Smith, "Citation Text Generation," 2019.

[11] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [Review Article]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[12] S. Santhanam, "Context based Text-generation using LSTM networks," 2020.

[13] D. Pawade, A. Sakhapara, M. Jain, N. Jain, and K. Gada, "Story Scrambler - Automatic Text Generation Using Word Level RNN-LSTM," *International Journal of Information Technology and Computer Science*, vol. 10, no. 6, pp. 44–53, 2018.

[14] T. Fujita, W. Bai, and C. Quan, "Long short-term memory networks for automatic generation of conversations," *Proceedings - 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2017*, pp. 483–487, 2017.

[15] M. . Nabila Nanda Widyastuti, Arif Bijaksana, Ir., M.Tech., Ph.D, Indra Lukmana Sardi, S.T., "Analisis Word2vec untuk Perhitungan Kesamaan Semantik

E-ISSN: 2655-867X
P-ISSN: 2655-8661
Volume 3 No. 01 Febuari 2021

antar Kata," *e-Proceeding of Engineering*, vol. Vol.5, No., no. 3, pp. 7603–7612, 2018.

[16] T. Mikolov, W. Yih, and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations," no. June, pp. 746–751, 2013.

[17] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, no. March, pp. 1–43, 2020.

[18] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language processing," *Interspeech 2012*, pp. 194–197, 2012.

[19] D. Carlson, Y. P. Hsieh, E. Collins, L. Carin, and V. Cevher, "Stochastic Spectral Descent for Discrete Graphical Models," *IEEE Journal on Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 296–311, 2016.

[20] H. Ardhi, H. Sujaini, and A. B. Putra, "Analisis Penggabungan Korpus dari Hadits Nabi dan Alquran untuk Mesin Penerjemah Statistik," *Jurnal Linguistik Komputasional (JLK)*, vol. 1, no. 1, p. 31, 2018.