Vol.7 No.2 Tahun 2021, pp.103-112
Jurnal Teknologi dan Manajemen Informatika
http:// http://jurnal.unmer.ac.id/index.php/jtmi
P-ISSN: 1693-6604 E-ISSN: 2580-8044

Perbandingan Model *Deep Learning* untuk Klasifikasi *Sentiment Analysis* dengan Teknik *Natural Languange Processing*

Firman Pradana Rachman[1], Handri Santoso[2]

[1,2] Program Studi Magister Teknologi Informasi, Universitas Pradita, Indonesia

Article Info

Article History
Received: 05-10-2021
Revised : 02-11-2021
Accepted : 06-11-2021

Keywords
Model Comparison
Deep Learning
Sentiment Analysis
Natural Language Processing.

ABSTRACT

Everyone has an opinion or opinion on a product, public figure, or government policy that is spread on social media. Opinion data processing is called sentiment analysis. In processing large opinion data, it is not enough to only use machine learning, but you can also use deep learning combined with NLP (Natural Language Processing) techniques. This study compares several deep learning models such as CNN (Convolutional Neural Network), RNN (Recurrent Neural Networks), LSTM (Long Short-Term Memory), and several variants to process sentiment analysis data from Amazon and Yelp product reviews.

✉ Corresponding Author
Firman Pradana Rachman,
Universitas Pradita,
Tel. +62 87785883652
firman.pradana.student@pradita.ac.id

PENDAHULUAN

Sentiment analysis atau penggalian opini adalah suatu teknik pemrosesan bahasa alami yang terdiri dari beberapa kategori seperti positif, netral maupun negatif. *Sentiment analysis* bagi pelaku bisnis banyak dimanfaatkan untuk mengetahui keinginan dan kebutuhan pelanggan. *Sentiment analysis* juga dapat dijadikan tolak ukur untuk melihat popularitas maupun dukungan masyarakat terhadap seorang *public figure* atau tokoh masyarakat.

Data opini dari setiap orang setiap harinya berjumlah sangat banyak. Data tersebut tersebar di Twitter, Facebook, Instagram maupun di media sosial lainnya. Pengelolaan data yang besar tersebut membutuhkan teknologi yang mampu mengolahnya. *Deep Learning* adalah salah satu cara untuk menganalisa data yang besar tersebut. Pada penelitian ini akan mencoba mengolah data *sentiment analysis* dari Amazon dan Yelp dengan menggunakan model *deep learning* seperti CNN (*Convolutional Neural Network*), RNN (*Recurrent Neural Network*), LSTM (*Long Short-Term Memory*) dan *bidirectional* LSTM yang dikombinasikan prosesnya menggunakan teknik dari NLP *(Natural Languange Processing).* Tools yang digunakan dalam penelitian ini menggunakan *Phyton* dengan *library* dari *scikit-learn* dan *keras*.

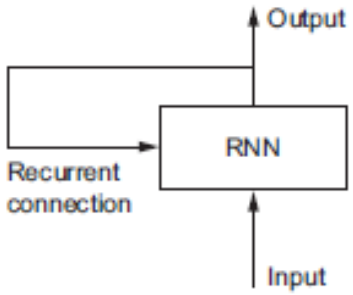
Tujuan dari penelitian ini untuk mengetahui perbedaan performa dari model *deep learning* untuk klasifikasi *sentiment analysis*. Karena berbasis teks, dalam penelitan ini juga menggunakan beberapa teknik dari NLP. Harapannya penelitian ini bisa menjadi referensi bagi yang ingin mengimplementasikan model *deep learning* untuk klasifikasi data *sentiment analysis* yang berbentuk teks.

Deep learning adalah metode khusus pembelajaran mesin yang menggabungkan jaringan saraf di lapisan berturut-turut untuk belajar dari data secara *iterative* dengan meniru cara kerja otak manusia sehingga komputer dapat dilatih untuk menangani abstraksi dan masalah yang didefinisikan dengan buruk[1]. *Deep Learning* pertama kali di perkenalkan olah Geoffrey Hinton pada tahun 2006. Keberadaan *deep learning* untuk mengatasi kekurangan dari metode *machine learning* konvensional biasa. Salah satunya adalah kemampuan *feature engineering* yang dapat merekayasa fitur secara otomatis. Teknologi *deep learning* mampu memberikan hasil yang lebih baik dan sebanding dengan jumlah data yang di tambahkan. *Deep learning* mempunyai beberapa kelebihan[2], yaitu:

a. *Universal, deep learning* dapat diimplementasikan pada hampir setiap ranah aplikasi.
b. *Robust, deep learning* tahan terhadap berbagai variasi data yang secara alami sesuai dengan fakta yang sebenarnya. *Deep learning* tidak memerlukan fitur yang didesain secara *artifisial* karena bisa mempelajarinya secara otomatis.
c. *Generalization, deep learning* bisa digunakan dalam berbagai aplikasi dan tipe data karena mempunyai kemampuan *transfer learning*.
d. *Scability, deep learning* mempunyai skalabilitas yang tinggi, contohnya sebuah jaringan ResNet yang di buat oleh Microsoft terdiri dari 1202 lapisan data yang di implementasikan pada sebuah komputer[3].

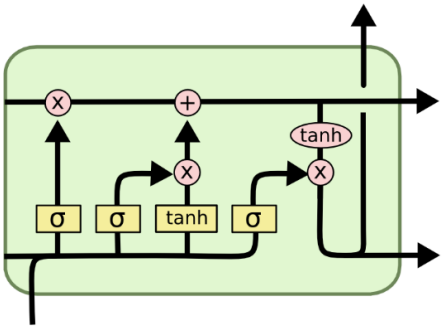
CNN *(Convolutional Neural Network)* adalah jenis jaringan saraf khusus untuk memproses data yang memiliki topologi seperti *grid*. Contohnya data deret waktu yang dapat dianggap sebagai kisi 1D ataupun data gambar, yang dapat dianggap sebagai kisi piksel 2D[4]. Komponen CNN terdiri dari satu jenis lapisan masukan (*input layer*), lapisan keluaran (*output layer*) dan beberapa lapisan yang tersembunyi. Lapisan yang tersembunyi itu terdiri dari *convolutional layers, pooling layers, normalization layer, relu layer, fully connected layers dan loss layer*[5]. Salah satu penerapan CNN dengan 1D *Convolution* yaitu bisa di gunakan untuk klasifikasi teks[6].

RNN *(Reccurant Neural Network)* mengadopsi prinsip yang sama dengan CNN namun dalam kondisi yang lebih sederhana yaitu memproses urutan dengan mengulangi melalui elemen urutan dan mempertahankan keadaan yang berisi informasi relatif dari apa yang di lihatnya sejauh ini. Sehingga RNN adalah jenis jaringan saraf tiruan yang memiliki *loop internal*[7].



Gambar 1. RNN dengan Pengulangan

LSTM *(Long Short Term Memory)* pertama kali diperkenalkan oleh Sepp Hochreiter dan Jurgen Schimdhuber pada tahun 1997. LSTM menggunakan mekanisme 4 gerbang untuk mengatasi masalah pada arsitektur RNN biasa, salah satunya bisa memproses *sequences* yang panjang. Gambar di bawah ini adalah arsitektur dari LSTM[8].



Gambar 2. Arsitektur LSTM (Hochreiter and Urgen Schidhuber, 1997)

Salah satu varian dari LSTM adalah *bidirectional* LSTM. Metode ini menghubungkan dua lapisan tersembunyi dari arah yang berlawanan ke *output* yang sama. Dengan bentuk pembelajaran mendalam generatif ini, lapisan keluaran dapat memperoleh informasi dari keadaan masa lalu (mundur) dan masa depan (maju) secara bersamaan[9]. Jika di LSTM hanya melihat searah yaitu masa lalu maka di *bidirectional* LSTM bisa melihat masa lalu dan masa depan. Penelitian sebelumnya menunjukkan bahwa metode *bidirectional* LSTM bisa digunakan untuk *Keyphrase Extraction*[10].

NLP *(Natural Languange Processing)* adalah bidang ilmu komputer yang berkembang dari studi bahasa, linguistik dan kecerdasan buatan. Tujuan NLP adalah untuk merancang dan membangun aplikasi yang memfasilitasi interaksi manusia dengan mesin dan perangkat lain melalui penggunaan bahasa alami[11]. Beberapa bidang utama NLP diantaranya adalah *question answering systems (qas), summarization, machine translation, speech recognition, document classification.*

Beberapa istilah dan teknik yang digunakan dalam NLP diantaranya adalah:

a. *Token*, yaitu unit kecil seperti kata, tanda petik, angka dari sebuah kalimat.
b. *Tokenization*, adalah proses pemecahan dari sebuah kalimat menjadi *token-token.*
c. *Sentence*, yaitu sebuah kalimat itu sendiri.
d. *Corpus*, merupakan kumpulan *text* dalam jumlah besar dan berbentuk *raw data*.
e. *Count Vectorizer,* adalah suatu cara untuk membuat *vector* kalimat. Tujuannya untuk mengambil kata-kata dari setiap kalimat dan menciptakan *vocabulary* dari semua kata unik dalam kalimat. *Vocabulary* ini kemudian dapat digunakan untuk membuat *feature vector* dari jumlah kata.
f. *Word Embeddings*, adalah suatu cara untuk merepresentasikan sebuah kata sebagai *vector*. *Word embeddings* tidak memahami teks seperti yang dilakukan manusia, tetapi mereka lebih memetakan struktur statistik dari bahasa yang digunakan dalam *corpus*. Tujuan mereka adalah memetakan makna semantik ke dalam ruang geometris. Ruang geometris ini kemudian disebut ruang *embedding*.

Penelitian yang terkait NLP dan *Sentiment Analysis* yang pernah dilakukan sebelumnya adalah Klasifikasi Sentimen pada Twitter dengan *Naive Bayes Classifier*[12]. Pada penelitian ini melakukan *sentiment analysis* di Twitter dengan topik Presiden Joko Widodo dan pemerintahannya. Hasil akhir menunjukan hasil akurasi 66.79% dengan menggunakan *Naive Bayes Classifier.* Komposisi data terdiri dari sentimen positif 28%, sentimen negatif 20% dan sentimen netral 52%.

Perbedaan performa *Simple* RNN dan LSTM dalam pemrosesan *text* pernah ditunjukkan dalam penelitian yang berjudul "LSTM And Simple RNN Comparison In The Problem Of Sequence To Sequence On Conversation Data Using Bahasa Indonesia"[13]. Pada kasus penelitian ini, pemprosesan teks dibuat untuk membuat model *chatbot*. Hasil akhirnya menunjukan bahwa model LSTM lebih cepat performanya dan memberikan hasil yang lebih baik.

Dalam penelitian lain metode *machine learning* bisa dikombinasikan dengan *metode*

deep learning. Pada penelitian yang berjudul "Sentiment Analysis Based Method for Amazon Product Reviews" mengusulkan bahwa penggunaan *Naive Bayes* memberikan solusi untuk klasifikasi, KNN membantu dalam pengelompokan. Kemudian data akan dilatih menggunakan model *deep learning* berbasis LSTM untuk memberikan akurasi solusi yang lebih baik[14].

Penggunaan *word embedding* sering dipakai dalam pemrosesan teks dengan teknik NLP. Perbandingan kinerja antara Word2vec, Glove, dan Fasttext pada klasifikasi teks tidak berbeda jauh dan sangat kompetitif[15]. Penggunaanya sangat tergantung dengan permasalahan dan data yang dihadapi.

Dari beberapa penelitian tersebut menunjukan bahwa metode *machine learning* atau *deep learning* bisa dipakai untuk klasifikasi *sentiment analysis*. Pada kesempatan ini, penelitian hanya berfokus pada penggunaan model *deep learning* untuk klasifikasi *sentiment analysis* dengan teknik NLP dan *pretrained word embedding* dengan menggunakan Glove. Metode yang digunakan adalah beberapa teknik dan varian model dari Neural Network, CNN, RNN dan LSTM.

METODE

Metode penelitian yang digunakan adalah metode eksperimen, yaitu melakukan beberapa pengujian model *deep learning* dan membandingkan hasilnya satu sama lain yang didasari dari *literatur review* sebelumnya. Terdapat beberapa tahap dalam penelitian ini, yaitu:

a. Pengumpulan data, yang diambil adalah data dari Kaggle yang berisi *sentiment analysis* dari *review* produk Amazon dan Yelp. Isi data terdiri dari 2 *column* yaitu kolom pertama adalah komentar dan kolom kedua adalah klasifikasi apakah data tesebut negatif atau positif. Jika positif maka bernilai 1 dan jika negatif bernilai 0. Jumlah data berisi 292,863 baris.
b. Preprocessing, pada tahap ini beberapa teknik dari NLP digunakan, yaitu *count vectorizer*, *tokenization* dan *pretrained word embeddings*
c. Pengujian model, beberapa metode *deep learning* yang diuji dalam penelian ini adalah:
    - *Neural Network* dengan penambahan *Layer* GlobalMaxPooling1D
    - CNN dengan 1D *Convolution*
    - *Simple* RNN
    - LSTM
    - *Bidirectional* LSTM
d. Hasil dan evaluasi, tahapan ini melakukan *review* hasil dari perbandingan model-model tersebut dan menarik sebuah kesimpulan.

HASIL DAN PEMBAHASAN

Implementasi dibagi menjadi 2 bagian besar yaitu *preprocessing* dan pengujian model.

Preprocessing

Tahapan sebelum melakukan uji coba model adalah pengumpulan data dan *preprocessing*. Data di-*download* dari Kaggle, kemudian dibagi menjadi dua bagian yaitu untuk *training* dan *test*. Tahap *preprocessing* mengolah data menggunakan teknik NLP, yaitu:

1. *Count Vectorizer*, menggunakan *scikit-learn library* untuk membuat vektor kalimat. *Library* ini akan mengambil kata-kata dari setiap kalimat dan menciptakan *vocabulary* dari semua kata unik dalam kalimat.

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
vectorizer.fit(sentences_train)

X_train = vectorizer.transform(sentences_train)
X_test  = vectorizer.transform(sentences_test)
```

Gambar 3. Implementasi *Count Vectorizer*

2. *Tokenization*, tujuannya untuk membuat *token-token* dengan menggunakan *library preproscessing* dari *keras.*

```
from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(sentences_train)

X_train = tokenizer.texts_to_sequences(sentences_train)
X_test = tokenizer.texts_to_sequences(sentences_test)

vocab_size = len(tokenizer.word_index) + 1  # Adding 1 because

from keras.preprocessing.sequence import pad_sequences

maxlen = 100

X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)
```

Gambar 4. Implementasi *Tokenizer*

3. *Pretrained Word Embedding*, menggunakan *word embedding* dari GloVe (*Global Vectors for Word Representation*) yang dikembangkan oleh Stanford NLP Group. *File* ada di http://nlp.stanford.edu/data/glove.6B.zip. Tujuannya untuk mengubah sebuah kata menjadi sebuah vektor atau *array* yang terdiri dari kumpulan angka.

Pengujian Model

Tahapan selanjutnya adalah pengujian model. Berikut ini adalah hasil dan pembahasan dari masing-masing model yang diuji tersebut.

1. *Neural network* tanpa *pretrained word embedding*
   Percobaan pertama menggunakan *neural network* tanpa menggunakan *pretrained word embedding.* Tujuannya untuk menunjukan apakah ada perbedaan antara menggunakan *pretrained word embedding* atau tanpa menggunakannya sama sekali.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 50)           2111150
_________________________________________________________________
flatten (Flatten)            (None, 5000)              0
_________________________________________________________________
dense (Dense)                (None, 10)                50010
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 11
=================================================================
Total params: 2,161,171
Trainable params: 2,161,171
Non-trainable params: 0
```

Gambar 5. Implementasi NN tanpa *Pretrained Word Embedding*

Hasilnya *training*-nya mencapai 86%, berikut gambaran detailnya.

- *Training loss*: 0.2733
- *Training accuracy*: 0.8660
- *Validation_loss*: 0.3052

- *Validation_accuracy*: 0.8617

2. *Neural network* dengan *pretrained word embedding*
   Percobaan selanjutnya menggunakan *neural network* dengan menggunakan *Pretrained Word Embedding* dan tambahan *Layer* GlobalMaxPooling1D. Tujuannya untuk mengurangi ukuran atau menurunkan *sample feature vector* yang masuk. Karena dalam *pooling max*, kita mengambil nilai maksimum semua fitur di *pooling* untuk setiap dimensi *feature*. Berikut implementasinya.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 100, 300)          12666900
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 300)               0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                3010
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 11
=================================================================
Total params: 12,669,921
Trainable params: 3,021
Non-trainable params: 12,666,900
```

Gambar 6. Implementasi NN dengan *Layer* GlobalMaxPooling1D

   Hasil akurasinya mencapai 87%. Dikarenakan menggunakan *Pretrained Word Embedding* ada sedikit peningkatan, maka percobaan selanjutya akan tetap menggunakan *Pretrained Word Embedding*. Berikut ini detail dari percobaan tersebut.

   - *Training loss*: 0.3074
   - *Training accuracy*: 0.8730
   - *Validation_loss*: 0.3155
   - *Validation_accuracy*: 0.8703

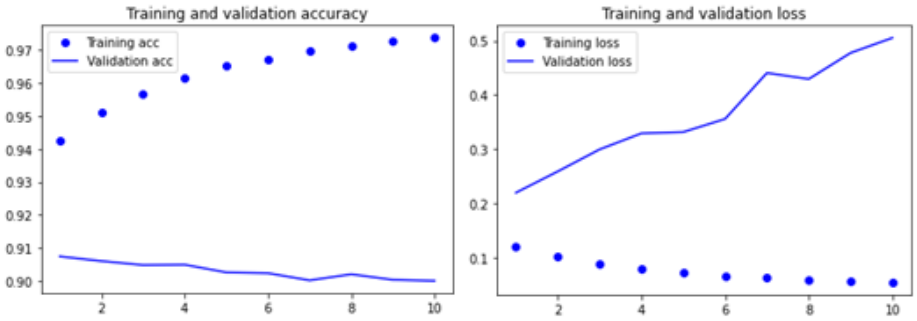
3. CNN dengan 1D *Convolution*. Percobaan ini menggunakan CNN dengan menambahkan lapisan Conv1D yang memang sering dipakai dalam pemrosesan *text*. Lapisan Conv1D ini ditambahkan sebelum lapisan GlobalMaxPooling1D. Berikut implementasinya menggunakan *phyton* dan *keras*.

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 100, 300)          12666900
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 96, 128)           192128
_________________________________________________________________
global_max_pooling1d_3 (Glob (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 11
=================================================================
Total params: 12,860,329
Trainable params: 12,860,329
Non-trainable params: 0
```

Gambar 7. Implementasi CNN dengan 1D *Convolution*

   Hasilnya terdapat peningkatan untuk nilai akurasi *training*-nya mencapai 97%, berikut rincian datanya.

   - *Training loss*: 0.0537
   - *Training accuracy*: 0.9737
   - *Validation_loss*: 0.5053
   - *Validation_accuracy*: 0.9001



Gambar 8. Grafik hasil dari model CNN dengan 1D *Convolution*

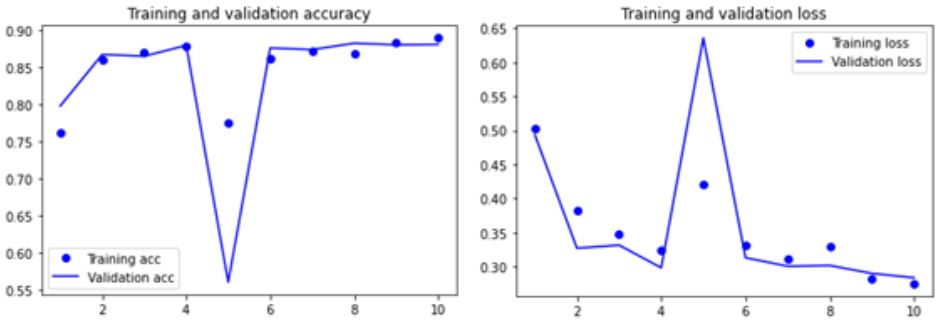
4. *Simple* RNN, merupakan salah satu varian dari model RNN yang tersedia di *Keras*. Beberapa penelitian menunjukan bahwa RNN bisa membaca kata dari sebuah kalimat satu persatu, kemudian memprediksi kata pada *output* satu persatu. Hasil akhirnya adalah sebuah kalimat[16]. Berikut implementasi dari metode *simple* RNN.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 100, 300)          12666900
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 100, 200)          100200
_________________________________________________________________
dense_2 (Dense)              (None, 100, 10)           2010
_________________________________________________________________
dense_3 (Dense)              (None, 100, 1)            11
=================================================================
Total params: 12,769,121
Trainable params: 12,769,121
Non-trainable params: 0
_________________________________________________________________
```

Gambar 9. Implementasi *Simple* RNN

Hasil akhirnya ternyata tidak lebih baik dibandingkan dengan percobaan menggunakan CNN dengan 1D *Convolution*. Berikut di bawah ini detail hasil dari *Simple* RNN.

- *Training loss*: 0.2784
- *Training accuracy*: 0.8880
- *Validation_loss*: 0.2837
- *Validation_accuracy*: 0.8807



Gambar 10. Grafik Hasil Dari *Simple* RNN

5. LSTM, merupakan metode *learning* yang sering digunakan untuk pemrosesan teks. Kelebihannya dibanding RNN adalah dia mampu membaca *sequence* yang panjang. RNN juga memiliki masalah yang disebut dengan *vanishing gradient* atau hilangnya

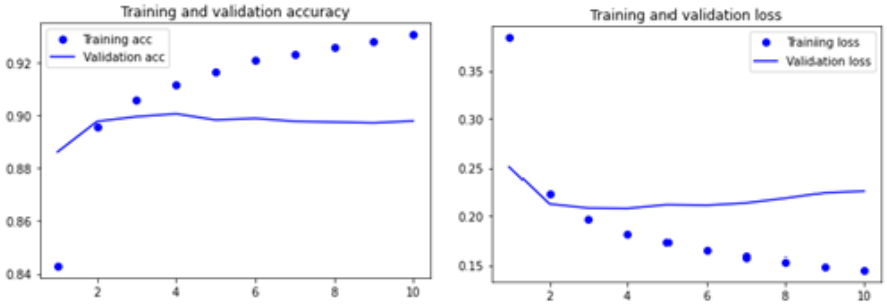
efektifitas *gradient*. Dengan LSTM masalah *vanishing gradient* tersebut bisa tertangani[8]. Berikut implementasinya di *phyton* dengan menggunakan library *keras*.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 300)          12666900
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 20)           25680
_________________________________________________________________
dense_1 (Dense)              (None, 100, 1)            21
=================================================================
Total params: 12,692,601
Trainable params: 12,692,601
Non-trainable params: 0
```

Gambar 11. Implementasi LSTM

Hasil akhirnya menunjukan bahwa LSTM memang lebih baik dibandingkan dengan *simple* RNN dengan angka 93% untuk *training accuracy*. Hal ini selaras dengan penelitan sebelumnya bahwa LSTM juga lebih baik dibandingkan dengan *simple* RNN dalam pemrosesan teks di *chatbot*[13]. Berikut hasil detailnya.

- *Training loss*: 0.1404
- *Training accuracy*: 0.9331
- *Validation_loss*: 0.2259
- *Validation_accuracy*: 0.8980



Gambar 12. Grafik hasil pengujian LSTM

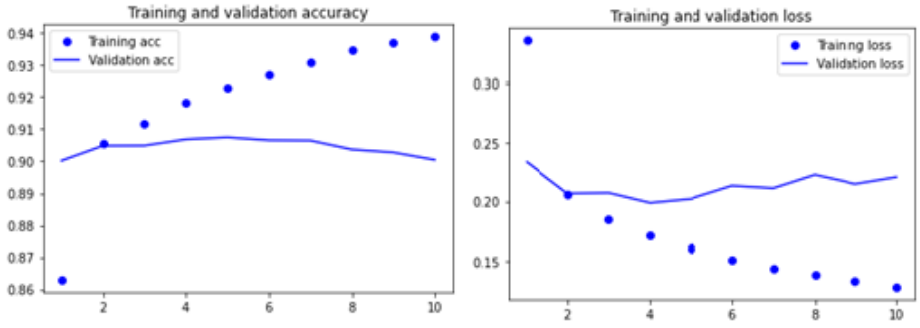
6. *Bidirectional* LSTM merupakan varian dari LSTM dan mempunyai kelebihan dibandingkan dengan LSTM biasa. *Bidirectional* LSTM mampu membaca dua arah yaitu data dari masa lampau dan juga data di masa yang akan datang. Sedangkan LSTM biasa hanya bisa membaca data hanya dari masa lampau. Berikut adalah model dari *bidirectional* LSTM.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 100, 300)          12666900
_________________________________________________________________
bidirectional_3 (Bidirection (None, 100, 40)           51360
_________________________________________________________________
dense_3 (Dense)              (None, 100, 1)            41
=================================================================
Total params: 12,718,301
Trainable params: 12,718,301
Non-trainable params: 0
```

Gambar 13. Implementasi *Bidirectional* LSTM

Hasilnya terdapat peningkatan jika dibandingkan dengan LSTM biasa dengan nilai akurasinya mencapai 94%. Berikut data detailnya.

- *Training loss*: 0.1249
- *Training accuracy*: 0.9411
- *Validation_loss*: 0.2205
- *Validation_accuracy*: 0.9004



Gambar 14. Grafik hasil dari *Bidirectional* LSTM

SIMPULAN DAN SARAN

Percobaan dalam pengujian model *deep learning* tersebut, hasilnya terangkum dalam tabel berikut.

Tabel 1. Hasil Perbandingan Model *Deep Learning*

Metode *Deep Learning*	*Training Loss*	*Training Accuracy*	*Validation Loss*	*Validation Accuracy*
NN (tanpa *word embedding*)	0.2733	0.8660	0.3052	0.8617
NN (dengan w*ord embedding*)	0.3074	0.8730	0.3155	0.8703
CNN (1D *Convolution*)	0.0537	0.9737	0.5053	0.9001
RNN (*Simple* RNN)	0.2784	0.8880	0.2837	0.8807
LSTM	0.1404	0.9331	0.2259	0.8980
Bidirectional LSTM	0.1249	0.9411	0.2205	0.9004

Hasil pengujian menempatkan CNN (1D *Convolution*) menempati posisi paling baik dengan nilai akurasi *training* mencapai 97%. Posisi kedua adalah *bidirectional* LSTM dengan nilai akurasi *training* mencapai 94% mengungguli LSTM dan *Simple* RNN. Sedangkan posisi terendah adalah *Neural Network* (tanpa *word embedding*) dengan nilai akurasi *training* hanya 86%. Sedikit lebih rendah jika dibandingkan dengan NN dengan *word embedding.*

Model CNN (1D *Convolution*) direkomendasikan untuk klasifikasi *sentiment analysis* berbasis teks dengan nilai akurasi yang paling tinggi dibandingkan dengan model lainnya. Percobaan ini menggunakan teknik NLP yaitu *Count Vectorizer* dan *Tokenization* untuk hasil yang lebih baik. Penggunaan *Pretrained Word Embedding* dari Glove juga direkomendasikan karena memberikan peningkatan hasil yang lebih baik dibandingkan dengan tidak menggunakannya sama sekali.

DAFTAR PUSTAKA

[1] J. Hurwitz and D. Kirsch, *Machine Learning For Dummies IBM Limited Edition*, 2018th ed. New Jersey: John Wiley & Sons, Inc, 2018.

[2] Suyanto, K. Ramadhani Nur, and S. Mandala, *Deep Learning Modernisasi Machine Learning untuk Big Data*. Bandung: INFORMATIKA, 2019.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp.

770–778, 2016, doi: 10.1109/CVPR.2016.90.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge: The MIT Press, 2016.

[5] M. Z. Alom *et al.*, "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," 2018, [Online]. Available: http://arxiv.org/abs/1803.01164.

[6] A. Jacovi, O. Sar Shalom, and Y. Goldberg, "Understanding Convolutional Neural Networks for Text Classification," no. January, pp. 56–65, 2019, doi: 10.18653/v1/w18-5408.

[7] F. CHOLLET, *Deep Learning with Phyton*. New York: Manning Publications Co., 2018.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[9] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997, doi: 10.1109/78.650093.

[10] M. Basaldella, E. Antolli, G. Serra, and C. Tasso, "Bidirectional LSTM recurrent neural network for keyphrase extraction," *Commun. Comput. Inf. Sci.*, vol. 806, no. January, pp. 180–187, 2018, doi: 10.1007/978-3-319-73165-0_18.

[11] J. Pustejovsky and A. Stubbs, *Natural Language Annotation for Machine Learning -- A guide to Corpus-building for applications*. 2013.

[12] S. Suryono, E. Utami, and E. T. Luthfi, "Klasifikasi Sentimen Pada Twitter Dengan Naive Bayes Classifier," *Angkasa J. Ilm. Bid. Teknol.*, vol. 10, no. 1, p. 89, 2018, doi: 10.28989/angkasa.v10i1.218.

[13] Y. D. Prabowo, H. L. H. S. Warnars, W. Budiharto, A. I. Kistijantoro, Y. Heryadi, and Lukas, "Lstm and Simple Rnn Comparison in the Problem of Sequence to Sequence on Conversation Data Using Bahasa Indonesia," *1st 2018 Indones. Assoc. Pattern Recognit. Int. Conf. Ina. 2018 - Proc.*, no. November 2020, pp. 51–56, 2019, doi: 10.1109/INAPR.2018.8627029.

[14] M. J. Budhwar, "Sentiment Analysis based Method for Amazon Product Reviews," pp. 54–57, 2021.

[15] A. Nurdin, B. Anggo Seno Aji, A. Bustamin, and Z. Abidin, "Perbandingan Kinerja Word Embedding Word2Vec, Glove, Dan Fasttext Pada Klasifikasi Teks," *J. Tekno Kompak*, vol. 14, no. 2, p. 74, 2020, doi: 10.33365/jtk.v14i2.732.

[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, pp. 3104–3112, 2014.