

Penggunaan RNN dan LSTM dalam Pengolahan Data Sekuensial

Paulus Setiawan Suryadjaja

Magister Informatika, Sekolah Teknik Elektro dan Informatika, Institut
Teknologi Bandung, Bandung, Jawa Barat, Indonesia
S_paulus@ymail.com

Abstrak. Data sekuensial membutuhkan teknik khusus dalam pengolahannya. Hal ini dikarenakan pada pengolahan data sekuensial, data dari tahapan input sebelumnya perlu dijadikan atribut tersendiri. Beberapa contoh metode yang dapat digunakan untuk mengolah data sekuensial adalah *Recursive Neural Network* (RNN) dan *Long Short Term Memory* (LSTM). RNN adalah salah satu bentuk arsitektur *Artificial Neural Networks* (ANN) yang dirancang khusus untuk memproses data sekuensial. RNN mampu menyimpan memori yang memungkinkan untuk mengenali pola data dengan baik, kemudian menggunakannya untuk membuat prediksi yang akurat. Sedangkan LSTM adalah modifikasi dari RNN untuk mengatasi masalah *vanishing gradient* yang terjadi pada RNN biasa dengan mengatur memori pada setiap masukannya dengan menggunakan *memory cells* dan *gate units*. Pada percobaan ini disajikan contoh penggunaan RNN untuk mengimplementasikan *character level language model* dan penggunaan LSTM untuk prediksi harga saham.

1 Latar Belakang

Data sekuensial (*sequence data*) mempunyai karakteristik di mana sampel diproses dengan suatu urutan (misalnya waktu), dan suatu sampel dalam urutan mempunyai hubungan erat satu dengan yang lain [4]. Contoh-contoh data sekuensial dan aplikasinya misalnya:

- rangkaian kata-kata dalam penerjemahan bahasa
- sinyal audio dalam pengenalan suara
- nada-nada dalam sintesa musik
- rangkaian kata-kata dalam klasifikasi sentimen
- deret DNA dalam pemrosesan rangkaian DNA [4]

Pemrosesan data sekuensial seperti di atas tidak cocok dilakukan dengan model umpan maju (*feed forward*) yang sudah dikenal sebelumnya seperti model linear, *multi-layer perceptron*, dan CNN [4]. Kendala utamanya adalah:

- Model-model ini tidak mempunyai “memori”; setiap sampel akan diproses secara sama rata tanpa ada pertimbangan atas sampel-sampel sebelumnya.
- Model-model tersebut mengasumsikan data yang IID (*independent and*

identically distributed). Asumsi ini tidak dapat dipenuhi oleh data sekuensial karena suatu sampel mempunyai ketergantungan yang erat dengan sampel-sampel lainnya.

- Model-model ini mengolah masukan dengan panjang *input* yang tetap, sedangkan *input* data sekuensial mempunyai panjang yang tidak menentu, dan terdapat kemungkinan memiliki *input* yang sangat panjang.
- Model-model ini tidak bisa melakukan generalisasi fitur yang dipelajari di satu posisi ke posisi lain (misalnya model mengenali Joni sebagai nama orang di posisi awal kalimat, seharusnya model juga mengenali Joni sebagai nama orang ketika dia muncul di posisi lain dalam kalimat) [4].

Oleh karena itu, dikembangkan model-model khusus yang dirancang untuk melakukan pengolahan data sekuensial, misalnya model *Recursive Neural Network* (RNN) dan *Long Short Term Memory* (LSTM). Pada bagian 2 laporan percobaan ini dijelaskan dasar teori mengenai RNN dan aplikasi RNN dalam membuat *character-level language model* untuk nama dinosaurus. Sedangkan pada bagian 3 laporan percobaan ini dijelaskan dasar teori mengenai LSTM dan aplikasi LSTM dalam memprediksi pergerakan harga saham.

2 *Recursive Neural Network* (RNN)

2.1 Dasar Teori

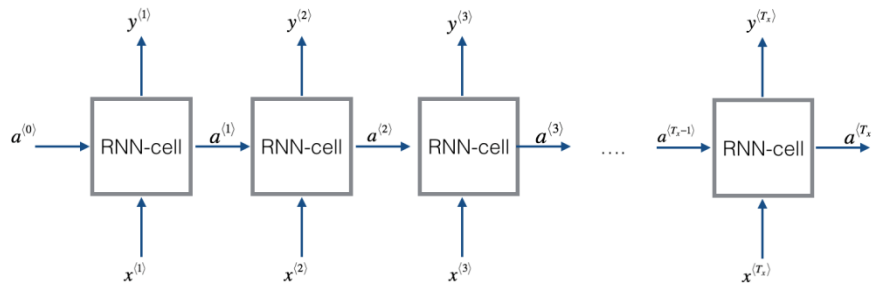
Recurrent Neural Networks (RNN) merupakan salah satu bentuk arsitektur *Artificial Neural Networks* (ANN) yang dirancang khusus untuk memproses data yang bersambung/ berurutan (*sequential data*). RNN biasanya digunakan untuk menyelesaikan tugas yang terkait dengan data *time series*, misalnya data ramalan cuaca. Sebagai contoh, cuaca hari ini dapat bergantung pada cuaca hari sebelumnya, jika hari sebelumnya mendung, maka kemungkinan hari ini akan hujan [1].

RNN tidak membuang begitu saja informasi dari masa lalu dalam proses pembelajarannya. Hal inilah yang membedakan RNN dari ANN biasa. RNN mampu menyimpan memori/ ingatan (*feedback loop*) yang memungkinkan untuk mengenali pola data dengan baik, kemudian menggunakannya untuk membuat prediksi yang akurat. Cara yang dilakukan RNN untuk dapat menyimpan informasi dari masa lalu adalah dengan melakukan iterasi di dalam arsitekturnya, yang secara otomatis membuat informasi dari masa lalu tetap tersimpan.[1] RNN dapat menerima urutan *input* satu per satu, dan mengingat beberapa informasi / konteks melalui aktivasi *hidden layer* yang diteruskan dari satu *time step* ke *time step* berikutnya. Ini memungkinkan RNN *uni-directional* untuk mengambil informasi dari masa lalu untuk memproses input selanjutnya. Sedangkan *bidirectional* RNN dapat mengambil konteks dari masa lalu dan masa depan [2].

Aplikasi-aplikasi RNN termasuk pemrosesan bahasa alami (*Natural Language Processing/ NLP*), pengenalan suara (*speech recognition*), terjemahan mesin (*machine translation*), pemodelan bahasa tingkat karakter (*character-level*

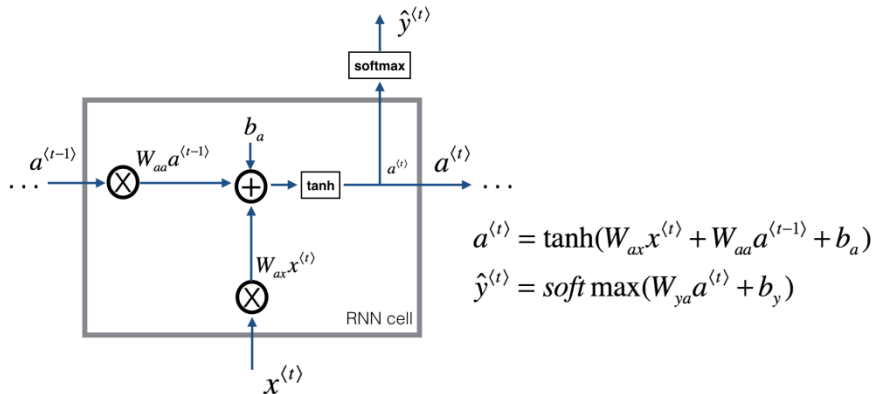
language modeling), klasifikasi gambar (*image classification*), keterangan gambar (*image captioning*), prediksi saham (*stock prediction*), dan rekayasa keuangan (*financial engineering*) [1].

Model dasar RNN yang akan diimplementasikan pada percobaan ini dapat dilihat pada Gambar 1. Tahap – tahap yang diperlukan untuk mengimplementasikan RNN adalah; terapkan perhitungan yang diperlukan untuk satu *time step* RNN, kemudian terapkan iterasi terhadap *time step* untuk memproses semua *input* satu per satu [2].



Gambar 1. Model dasar RNN. [2]

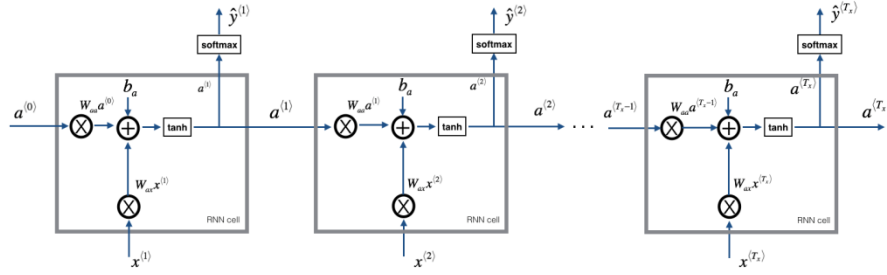
RNN dibentuk dari pengulangan dari operasi yang ada pada setiap sel. Pertama-tama akan diimplementasikan perhitungan untuk satu *time step*. Operasi yang dijalankan pada setiap *time step* RNN ditunjukkan pada Gambar 2 [2].



Gambar 2. Operasi yang dijalankan pada setiap *time step* RNN. [2]

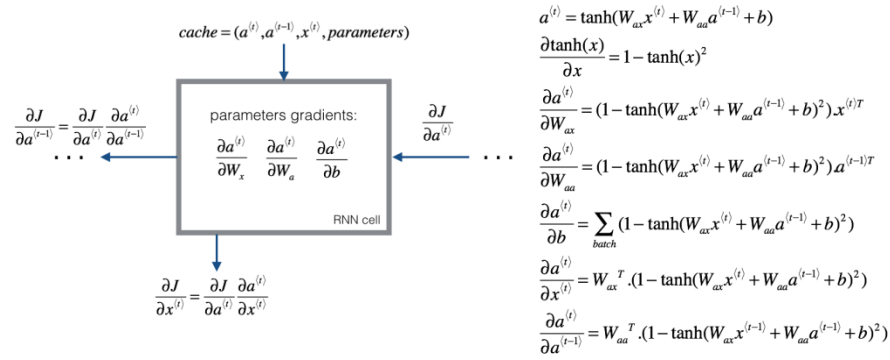
RNN dapat dilihat sebagai pengulangan sel yang dibuat pada tahap

sebelumnya. Jika urutan *input* data dilakukan dalam 10 *time step*, maka sel RNN akan diduplikasi 10 kali. Setiap sel mengambil *input hidden state* dari sel sebelumnya ($a^{(t-1)}$) dan data *input time step* saat ini ($x^{(t)}$). Proses ini menghasilkan *hidden state* ($a^{(t)}$) dan prediksi ($\hat{y}^{(t)}$) untuk *time step* saat ini [2]. Struktur dasar jaringan RNN dapat dilihat pada Gambar 3.



Gambar 3. Struktur dasar jaringan RNN. [2]

Selanjutnya, dilakukan proses *backpropagation*. Pada *fully connected neural network*, *backpropagation* untuk menghitung turunan terhadap *cost* untuk memperbarui parameter. Demikian pula, pada RNN dapat dihitung turunan terhadap *cost* untuk memperbarui parameter [2]. Perhitungan *backward pass* yang dilakukan pada sebuah sel RNN ditunjukkan pada Gambar 4.



Gambar 4. Perhitungan *backward pass* yang dilakukan pada sebuah sel RNN. [2]

Untuk melakukan *backward pass* untuk keseluruhan jaringan RNN, perlu dilakukan iterasi perhitungan *backward pass* untuk setiap *time step*, dimulai dari akhir jaringan. Pada setiap langkah, ditambahkan nilai dba , $dWaa$, $dWax$ dan menyimpan dx . Pada tahap ini, perhitungan *gradien cost* terhadap $a^{(t)}$ pada setiap *time step* dilakukan karena inilah yang membantu proses *backpropagation*

gradien ke sel RNN sebelumnya [2].

2.2 Penerapan RNN untuk membuat *character-level language model*

Pada bagian ini, ditunjukkan penggunaan RNN untuk membuat *character-level language model* untuk nama dinosaurus. Model tersebut kemudian dapat digunakan untuk membuat nama dinosaurus baru yang nampak seperti nama-nama dinosaurus yang telah ada.

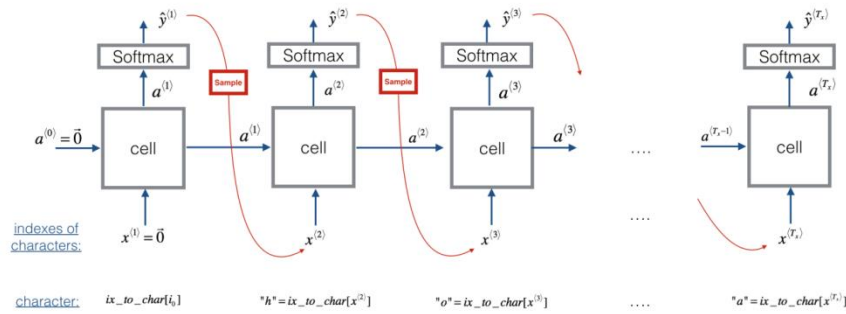
Pertama-tama, dibutuhkan *dataset* berupa nama-nama dinosaurus yang benar-benar ada. Dalam percobaan ini, *dataset* tersebut disimpan dalam suatu *file* dengan format txt. Kemudian, dibuat daftar karakter unik yang ada dalam *dataset* tersebut, dalam percobaan ini ditemukan 27 karakter unik yakni 26 buah alfabet a-z ditambah satu karakter `\n` (*newline character*). Selanjutnya dibuat *python dictionary* yang memetakan setiap karakter yang telah didapat dari tahap sebelumnya ke dalam indeks 0-26. Juga dibuat *dictionary* lainnya yang memetakan setiap indeks terhadap karakter yang diwakilinya.

Selanjutnya, dibuat *character-level language model*. Pembuatan model akan melalui empat tahapan, yakni *forward propagation* dan perhitungan *loss*, *backward propagation* dan perhitungan *gradient loss* terhadap parameter, *gradient clipping*, dan *update parameter* berdasarkan *gradient descent*. *Gradient clipping* perlu dilakukan untuk memastikan bahwa nilai *gradient* tidak "meledak" menjadi nilai yang terlalu besar. Pada percobaan ini diimplementasikan salah satu varian *gradient clipping* yaitu *element wise gradient clipping*. Metode ini akan memaksa nilai *gradient* terletak di antara $[-N, N]$. Secara umum, akan ditentukan nilai ``maxValue`` (misalnya 10). Dalam percobaan ini, jika ada komponen vektor *gradient* bernilai lebih besar dari 10, akan diubah nilainya menjadi 10; dan jika ada komponen vektor *gradient* kurang yang nilainya dari -10, akan diubah nilainya menjadi -10. Jika nilainya berada di antara -10 dan 10, akan dibiarkan apa adanya.

Selanjutnya, dilakukan proses *training* terhadap model. Diberikan *dataset* nama dinosaurus. Setiap baris *dataset* (satu nama) digunakan sebagai satu *training example*.

Setelah model melalui proses *training*, akan dilakukan proses *sampling* terhadap teks yang tersedia untuk menghasilkan teks (nama dinosaurus baru). Proses tersebut dijelaskan pada Gambar 5. Nama dinosaurus yang dihasilkan dari iterasi yang lebih tinggi jumlahnya akan terlihat lebih masuk akal, misalnya cenderung berakhiran, "saurus", "don", "aura", dan "tor".

Nama dinosaurus yang dihasilkan dari iterasi yang lebih tinggi jumlahnya akan terlihat lebih masuk akal, misalnya cenderung berakhiran, "saurus", "don", "aura", dan "tor". Sebagai contoh, pada Tabel 1 diberikan contoh nama-nama dinosaurus yang dihasilkan dari iterasi training pertama, training ke-2000, training ke-10000, training ke-20000, dan training ke-34000.



Gambar 5. Proses sampling setelah model melalui proses *training*. [2]

3 Long Short Term Memory (LSTM)

3.1 Dasar Teori

Long Short Term Memory networks (LSTM) merupakan sebuah evolusi dari arsitektur RNN. RNN memiliki kekurangan, kekurangan itu dapat dilihat pada inputan X_0 , X_1 memiliki rentang informasi yang sangat besar dengan X_b X_{t+1} sehingga ketika h_{t+1} memerlukan informasi yang relevan dengan X_0 , X_1 RNN tidak dapat untuk belajar menghubungkan informasi karena memori lama yang tersimpan akan semakin tidak berguna dengan seiringnya waktu berjalan karena tertimpa atau tergantikan dengan memori baru. Berbeda dengan RNN, LSTM tidak memiliki kekurangan tersebut karena LSTM dapat mengatur memori pada setiap masukannya dengan menggunakan *memory cells* dan *gate units*. LSTM menyimpan informasi terhadap pola-pola pada data. LSTM dapat mempelajari data mana saja yang akan disimpan dan data mana saja yang akan dibuang, karena pada setiap neuron LSTM memiliki beberapa *gates* yang mengatur memori pada setiap neuron itu sendiri [3]. Sehingga, LSTM memiliki kinerja yang lebih baik dalam mengatasi masalah *vanishing gradient* [2]. LSTM banyak digunakan untuk pemrosesan teks, video, dan data *time series* [3].

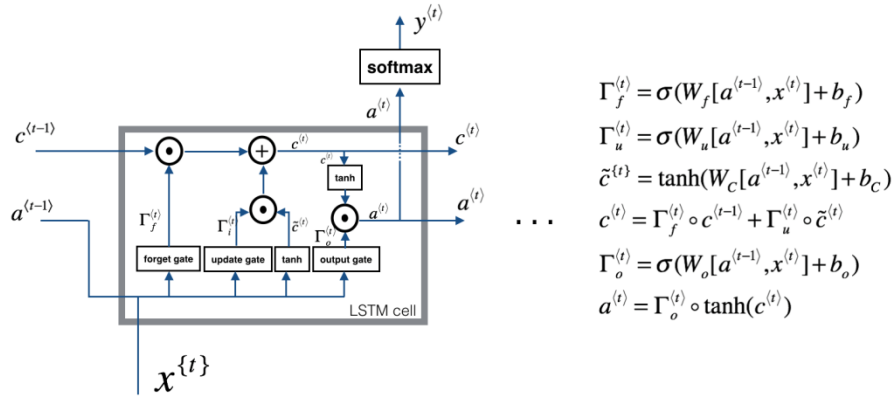
Terdapat empat proses fungsi aktivasi pada setiap masukan pada *neurons* yang selanjutnya disebut sebagai *gates units*. *Gates units* tersebut ialah *forget gates*, *input gates*, *cell gates*, dan *output gates*. Pada *forget gates* informasi pada setiap data masukan akan diolah dan dipilih data mana 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. Pada *input gates* terdapat dua *gates* yang akan dilaksanakan, 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*. 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*. Pada *output gates* terdapat dua *gates* yang akan dilaksanakan, pertama akan diputuskan nilai pada bagian *memory cell* mana yang akan dikeluarkan dengan menggunakan fungsi aktivasi sigmoid. Selanjutnya akan ditempatkan nilai pada *memory cell* dengan menggunakan fungsi aktivasi *tanh*. Terakhir kedua *gates* tersebut dikalikan sehingga menghasilkan nilai yang akan dikeluarkan [3].

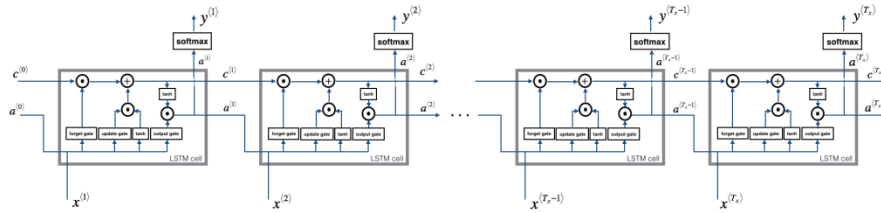
Tabel 1. Perbandingan nama-nama dinosaurus yang dihasilkan dari berbagai jumlah proses training.

| Training ke- | Nama-nama dinosaurus yang dihasilkan |
|--------------|---|
| 1 | Nkzxwdmfqoeyhsqwasjkjvu Kneb Kzxwdmfqoeyhsqwasjkjvu Neb Zxwdmfqoeyhsqwasjkjvu Eb Xwdmfqoeyhsqwasjkjvu |
| 2000 | Liusskeomnolxeros Hmdaairus Hytroligoraus Lecalosapaus Xusicikoraus Abalpsamantisaurus Tpraneronxeros |
| 10000 | Onyusaurus Klecalosaurus Lustodon Ola Xusodonia Eaeosaurus Troceosaurus |
| 20000 | Nousmofonosaurus Loma Lytrognatiasaurus Ngaa Ytroenetiaudostarmilus Eiafosaurus Troenchulunosaurus |
| 34000 | Mavptokekus Ilabaisaurus Itosaurus Macaesaurus Yrosaurus Eiaeosaurus Trodon |

Sama seperti jaringan RNN yang dibentuk dengan melakukan iterasi terhadap operasi-operasi yang terdapat pada setiap sel RNN, jaringan LSTM juga dibentuk dari iterasi operasi-operasi yang ada pada sebuah sel LSTM sesuai dengan jumlah urutan data input [2]. Operasi – operasi yang dilakukan pada satu sel LSTM ditunjukkan pada Gambar 6. Struktur dasar jaringan LSTM dapat dilihat pada Gambar 7.



Gambar 6. Operasi – operasi yang dilakukan pada satu sel LSTM.



Gambar 7. Struktur dasar jaringan LSTM.

Pada LSTM juga dapat dilakukan *backward pass*. Pertama-tama dibuat suatu variabel dengan dimensi yang sama dengan dimensi *return variable*. Kemudian lakukan iterasi terhadap semua *time step* mulai dari akhir dan jalankan fungsi yang diterapkan untuk LSTM di setiap *iterasi*. Kemudian lakukan *parameter update* dengan menjumlahkannya secara individual. Terakhir kembalikan suatu *dictionary* yang berisi gradien baru.

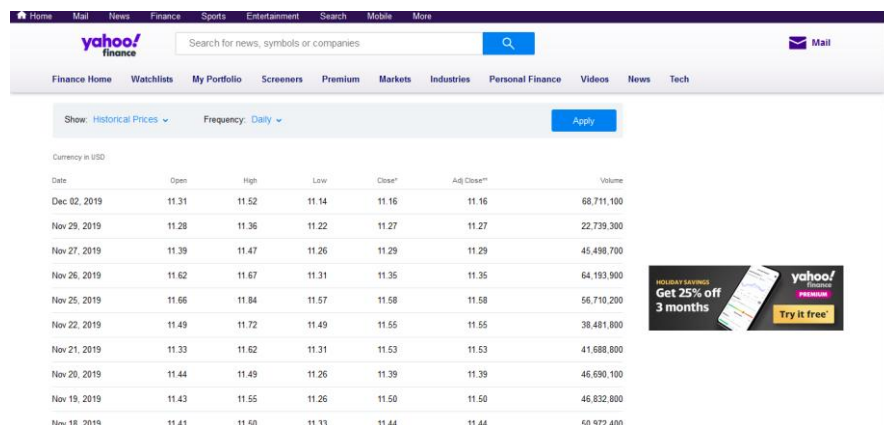
3.2 Penerapan LSTM untuk Prediksi Harga Saham

Pada percobaan ini juga dilakukan percobaan memprediksi harga saham

menggunakan LSTM. Dilakukan percobaan terhadap dua dataset saham, yakni saham GE (General Electric) dan saham TATAGLOBAL. Masing-masing *dataset* memiliki karakteristik tersendiri. *Dataset* saham TATAGLOBAL diperoleh dari modul praktikum dalam bentuk file berformat .csv. *Dataset* tersebut terdiri dari harga saham TATAGLOBAL pada periode 21 Juli 2010 hingga 28 September 2018 (2035 tuple data) untuk data *training* dan harga saham TATAGLOBAL pada periode 1 Oktober 2018 hingga 24 Oktober 2018 (16 tuple data) untuk data *testing*. Sehingga, *dataset* tersebut memiliki rasio *training:testing* sebesar 99,22:0,78. Sedangkan *dataset* saham GE didapat dari situs Yahoo! Finance. Data tersebut memuat pergerakan harga saham GE selama 250 hari kerja, yakni dari tanggal 12 Nopember 2018 hingga tanggal 8 Nopember 2019. Contoh format *dataset* yang didapatkan dari situs Yahoo! Finance dapat dilihat pada Gambar 8. *Dataset* tersebut dipisahkan untuk keperluan *training* dan *testing*, dengan ketentuan sebagai berikut;

- Data 7 Oktober 2019 hingga 8 Nopember 2019 (25 hari) digunakan sebagai data *testing*.
- Data 12 nopember 2018 hingga 4 Oktober 2019 (225 hari) digunakan sebagai data *training*.

Sehingga, *dataset* tersebut memiliki rasio *training:testing* sebesar 90:10. Masing - masing *dataset* kemudian disimpan dalam format CSV.



| Date | Open | High | Low | Close | Adj Close | Volume |
|--------------|-------|-------|-------|-------|-----------|------------|
| Dec 02, 2019 | 11.31 | 11.52 | 11.14 | 11.16 | 11.16 | 68,711,100 |
| Nov 29, 2019 | 11.28 | 11.36 | 11.22 | 11.27 | 11.27 | 22,739,300 |
| Nov 27, 2019 | 11.39 | 11.47 | 11.26 | 11.29 | 11.29 | 45,498,700 |
| Nov 26, 2019 | 11.62 | 11.67 | 11.31 | 11.35 | 11.35 | 64,193,900 |
| Nov 25, 2019 | 11.66 | 11.84 | 11.57 | 11.58 | 11.58 | 56,710,200 |
| Nov 22, 2019 | 11.49 | 11.72 | 11.49 | 11.55 | 11.55 | 38,481,800 |
| Nov 21, 2019 | 11.33 | 11.62 | 11.31 | 11.53 | 11.53 | 41,688,800 |
| Nov 20, 2019 | 11.44 | 11.49 | 11.26 | 11.39 | 11.39 | 46,690,100 |
| Nov 19, 2019 | 11.43 | 11.55 | 11.26 | 11.50 | 11.50 | 46,832,800 |
| Nov 18, 2019 | 11.41 | 11.50 | 11.33 | 11.44 | 11.44 | 50,972,400 |

Gambar 8. Contoh format *dataset* yang didapatkan dari situs Yahoo! Finance.

Untuk mengolah *dataset-dataset training* tersebut menjadi model, pertama-tama dilakukan *feature scaling*. Perlu dilakukan scaling pada masing-masing dataset yang digunakan untuk mendapatkan performa model yang optimal. Pada tugas ini digunakan Scikit-Learn MinMaxScaler untuk memetakan data ke dalam skala antara nol dan satu.

LSTM mensyaratkan data yang digunakan telah dikodekan dalam format tertentu, umumnya array 3D. Pertama-tama, masing-masing dataset dibagi dalam *timesteps* dengan jumlah tertentu dan dikonversi menjadi sebuah array

menggunakan NumPy. Selanjutnya, data dikonversi menjadi *array* 3 dimensi dengan sampel *X_train*, *timestamp* dengan jumlah tertentu, dan satu fitur di setiap *step*. Untuk *dataset* TATAGLOBAL data dibagi dalam 60 *timesteps* sedangkan pada *dataset* GE data dibagi dalam 7 *timesteps* karena jumlah data *training* yang tersedia jauh lebih sedikit.

Untuk membangun LSTM, perlu diimpor beberapa *library* dari Keras yaitu; Sequential untuk membuat *neural network*, Dense untuk membuat *densely connected neural network layer*, LSTM untuk membuat *Long Short-Term Memory layer*, Dropout untuk membuat *dropout layers* yang dapat mencegah *overfitting*.

Selanjutnya, dibuat *LSTM layer* dengan beberapa *argument* yaitu; 50 *units*, yang menggambarkan dimensi *output space*, *return_sequences=True*, yang memerintahkan *output* terakhir dari *output sequence* dijadikan *return value*, dan Input shape yang menggambarkan ukuran training set. Pada argumen pembuatan pada *Dropout layer*, didefinisikan nilai 0,2. Artinya, 20% dari *layer* akan dieliminasi. Kemudian, ditambahkan *Dense layer* yang mendefinisikan *output* dari satu *unit*. Model tersebut *compile* menggunakan *Adam optimizer* dan *loss* dihitung dengan *mean_squared_error*. Model dijalankan sebanyak 100 *epoch* dengan *batch size* sebesar 3 untuk *dataset* GE. Sedangkan untuk *dataset* TATAGLOBAL, model dijalankan sebanyak 100 *epoch* dengan *batch size* sebesar 32.

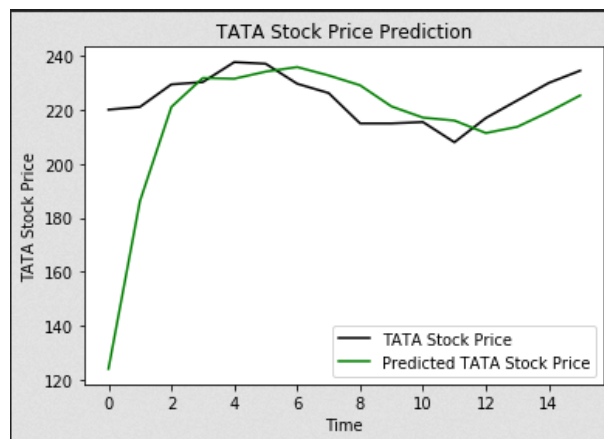
Setelah membangun model, dilakukan pengujian untuk mengetahui kinerja model-model tersebut. Pertama-tama, perlu diimpor *dataset testing* yang akan digunakan untuk membuat prediksi. *Dataset* tersebut memiliki format yang sama dengan *dataset training*. Setelah *dataset* dimuat, gabungkan *training set* dan *test set* pada sumbu 0. Kemudian atur *time step* sebesar 60 untuk *dataset* TATAGLOBAL dan sebesar 7 untuk *dataset* GE, seperti pada tahap sebelumnya. Gunakan *MinMaxScaler* untuk mentransformasikan *dataset* baru tersebut lalu kembalikan bentuk *dataset* menjadi bentuk sebelumnya. Setelah membuat prediksi, gunakan *inverse_transform* untuk mengembalikan harga saham ke dalam format yang mudah dibaca. Perbandingan hasil prediksi saham TATAGLOBAL menggunakan LSTM dan *dataset* saham sebenarnya (*dataset testing*) ditampilkan pada Gambar 9. Perbandingan hasil prediksi saham GE menggunakan LSTM dan *dataset* saham sebenarnya (*dataset testing*) ditampilkan pada Gambar 10.

Pada grafik tersebut dapat dilihat bahwa LSTM dapat memprediksi kenaikan dan penurunan harga saham TATAGLOBAL dan GE. Dengan demikian, dapat disimpulkan bahwa LSTM dapat memprediksi data sekuensial secara akurat.

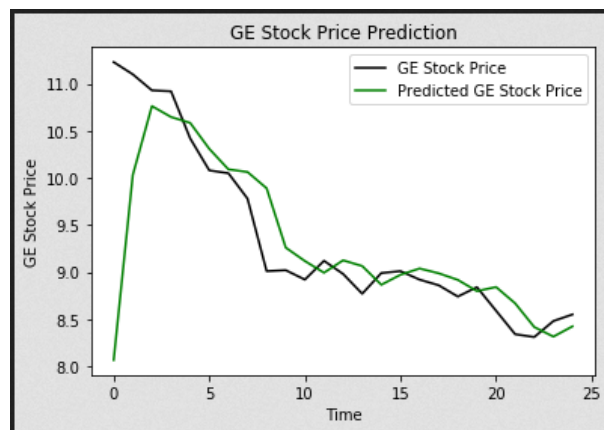
4 Kesimpulan

Dengan menggunakan model-model yang dirancang secara khusus, dapat dilakukan pengolahan data sekuensial dengan hasil yang memuaskan. Pada percobaan ini telah dibuktikan bahwa model RNN dan LSTM telah menunjukkan kinerja yang baik. Pada percobaan *character-level language modelling* menggunakan RNN, setelah melalui iterasi *training* dalam jumlah yang besar,

dapat dihasilkan nama-nama dinosaurus yang memuaskan, yang mendekati nama-nama dinosaurus yang sesungguhnya. Pada percobaan prediksi harga saham dengan LSTM, hasil prediksi LSTM dapat mengikuti kenaikan dan penurunan harga saham yang sebenarnya, baik pada *dataset* saham TATAGLOBAL maupun pada saham GE. Hal ini menunjukkan bahwa model-model yang dirancang khusus untuk data sekuensial dapat mengolah data sekuensial tersebut secara memuaskan.



Gambar 9. Perbandingan hasil prediksi saham TATAGLOBAL menggunakan LSTM dan *dataset* saham sebenarnya (*dataset testing*).



Gambar 10. Perbandingan hasil prediksi saham GE menggunakan LSTM dan *dataset* saham sebenarnya (*dataset testing*).

Referensi

1. Recurrent Neural Network (RNN) – Universitas Gadjah Mada Menara Ilmu Machine Learning, <http://machinelearning.mipa.ugm.ac.id/2018/07/01/recurrent-neural-network-rnn/>
2. Kulbear/deep-learning-coursera: Deep Learning Sprcialization by Andrew Ng on Coursera, <https://github.com/Kulbear/deep-learning-coursera/>
3. Aldi, M.W.P., Jondri, Aditsania, A.: Analisis dan Implementasi Long Short Term Memory Neural Network untuk Prediksi Harga Bitcoin. In: e-Proceeding of Engineering : Vol.5, No.2, pp. 3548–3555. (2018)
4. Pengenalan Recurrent Neural Network (RNN) – Bagian 1 - Belajar Pembelajaran Mesin Indonesia <https://indoml.com/2018/04/04/pengenalan-rnn-bag-1/>