Klasifikasi Kalimat Ilmiah Menggunakan Recurrent Neural Network

Muhamad Rizal Firmansyah[1], Ridwan Ilyas[2], Fatan Kasyidi[3]

[1]*Jurusan Teknik Informatika, Universitas Jenderal Ahmad Yani, Cimahi* 40285
E-mail :[1] *muhamad.rizalfirmansyah@unjani.student.ac.id*
[2]*Jurusan Teknik Informatika dan Elektro, Institut Teknologi Bandung, Bandung 40132*
E-mail : rdwnilyas@gmail.com
[3]*Jurusan Teknik Informatika, Universitas Jenderal Achmad Yani, Bandung 40531*
fatan.kasyidi@lecture.unjani.ac.id

ABSTRAK

Pengklasifikasian hanya berbentuk satu kalimat ilmiah tunggal dan tidak terpengaruh oleh kalimat sebelum dan sesudah tetapi hanya berfokus pada satu kalimat ilmiah saja. Recurrent Neural Network (RNN) adalah 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 klasifikasi kalimat menggunakan RNN, dengan fitur yang telah diekstraksi menggunakan fungsi Word2Vec untuk menghasilkan satu set vektor. Dalam melakukan proses klasifikasi penelitian ini menggunakan total 2019 data pelatihan kalimat ilmiah yang telah dilabeli menjadi empat kelas, yaitu Weak, Comparison, Point, dan Neutral. Penelitian ini telah dibandingkan dengan empat optimasi yaitu Adam, SGD Adadelta, dan Adamax untuk menemukan tingkat pembelajaran terbaik dan cocok untuk klasifikasi kalimat. Hasil tingkat pembelajaran terbaik diperoleh dengan pengoptimalan SGD dengan nilai akurasi 77,48% dan Loss 0,71%**.** SGD tidak menggunakan banyak memori Gradient Descent sehingga konvergen lebih cepat. Selain itu SGD bekerja dengan memilih data sampel acak dari satu atau beberapa bagian dari data pelatihan dalam satu iterasi dengan cara yang iteratif. Data sampel acak ini dikoreksi berdasarkan aturan yang melibatkan gradien pertama untuk mengukur perubahan fungsi bersama dengan perubahan nilai input. Juga, hasil akurasi percobaan ini menunjukkan bahwa skor *F-Measure* mencapai 39,5%.

Kata Kunci
Kalimat; Klasifikasi; Word2vec; Recurrent Neural Network, Long Short Term Memory, Optimasi

1. PENDAHULUAN

Kalimat merupakan satuan bahasa terkecil yang mengungkapkan suatu gagasan baik secara lisan ataupun tulisan. Sebuah kalimat dibuat untuk menyampaikan pesan atau informasi yang tepat. Agar pesan yang ingin disampaikan ini mudah dimengerti sehingga tidak menimbulkan kesalahpahaman. Menurut ilmu pengetahuan definisi dari klasifikasi adalah mengelompokkan suatu objek menjadi beberapa indikator yang didasarkan pada ciri-ciri atau karakteristik yang dimilikinya. Sedangkan klasifikasi kalimat adalah sebuah kalimat yang topik utamanya dikembangkan dengan mengelompokanya ke dalam beberapa kelompok berdasarkan kelas yang telah ditentukan sebelumnya. Kalimat rujukan yang tertulis dalam karya tulis ilmiah dapat dianalisis sebagai acuan untuk melihat arti dari kalimat rujukan tersebut, para peneliti dalam mengembangkan argumen mereka ditinjau dari fungsi kutipan yang telah dipetakan oleh kelas-kelas yang didefinisikan [1]. Setiap karya ilmiah mempunyai proses analisis *bibliometric* dan ketika memiliki ribuan kalimat sitasi [2] maka hal tersebut dapat diatasi oleh model klasifikasi kalimat agar mempermudah proses analisis *bibliometric*. Klasifikasi merupakan proses penggolongan atau pengelompokan suatu objek yang telah diberi label atau kelas yang telah ditentukan sebelumnya [3].

Natural Language Processing (NLP) dapat membuat mesin yang bisa memahami stuktur makna bahasa manusia, selain itu dalam beberapa penelitian lain klasifikasi kalimat merupakan bagian dari NLP. Dengan hasil tersebut NLP memberikan respon yang sesuai berdasarkan interaksi antara komputer dan bahasa alami manusia sehingga metode ini telah mencapai hasil yang baik dalam implementasinya mencapai akurasi sebesar 72% [4]. Beberapa penelitian terdahulu NLP dapat bekerja dalam Klasifikasi Analisis Sentimen Ulasan Film [5], Respon Crisis pada Twitter [6], Judul Berita [7],

Relasi Kata dan Kalimat [8]. Dengan hasil tersebut NLP memberikan respon yang sesuai berdasarkan interaksi antara komputer dan bahasa alami manusia sehingga metode ini telah mencapai hasil yang baik dalam implementasinya [5].

Hasil penelitian sebelumnya hanya berfokus pada pengembangan *annotation scheme* dengan metode *linguistic analysis* untuk klasifikasi kalimat sitasi secara otomatis yang ditinjau dari perilaku pengutip terhadap karya ilmiah yang dikutip dan dibedakan menjadi 12 kelas : *Weak*, *CoCoGM*, *CoCo-*, *CoCoR0*, *CoCoXY*, *PBas*, *PUse*, *PModi*, *PMot*, *PSim*, *Psup* dan *Neut* [9]. Sehingga dalam penelitian ini perlu adanya pendekatan kembali dalam melakukan anotasi ulang untuk membentuk *corpus* kalimat yang baru dengan proses pelabelan data menggunakan kelas yang sama. Selain itu terdapat juga penelitian yang mengimplementasikan metode *annotation scheme* yang bertujuan untuk melakukan klasifikasi kalimat retoris dari karya ilmiah yang dibedakan menjadi 12 bagian : *Weak*, *CoCoGM*, *CoCo-*, *CoCoR0*, *CoCoXY*, *PBas*, *PUse*, *PModi*, *PMot*, *PSim*, *Psup* dan *Neut* [10]. Hasil yang didapatkan pada penelitian tersebut menghasilkan *akurasi F-Measure* sebesar 43,44% dengan menggunakan dataset makalah sebanyak 20.155 [10]. Dan dari hal tersebut terdapat peluang untuk mengembangkan model komputasi yang dapat melakukan klasifikasi kalimat agar mendapatkan hasil pengelompokan kalimat berdasarkan kelas yang sesuai.

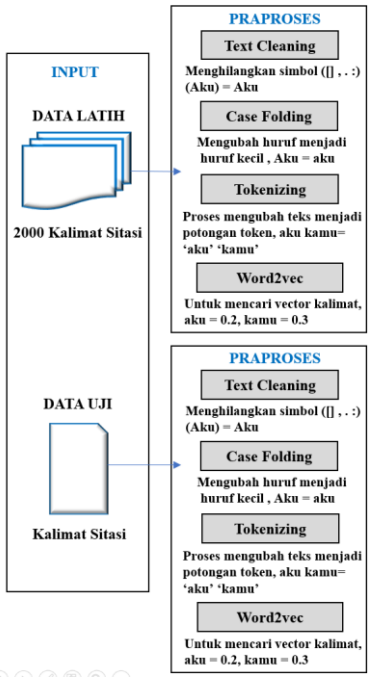
Pada penelitian lain dapat melakukan klasifikasi kalimat menggunakan dataset SST-2 (Analisis Sentimen Ulasan Film) namun dataset ini mengalami hasil yang kurang baik karena disebabkan oleh beberapa lapisan *layer* yang dapat membagikan *output* dari satu *layer* (*shared layer)* sehingga membuat data kalimat menjadi tidak terstruktur (*complicated sentence*) [11]. Pada penelitian tersebut mengalami ketidakcocokan data pada dataset yang digunakan sehingga terdapat kesalahan pada proses klasifikasi. Kesalahan kalimat yang tidak terstruktur dapat diperbaiki dengan perancangan praproses yang akan digunakan menggunakan Text Cleaning, Case Folding, Tokenizing, dan Word2vec agar menjadi kalimat yang terstukur.

RNN adalah jenis arsitektur jaringan saraf tiruan yang pemrosesannya berulang kali dipanggil untuk memproses *input* data sekuensial. RNN termasuk dalam kategori Deep Learning karena data diproses melalui banyak lapisan. RNN telah mengalami kemajuan pesat dan telah merevolusi bidang-bidang seperti NLP [11]. Long Short Term Memory merupakan bagian dari arsitektur RNN yang merupakan jaringan berulang. LSTM akan lebih mudah dalam mempelajari pola yang diberikan dan LSTM digunakan untuk memodelkan isi bobot dari jaringan sehingga proses klasifikasi akan lebih mudah [12].

Akhir bagian pendahuluan memuat rumusan singkat tentang hal-hal pokok yang akan dibahas. Penelitian ini telah membangun sistem yang dapat melakukan klasifikasi kalimat secara otomatis menggunakan Word2vec model CBOW dan RNN. Tahapan yang dilakukan adalah praproses kalimat, mengubah kalimat menjadi vektor dengan Word2vec, dan tahapan proses klasifikasi menggunakan RNN. Proses klasifikasi terbagi menjadi empat kategori kelas yaitu Weak, Comparison, Point dan Neutral.

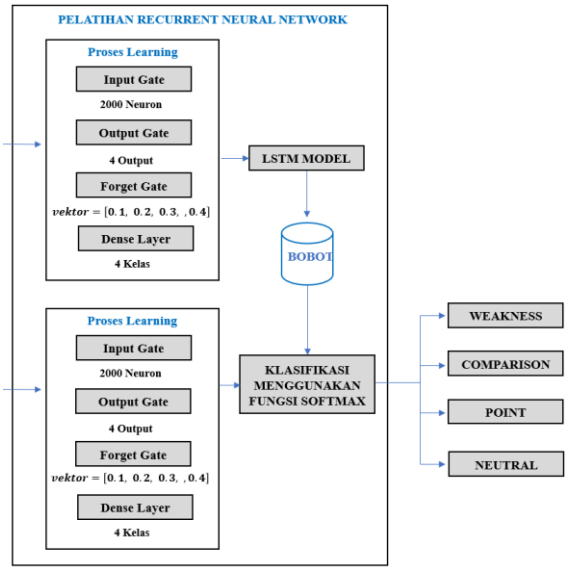
2. METODOLOGI PENELITIAN

Tahap penelitian dimulai dengan mendapatkan *corpus* dataset kalimat ilmiah sebagai subjek yang digunakan untuk melatih data. Data pada sistem klasifikasi 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 didapat menjadi terstruktur. Hasil dari ekstraksi fitur selanjutnya dilakukan pembelajaran mesin menggunakan RNN untuk mendapatkan nilai bobot dan hasil tersebut akan dilakukan klasifikasi menggunakan LSTM. Sistem klasifikasi kalimat ilmiah dapat dilihat pada Gambar 1



Gambar 1 Model Klasifikasi Kalimat Ilmiah

Setelah dilakukannya proses praproses pada data latih dan data uji, hasil dari praproses tersebut akan diproses pelatihan menggunakan RNN-LSTM, dapat dilihat pada gambar 2.



Gambar 2 Model Pelatihan Menggunakan RNN

2.1 Studi Pustaka

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

2.2 Perolehan Data

Perolehan data diambil dari kalimat rujukan atau kalimat yang memiliki sitasi dan minimal terdiri tiga kata yaitu subjek, predikat dan objek atau keterangan pada paper komputasi berbahasa Inggris, Selanjutnya *corpus* dataset tersebut dianotasi ulang agar dapat digunakan untuk memenuhi tujuan dari penelitian dengan menghasilkan pengelompokan kalimat sitasi berdasarkan kelas yang telah ditentukan sebelumnya yang terbagi menjadi empat kategori kelas yaitu, *Weak*, *Comparison*, *Point*, dan *Neutral*. Corpus dataset pada penelitian ini sebanyak 2019 kalimat sitasi, data tersebut dibagi menjadi 2 bagian yaitu data latih dan data uji.

2.3 Praproses

Praproses merupakan tahapan awal dalam memproses input data sebelum melewati proses klasifikasi. Tujuan dari praproses pada penelitian ini agar mempermudah dalam mencari nilai vektor yang nantinya akan menghasil bobot untuk diproses pada model RNN dan LSTM.

2.3.1 Text Cleaning

Text Cleaning adalah langkah utama dalam proses 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.

2.3.2 Case Folding

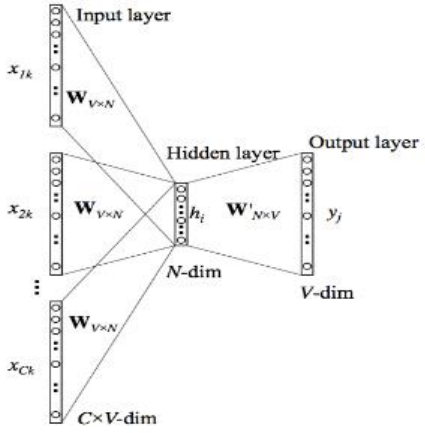
Case folding adalah salah satu bentuk *text preprocessing* yang paling sederhana dan efektif meskipun sering diabaikan. Tujuan dari case folding untuk mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf ‘a’ sampai ‘z’ yang diterima.

2.3.3 Tokenizing

Pada bagian Tokenizing, kalimat hasil dari Case Folding di pecah menjadi beberapa bagian kata. Pemecahan kalimat berdasarkan tanda spasi antar kalimat, sehingga dibuatlah *list* yang terdiri dari kumpulan kata yang disebut token.

2.3.4 Word2Vec

Word2Vec adalah algoritma *embedding* dan jaringan saraf dua lapis yang digunakan untuk memproses pemetaan kata dan mencari nilai vektor kalimat. Panjang musukan kalimat ditentukan dengan 40 kata dalam setiap kalimat, atau biasa disebut *padding*. Apabila kata dalam kalimat kurang dari 40, maka akan diberi nilai 0 untuk melengkapi *padding* 40. Apabila kata dalam kalimat lebih dari 40, maka akan diambil 40 kata dari awal kalimat.



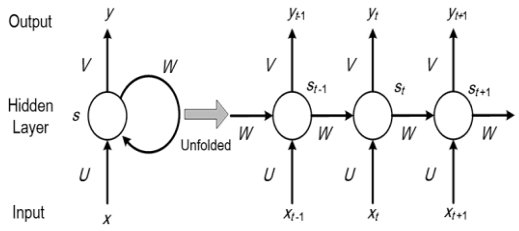
Gambar 3 Word2Vec Model Continous Bag-of-Word

Word2Vec pada penelitian ini menggunakan model CBOW. CBOW bekerja pada model yang memprediksi kata yang diberikan konteks, konteks dapat berupa kalimat atau sekelompok kata. Masukan dari CBOW adalah kedekatan kata, yaitu kata sebelum

dan sesudah, Tujuan dari kedekatan kata adalah pemberian nilai bobot.

2.4 Recurrent Neural Network

RNN dalam penelitian ini berfungsi untuk melakukan proses *input* data secara sekuensial. Dalam tiap pemrosesan, *output* yang dihasilkan tidak hanya merupakan fungsi dari sampel itu saja, tetapi juga berdasarkan *state internal* yang merupakan hasil dari pemrosesan sampel-sampel sebelumnya (atau setelahnya, pada *bidirectional* RNN) [13]. Pemodelan RNN dapat menyelesaikan berbagai tugas kategorisasi kalimat dan dapat melakukan klasifikasi [14], karena kemampuan dalam memprosesnya dipanggil berulang-ulang dengan hasil dapat menangani *input* dan *output variable* yang panjangnya bervariasi. Pada intinya RNN adalah jaringan syaraf tiruan yang menggunakan rekurensi dengan memanfaatkan data masa lalu. Karena itu, beberapa studi terbaru mengenai RNN cukup kuat untuk permasalahan klasifikasi. RNN memiliki arsitektur yang dapat digunakan untuk data berbentuk sekuensial seperti yang dapat dilihatdilihat pada Gambar 3



Gambar 4 Recurrent Neural Network

Cara yang dilakukan RNN untuk dapat menyimpan informasi dari masa lalu adalah 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. Pada komputasi menggunakan RNN terdapat 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 penjumlahkan kedua matriks dikalikan dengan *tanh*, seperti pada Persamaan 4.

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

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.

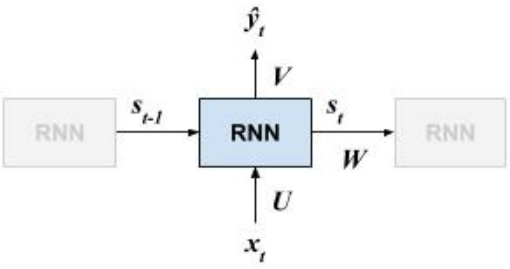
$$y_t = W_{hy}h_t \quad (5)$$

Proses pelatihan pada RNN menggunakan algoritma Backpropagation dengan beberapa putaran. Karena parameter yang dibagikan secara merata pada setiap *time step*, maka *gradient* untuk setiap *output* tergantung tidak hanya pada kalkulasi dari *time step* saat ini, tetapi juga pada *time step* sebelumnya. Pada arsitektur RNN terdapat beberapa unit seperti Gate Recurrent Unit (GRU), Backpropagtaion Through Time (BPTT) dan Long Short-Term Memory (LSTM). 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 100 dan jumlah kata unik sebanyak 2696, sehingga menghasilkan jumlah *neuron* sebanyak 495.204.

2.5 Long Short Term Memory

LSTM secara fundamental tidak memiliki perbedaan arsitektur dari RNN, LSTM memiliki fungsi untuk mengkomputasi *hidden state* yang dapat merekam *long-term dependencies* (ketergantungan jangka pajang) selain itu LSTM adalah arsitektur jaringan saraf yang cukup baik untuk memproses data sekuensial. Dalam melakukan klasifikasi dibutuhkan

proses *forward propagation* terlebih agar fungsi aktivasi *softmax* dapat dilakukan.



Gambar 5 Modul Recurrent Neural Network

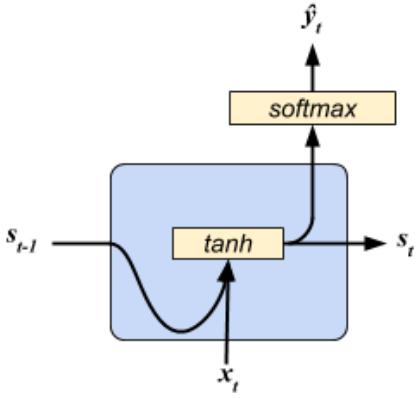
Untuk setiap langkah waktu *t*, pertama harus mengkalkulasi *state* s_t dari input (x_t) dan *state* sebelumnya (s_{t-1}), masing-masing dikalikan dengan parameter *U* dan *W* lalu diproses dengan fungsi aktivasi *tanh* dilihat pada persamaan 6.

$$s_t = tanh(U \cdot x_t + W \cdot s_{t-1}) \quad (6)$$

Dari s_t kemudian dikalkulasi output $\hat{y}_t$ dengan cara mengalikan dengan parameter *V* dan melewatkan pada fungsi aktivasi *softmax* dilihat pada persamaan 7.

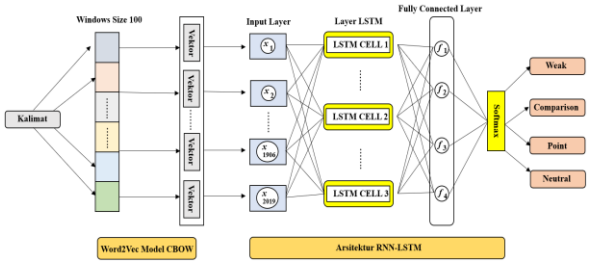
$$\hat{y}_t = softmax(V \cdot s_t) \quad (7)$$

Proses tersebut sering divisualisasikan pada gambar 5.



Gambar 6 Visualisasi Fungsi Aktivasi Softmax

Keterangan :
s_t : *Current State*
s_{t-1} : *State* Sebelumnya
tanh : fungsi aktivasi tangent dengan rentang (-1,1)
x_t : *Input*
W : Bobot
$\hat{y}_t$: *Output*
V : Parameter



Gambar 7 Perancangan Klasifikasi Kalimat Ilmiah

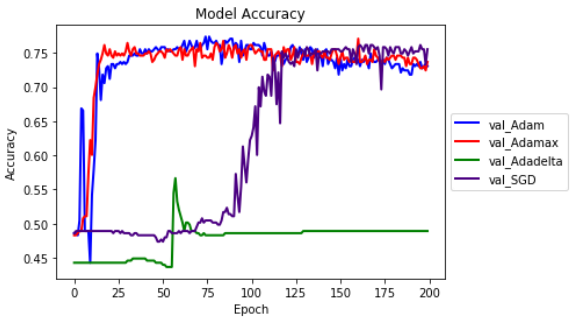
3. HASIL PENELITIAN

Penelitian ini menggunakan data pelatihan yang telah dilakukannya proses pelabelan kalimat berdasarkan empat pengelompokan kelas. Setiap kalimat terdiri dari *Weak*, *Comparison*, *Point*, dan *Neutral*. Data dibagi menjadi 2 bagian, 80% untuk pelatihan 20% untuk pengujian. Pelatihan dan pengujian menggunakan library Keras dengan python 3.0. Corbus dataset akan diekstraksi dengan praproses untuk mendapatkan nilai vektor. Vektor tersebut akan dilakukan proses pelatihan dengan RNN untuk mendapatkan nilai bobot sebagai lapisan *input* dari LSTM. Selanjutnya, 3 lapisan yaitu *input*, *fully connected layer*, dan *output layer* yang terhubung digunakan untuk menyaring hasil *cell* LSTM.

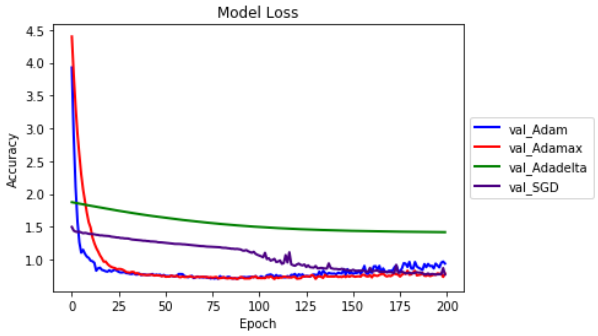
Tabel 1 Hasil Pengujian Model Optimasi

Optimasi	EPOCH	Data Latih		Data Uji	
		Akurasi (%)	*Loss*	*Akurasi (%)*	*Loss*
ADAM	100	80,37	0,58	75,50	0,70
	200	81,24	0,56	74,26	0,78
ADAMAX	100	79,26	0,61	76,49	0,67
	200	80,68	0,56	76,49	0,69
ADADELTA	100	52,23	1,44	54,71	1,44
	200	51,76	1,37	51,73	1,37
SGD	100	57,96	1,07	57,92	1,07
	200	78.27	0,68	77,48	0,71

Akurasi untuk data uji paling tinggi ditunjukkan pada gambar 8 dan loss data uji paling rendah ditunjukkan Gambar 9.



Gambar 8 Perbandingan Akurasi Data Uji



Gambar 9 Perbandingan Loss Data Uji

Dalam proses meningkatkan parameter pembelajaran klasifikasi dari ke empat optimasi yang digunakan masih fluktuasi dan tidak asimtotik, Optimasi Adam dan Adamax dapat melakukan pembelajaran lebih cepat dan terlihat bahwa perubahan nilai Loss lebih signifikan pada awal pembelajaran menuju konvergensi. Adam dan Adamax efisien secara komputasi dan memiliki minimal *requirement memory*. Namun pada optimasi Adadelta mengalami ketidakcocokan dalam proses klasifikasi yang disebabkan optimasi ini lebih condong terhadap metode penurunan *gradien stochastic* yang didasarkan pada tingkat pembelajaran adaptif perdimensi untuk mengatasi dua kelemahan:

1. Tingkat *decay of learning rate* yang terus menerus selama pelatihan
2. Kebutuhan akan tingkat *learning rate* yang dipilih secara manual

Model optimisasi SGD menghasilkan akurasi yang lebih baik daripada Adam dan Adamax. Akurasi yang diperoleh oleh Optimimasi SGD adalah 77,48%, Adam mendapatkan akurasi 75,50%, sedangkan Adamax mendapatkan akurasi 76,49%.
Hasil akurasi dan Loss per epoch ditunjukkan pada Tabel 2.

Tabel 2 Perbandingan Hasil per Epoch Akurasi dan Loss Menggunakan Optimasi SGD

No	Epoch	Data Latih		Data Uji	
		Akurasi (%)	***Loss***	***Akurasi (%)***	***Loss***
1	50	53,13	1,22	54,95	1,22
2	100	70,40	0,97	71,78	0,98
3	150	77,96	0,76	76,73	0,77
4	200	78.27	0,68	77,48	0,71

Model optimisasi SGD menunjukkan kinerja terbaik dalam melakukan proses klasifikasi kalimat. Tabel 3 menunjukkan bahwa kelas *Point* dan *Neutral* lebih mudah diidentifikasi daripada kelas *Weak* dan *Comparison*.

Tabel 3 F-Measure Optimasi SGD

Kelas	F-measure
Weak	0
Comparison	0
Point	79
Neutral	79

Hasil dari percobaan pemodelan menunjukkan bahwa nilai *F-Measure imbalance* yang terdapat pada kelas *Weak* dan *Comparison*. Hal tersebut disebabkan sebaran data kelas tidak merata dan model komputasi sulit untuk mengenali kelas tersebut hingga pada akhirnya data mengalami gap.

Selain tahapan yang sudah dilakukan dalam melakukan model komputasi yang memiliki label (*supervised learning*) Confusion Matrix menjadi acuan untuk merepresentasikan prediksi dan kondisi sebenarnya (aktual) dari data yang dihasilkan oleh *machine learning*, Confusion Matrix dapat memberikan informasi perbandingan hasil klasifikasi seperti menghitung nilai *accuracy*, *precision*, *recall*, dan *F-Measure*.

1. *Accuracy* merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Dengan kata lain, *accuracy* merupakan tingkat kedekatan nilai prediksi dengan nilai aktual (sebenarnya). Dapat dilihat pada persamaan 8.
   Akurasi = (TP + TN) / (TP + TN + FP + FN)) * 100% (8)
2. *Precision* adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Dalam *categorical classification*, *precision* dapat dibuat sama dengan nilai prediksi positif. Dapat dilihat pada persamaan 9.
   Precision = (TP / (TP + FP)) * 100% (9)
3. *Recall* adalah data penghapusan yang berhasil diambil dari data yang relevan dengan *query* atau tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Dapat dilihat pada persamaan 10.
   Recall = (TP / (TP + FN)) * 100% (10)
4. F-Measure merupakan perbandingan rata-rata presisi dan recall yang dibobotkan. Nilai *recall* dan *precission* pada suatu keadaan dapat memiliki bobot yang berbeda. Ukuran yang menampilkan timbal balik antara *recall* dan *precission* adalah *F-Measure* yang merupakan bobot *harmonic mean* dan *reall* dan *precission*. Dapat dilihat pada persamaan 11.
   F-Measure = (2 ∗ Recall ∗ Precision) / (Recall + Precision) (11).

Tabel 4 Confusion Matrix Klasifikasi Kalimat

True Label	Weak	0	0	4	1
	Comparison	0	0	13	4
	Neutral	0	0	161	11
	Point	0	0	60	150
		Weak	Comparision	Neutral	Point
	Predicted Label				

Tabel 5 Tabel Hasil Akurasi, Precision, Recall, dan F-Measure

Akurasi	Precision	Recall	F-Measure
0,77	**0,77**	**0,77**	**39,5**

Dari hasil Confusion matrix pada tabel 4 dapat dilihat bahwa sebaran kelas terjadi *imbalance* dan sistem sulit untuk membaca kelas *Weak* dan *Comparison*. Hal tersebut disebabkan adanya gap yang cukup jauh dengan kelas *Point* dan *Neutral*.

4. DISKUSI

Dalam penelitian ini masih adanya fluktuasi dan belum asimtotik pada proses pembelajaran mesin sehingga terdapat peluang untuk pengembangan pada penelitian yang akan datang. Beberapa saran untuk pengembangan penelitian ini diantaranya dapat melakukan klasifikasi kalimat dengan tidak hanya satu bahasa saja, malainkan klasifikasi kalimat dalam berbagai bahasa (*multi-language sentence classification*), selanjutnya dalam proses pembelajaran mesin tidak hanya ditinjau dari model optimasi saja namun dengan banyaknya kelas yang digunakan agar menciptakan variasi pola klasifikasi yang lebih variatif, Selain itu diharapkan pada penelitian selanjutnya dapat mengkombinasikan Recurrent Convolutional Neural Network (RCNN) atau menggunakan arsitektur Attention Mechanism sebagai pebandingan dengan LSTM pada model komputasi klasikasi kalimat. Dengan saran-saran tersebut pada penelitian selanjutnya diharapkan dapat membantu dalam melakukan klasifikasi kalimat yang lebih optimal.

5. KESIMPULAN

Penelitian ini telah mengembangkan klasifikasi kalimat menggunakan RNN dengan menunjukkan bahwa hasil tingkat pembelajaran terbaik diperoleh dengan pengoptimalan SGD dengan nilai akurasi 77,48% dan Loss 0,71%. SGD bekerja dengan memilih data sampel acak dari satu atau beberapa bagian dari data pelatihan dalam satu iterasi dengan cara yang iteratif. Data sampel acak ini dikoreksi berdasarkan aturan yang melibatkan gradien pertama untuk mengukur perubahan fungsi bersama dengan perubahan nilai *input*. Sehingga hasil akhir pada penelitian ini menunjukkan bahwa skor *F-Measure* mencapai 39,5%.

DAFTAR PUSTAKA

[1] D. Jurgens, S. Kumar, R. Hoover, D. McFarland, and D. Jurafsky, "Citation Classification for Behavioral Analysis of a Scientific Field," 2016.

[2] S. Teufel, A. Siddharthan, and D. Tidhar, "Automatic classification of citation function," *COLING/ACL 2006 - EMNLP 2006: 2006 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 103–110, 2006.

[3] D. Han, H. Koide, and A. Inoue, "Acquisition of Scientific Literatures based on Citation-reason Visualization," *International Joint Conference on Computer Vision, Imaging, and Computer Graphics Theory and Applications*, vol. 2, pp. 123–130, 2016.

[4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations ofwords and phrases and their compositionality," *Advances in Neural Information Processing Systems*, pp. 1–9, 2013.

[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] R. ALRashdi and S. O'Keefe, "Deep Learning and Word Embeddings for Tweet Classification for Crisis Response," *International Conference on Machine Learning, Big Data and Business Intelligence*, vol. Conference, no. October, 2019.

[7] F. Zhang, W. Gao, and Y. Fang, "News title classification based on sentence-Linear Discriminant Analysis Model and Word Embedding," *Proceedings - 2019 International Conference on Machine Learning, Big Data and Business Intelligence, MLBDBI 2019*, pp. 237–240, 2019.

[8] 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, pp. 12467–12475, 2019.

[9] S. Teufel, A. Siddharthan, and D. Tidhar, "An annotation scheme for citation function," *COLING/ACL 2006 - SIGdial06: 7th SIGdial Workshop on Discourse and Dialogue, Proceedings of the Workshop*, pp. 80–87, 2006.

[10] G. H. Rachman, M. L. Khodra, and D. H. Widyantoro, "Word embedding for rhetorical sentence categorization on scientific articles," *Journal of ICT Research and Applications*, vol. 12, no. 2, pp. 168–184, 2018.

[11] P. Liu, X. Qiu, and H. Xuanjing, "Recurrent

neural network for text classification with multi-task learning," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2016-Janua, pp. 2873–2879, 2016.

[12] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM Neural Network for Text Classification," 2015.

[13] D. Yogatama, C. Dyer, W. Ling, and P. Blunsom, "Generative and Discriminative Text Classification with Recurrent Neural Networks," 2017.

[14] W. Xia, W. Zhu, B. Liao, M. Chen, L. Cai, and L. Huang, "Novel architecture for long short-term memory used in question classification," *Neurocomputing*, vol. 299, pp. 20–31, 2018.