Penggunaan *Artificial Neural Network* pada Data Sekuensial dengan *Recurrent Neural Network* dan *Long Short-Term Memory Network*

Keenan Adiwijaya Leman / 23519018

Abstrak. Penggunaan *neural network architeture* pada data sekuensial membutuhkan arsitektur khusus yang dapat menggunakan kembali informasi yang telah didapatkan sebelumnya dari urutan data-data yang telah lalu. Untuk itu digunakan *network achitecture* khusus, yaitu, *recurrent neural network*. L*ong short-term memory network* sebagai perkembangan dari *recurrent neural network* digunakan ketika proses *training* membutuhkan informasi dari banyak langkah yang lampau. Hasil percobaan dengan membangkitkan kembali keterurutan *training data* yang dimasukkan sebelumnya, menunjukan bahwa *recurrent neural network* berhasil menggunakan kembali informasi yang telah ditemui sebelumnya. Hasil percobaan untuk memprediksi harga harian pada pasar saham, menunjukan bahwa model yang dihasilkan dari pelatihan *long short-term memory network* berhasil memberikan model yang mampu memprediksi tren pasar saham.

1 Pendahuluan

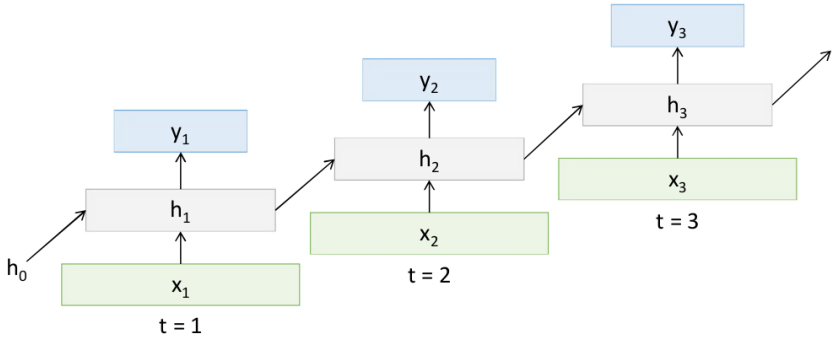
Sekuensial data tidak dapat dimodelkan dengan baik dengan menggunakan *feed forward neural network.* Dengan *recurrent neural network* informasi yang telah diketahui sebelumnya dapat digunakan untuk memprediksi data setelahnya. Adanya struktur keterurutan inilah yang digunakan oleh *recurrent neural network* untuk meningkatkan performa dari model. Walapun *recurrent neural network* dapat dengan baik mengunakan kembali informasi dari data-data sebelumnya, namun, model ini memiliki kekurangan ketika panjang urutan data semakin besar. Dibutuhkan model lain yang mampu menangani urutan data yang panjang tanpa mengalami *exploding gradient* dan *vanishing gradient*. *Long short-term memory network* adalah model yang diusung sebagai perkembangan dari *recurrent neural network* yang mampu menangani urutan data yang panjang dengan baik.

2 Recurrent Neural Network

2.1 Model

Model r*ecurrent neural network* (RNN) bekerja dengan menggunakan keluaran langkah (*time-step*) sebelumnya sebagai masukan pada langkah setelahnya. Sebuah

langkah merujuk kepada proses penghitungan sebuah masukan. Karena masukan dari model RNN adalah data-data yang terurut berdasarkan keterurutan tertentu. Model RNN memproses data tersebut satu persatu seperti halnya *feed-forward neural network* (FFNN). Perbedaannya pada FFNN hasil pemrosesan data pada langkah sebelumnya tidak mempengaruhi pemrosesan pada langkah selanjutnya. Bentuk pengaruh hasil pemrosesan data pada langkah sebelumnya adalah berupa sebuah *hidden state* yang menjadi masukan pada langkah selanjutnya.

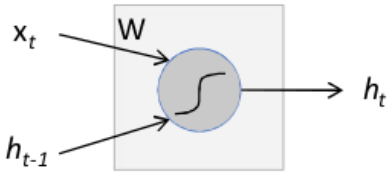


Gambar 1

Gambar 1 menunjukan bagaimana RNN bekerja. Variabel *t* pada gambar diatas menunjukan langkah yang sedang diproses. Variabel *x* menunjukan masukan pada langkah tersebut. Variabel *h* menunjukan sebuah *hidden state* yang dihasilkan pada langkah tersebut. Dan Variabel *y* merupakan keluaran yang dihasilkan dengan memproses *h* pada suatu langkah.

Nilai h_1 dihitung dengan memproses x_1, yang merupakan masukan pada langkah pertama, dan h_0, yang merupakan *hidden state* dari langkah sebelumnya. Namun dalam kasus ini, h_0 parameter, karena x_1 merupakan data pada urutan pertama.

Pada Gambar 1 terlihat bahwa h_2 dihasilkan dari hasil pemrosesan x_2 dan h_1, dengan kata lain nilai h_2 dipengaruhi oleh hasil perhitungan dari langkah selanjutnya.



Gambar 2

Gambar 2 menunjukan bagaimana nilai *ht* yang merupakan *hidden state* langkah ke-*t* dihitung berdasarkan nilai h_{t-1} dan x_t. Seperti halnya gambar sebelumnya, yaitu Gambar 1, Gambar 2 memperlihatkan secara detil bagaimana *hidden state* dihitung. Nilai h_t tidak hanya dihitung dari h_{t-1} dan x_t saja namun terdapat komponen lain yaitu *W* yang merupakan sebuah vektor yang merepresentasikan bobot dari sebuah *layer* dari *neural network*. Persamaan 1 menunjukan perhitungan h_t dengan menggunakan fungsi *hyperbolic tangent* yang bergantung pada nilai *W* dengan parameter x_t dan h_{t-1} untuk menghasilkan *ht*. Pada Persamaan 1, *b* merupakan *bias*.

$$h_t = \tanh\left(\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} W + b\right)$$

Persamaan 1

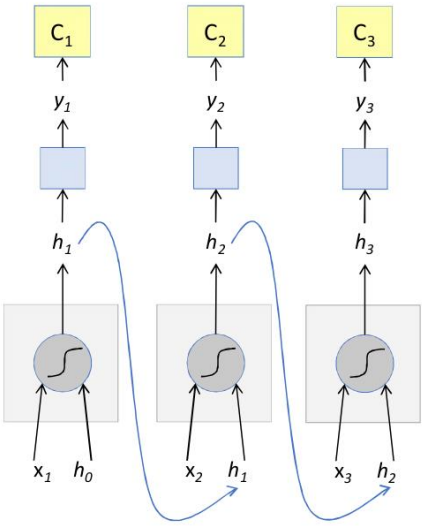
Proses ini dijalankan untuk setiap *training data* yang diberikan. Dan untuk setiap langkah, prediksi keluaran selanjutnya dengan menggunakan fungsi *softmax* seperti yang ditunjukan Persamaan 2. Dengan *Wy* merupakan sebuah vektor yang merepresentasikan sebuah *layer* atau bobot yang digunakan untuk mengubah h_t menjadi y_t, yang merupakan keluaran yang juga merupakan prediksi dari keluaran selanjutnya. Pada Persamaan 2, b_y merupakan *bias*.

$$y_t = softmax(h_t W_y + b_y)$$

Persamaan 2

2.2 Forward Propagation

Proses *forward propagation* pada RNN atau *RNN forward* merupakan proses *recurrent* di mana *layer W* digunakan berulang-ulang dengan masukan dan *hidden state* yang berbeda untuk setiap langkah. Karena proses ini bersifat *recurrent* artinya *layer* atau bobot yang digunakan untuk menghitung setiap *hidden state* merupakan *layer* yang sama, dengan kata lain nilai *W* tidak berubah. Proses ini di-ilustrasikan dengan Gambar 3.



Gambar 3

Pada proses ini, selain penghitungan nilai h_t, hal lain yang juga dilakukan adalah penghitungan y_t yang merupakan keluaran pada langkah ke-*t*. Penghitungan ini dapat dilakukan dengan menggunakan *neural network* lain atau fungsi yang telah disiapkan dengan masukan h_t. Inti dari proses ini adalah mengubah nilai h_t, yang merupakan *hidden state* langkah ke-*t*, menjadi y_t, yang merupakan keluaran langkah ke-*t*. Proses ini ditunjukan dengan kotak biru pada Gambar 3.

2.3 Menghitung Nilai Loss

Nilai *loss* merupakan nilai yang dihitung dari prediksi atau keluaran setiap langkah dan nilai yang seharusnya menurut *training data*. Pilihan fungsi *loss* yang digunakan pada implementasi adalah fungsi *cross-entropy loss*, yang dapat dihitung seperti yang ditunjukan pada Persamaan 3.

$$C = -\frac{1}{N}\sum_{t=1}^{N} y_{at} \log y_t + (1 - y_{at}) \log(1 - y_t)$$

Persamaan 3

Pada Persamaan 3, y_t merupakan prediksi pada langkah ke-*t* dan y_{at} merupakan nilai yang seharusnya pada langkah ke-*t*. Dan nilai *N* merupakan banyaknya *training data*. Sedangkan *C* adalah *loss function*.

Dengan menghitung nilai dari Persamaan 3, akan didapatkan nilai *loss* dari satu iterasi atau satu *epoch*. Nilai *loss* yang telah dihitung dapat digunakan sebagai indikator konvergensi dari model. Semakin kecil nilai *loss*, maka, semakin baik performa model terhadap *training data*.

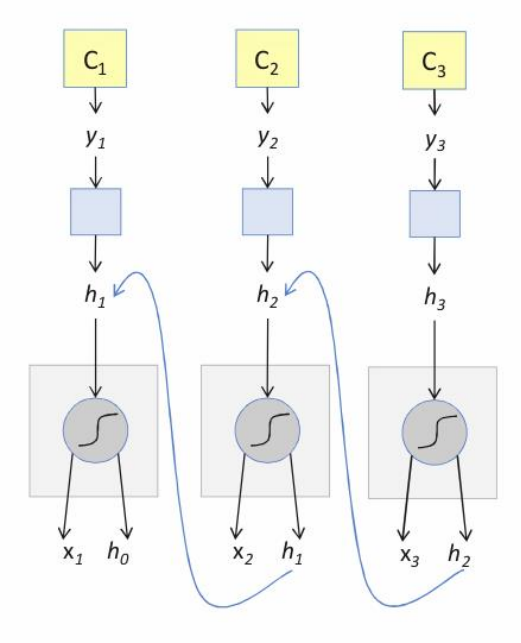
Setelah beberapa *epoch* diharapkan nilai *loss* akan semakin kecil yang berarti performa model membaik dan diharapkan akan konvergen.

2.4 Backpropagation Through Time

Proses *backpropagation* dilakukan setelah proses *RNN forward* dari sebuah *epoch* berakhir. *Backpropagation through time* bergerak dengan arah kebalikan dari langkah atau *time-step* dari *RNN forward*. Pertama, hitunglah *gradient* dari *loss function* terhadap h_n di mana n adalah indeks dari *hidden state* terkahir. Lalu gunakan *gradient* tersebut untuk menghitung *gradient loss function* terhadap h_{n-1}, dan seterusnya hingga h_1. Perhitungan ini berlaku karena h_n merupakan fungsi dari h_{n-1} yang berarti, h_{n-1} digunakan untuk menghitung h_n. Oleh karena itu berlaku aturan penurunan *chain rule* seperti yang ditunjukan oleh Persamaan 4 dan di-ilustrasikan oleh Gambar 4

$$\begin{aligned}\frac{\partial C_t}{\partial h_1} &= \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_1}\right) \\ &= \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_t}\right)\left(\frac{\partial h_t}{\partial h_{t-1}}\right)\cdots\left(\frac{\partial h_2}{\partial h_1}\right)\end{aligned}$$

Persamaan 4



Gambar 4

Pada Persamaan 4, *C* merupakan *loss function* yang digunakan. Sedangkan h_t merupakan *hidden state* pada langkah ke-*t*, dan y_t, merupakan prediksi atau keluaran pada langkah ke-*t*.

2.5 Gradient Clipping

Teknik *gradient clipping* digunakan agar *gradient* yang dihasilkan tidak menjadi terlalu besar atau pun terlalu kecil. Jika *gradient* terlalu besar maka model akan sulit untuk konvergen karena parameter yang dioptimasi berubah terlalu besar. Jika *gradient* terlalu kecil maka model akan lambat untuk konvergen, karena perubahan parameter terlalu kecil. *Gradient clipping* dapat dilakukan dengan menggunakan perhitungan yang ditunjukan oleh Persamaan 5.

$$clip(x,a,b) = \begin{cases} a, & x < a \\ x, & a \leq x \leq b \\ b, & x < b \end{cases}$$

Persamaan 5

Persamaan 5 melakukan *clipping* dengan membatasi nilai *x* agar tetap berada pada rentang *a* sampai dengan *b*. Jika nilai *x* kurang *a* maka fungsi akan mengambalikan *a*. Jika nilai *x* lebih dari *b* maka fungsi akan mengembalikan *b*. Jika nilai *x* lebih besar sama dengan *a* dan lebih kecil sama dengan *b*, maka, fungsi akan

mengembalikan *x*.

3 Studi Kasus RNN: Dinosaurus Land

3.1 Permasalahan

Pada permasalahan ini digunakan sebuah *dataset* berisi nama-nama dinosaurus sebanyak 1536 nama. Keluaran yang di-inginkan adalah sebuah model yang dapat mengeluarkan atau membangkitkan nama-nama yang mirip seperti nama-nama dinosaurus pada *dataset*. *Dataset* dapat dilihat pada *file* "dino.txt".

3.2 Garis Besar Pengerjaan

Menggunakan model RNN, akan dilakukan *training* terhadap setiap nama yang berada pada *dataset*. Setiap nama akan dimodelkan seolah-olah merupakan data sekuensial. Urutan karakter pada nama-nama dinosaurus pada *dataset* adalah yang akan dicoba untuk diprediksi oleh model. Dengan melakukan *sampling* per karakter, diharapkan model dapat menghasilkan nama-nama yang mirip seperti nama-nama di dalam *dataset*.

3.3 Gradient Clipping

Pada kasus ini, digunakan teknik *gradient clipping* untuk mencegah *exploding gradient* dan *vanishing gradient*. Parameter *a*, yang merupakan batas bawah *clipping*, bernilai -5. Dan parameter *b* yang merupakan batas atas *clipping*, bernilai 5. Proses *clipping* dihitung dengan Persamaan 5. Proses *gradient clipping* dilakukan pada setelah *backpropagation through time* dilakukan, parameter diperbaharui dengan *gradient* yang telah melalu proses *clipping*.

3.4 Hasil

Tabel 1 merupakan hasil *sampling* 7 nama dari model yang telah di-*training* sebanyak 0 iterasi, 2000 iterasi, 20000 iterasi, dan 34000 iterasi. Pada hasil lengkap yang berada pada *file* "Dinosaurus Land.ipynb" terdapat nama-nama dinosaurus yang dihasilkan setiap 2000 iterasi. Namun, tren yang jelas terlihat dengan mencermati nama-nama dinosaurus pada setelah 0, 2000, 20000, dan 34000 iterasi. Nama dinosaurus yang dihasilkan model semakin mirip dengan nama-nama dinosaurus pada *file* masukan, yaitu *file* "dino.txt", seiring bertambahnya iterasi yang telah dilakukan model.

Nama Dinosaurus	**Banyak Iterasi**
Nkzxwtdmfqoeyhsqwasjkjvu	0

Kneb Kzxwtdmfqoeyhsqwasjkjvu Neb Zxwtdmfqoeyhsqwasjkjvu Eb Xwtdmfqoeyhsqwasjkjvu	
Liusskeomnolxeros Hmdaairus Hytroligoraurus Lecalosapaus Xusicikoraurus Abalpsamantisaurus Tpraneronxeros	2000
Nkwusaurus Loha Lyusaurus Necalosaurus Yusochosaurus Eiaeros Trrandon	20000
Matrus Ineca Jusplbianiangosaurus Macaestegantitan Ytosaurus Elaepteka Trodon	34000

Tabel 1

4 Long Short-Term Memory Network

4.1 LSTM vs RNN: Analisis Exploding dan Vanishing Gradient

Model *Long Short-Term Memory Network* (*LSTM Network*) bekerja dengan data sekuensial seperti RNN, tetapi dibandingkan dengan RNN, LSTM tidak terlalu dipengaruhi oleh *vanishing gradient* ataupun *exploding gradient*. Artinya, propagasi *gradient* pada model LSTM tidak mudah untuk menjadi terlalu besar atau terlalu kecil.

Pada RNN, seperti yang telah dijelaskan sebelumnya, menggunakan teknik *gradient clipping* untuk mencegah *vanishing gradient* dan *explosing gradient*. Walaupun teknik ini berhasil mencegah kedua masalah tersebut, teknik *clipping gradient* membuat RNN tidak dapat menyimpan informasi dari langkah-langkah yang jauh ke belakang, karena nilai tersebut akan dipotong pada proses *clipping*. Terdapat

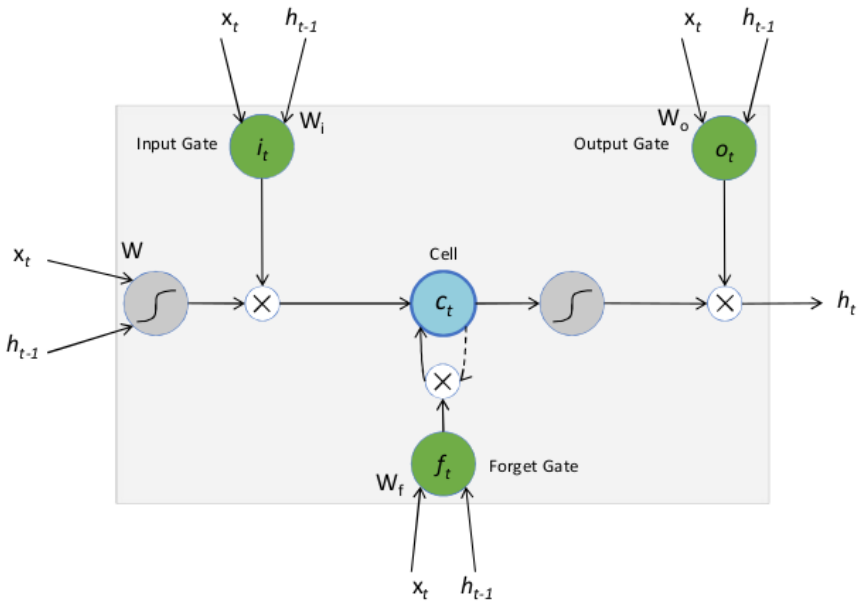
trade-off pada penggunaan teknik *clipping gradient*.

Pada kasus Dinosaurus Land, teknik *gradient clipping* berkerja dengan baik, dan memberikan hasil yang cukup memuaskan. Analisis selanjutnya mungkin akan memberikan mengapa untuk kasus Dinosaurus Land RNN dengan *gradient clipping* bekerja dengan baik. Permasalahan pada kasus Dinosaurus Land, berkutat pada *sampling* dari distribusi kemungkinan karakter selanjutnya pada suatu nama dinosaurus yang hendak dibangkitkan. Oleh karena itu, proses *training* adalah dengan melakukan *fitting* terhadap distribusi kemungkinan urutan karakter pada nama-nama dinosaurus pada *dataset.*

Nama dinosaurus pada *dataset* kasus Dinosaurus Land tidaklah begitu panjang, sehingga RNN tidak ada kebutuhan model untuk mampu mengingat informasi yang ditemukan pada langkah yang sangat jauh ke belakang. Oleh karena itu, RNN dengan *gradient clipping* bekerja dengan baik untuk membatasi nilai *gradient* tanpa kehilangan terlalu banyak informasi. Namun tidak semua masalah memiliki kebutuhan model yang sama.

4.2 Model

Setiap *cell* LSTM bekerja dengan menggunakan tiga buah *layer* yang memiliki fungsi yang berbeda-beda. RNN bekerja dengan hanya satu buah *layer*, yaitu *layer hyperbolic tangent* dengan masukan *hidden state* langkah sebelumnya, dan masukan data pada langkah tersebut. LSTM bekerja dengan prinsip yang sama dengan RNN, yaitu operasi *recurrent* pada data sekuensial. Hanya saja, LSTM menggunakan tiga buah *layer* dibandingkan RNN yang hanya menggunakan satu buah *layer*.



Gambar 5

Gambar 5 menunjukan sebuah *cell* LSTM. Sebuah *cell* LSTM memiliki 4 buah *layer*, *layer W*, *layer* W_i, *layer* W_f, *dan layer* W_o. Pada LSTM, terdapat 3 buah *layer* yang bekerja sebagai *gate*. Sebuah *gate* digunakan sebagai pengaturan terhadap perubahan *cell state* yang didenotasikan sebagai c_t dan seberapa besar pengaruh c_t dalam menghitung nilai keluaran h_t. Pada Gambar 5, variabel *t* menandakan langkah ke-*t*. Setiap langkah (*time-step*) pada LSTM, akan menghasilkan dua nilai, yaitu c_t sebagai *cell state* dari langkah ke-*t*, Dan h_t sebagai keluaran dari langkah ke-*t*.

4.3 Penghitungan Cell State

Gate W_i, disebut juga *input gate*, digunakan untuk menghitung i_t, yang berguna untuk mengatur besar pengaruh masukan x_t terhadap nilai c_t. Sedangkan nilai i_t dihitung berdasarkan berdasarkan h_{t-1}, x_t, dan W_i. Dengan kata lain, masukan x_t secara langsung diatur pengaruhnya berdasarkan keluaran langkah sebelumnya h_{t-1}, masukan langkah ke-*t* itu sendiri yaitu x_t, dan *gate* W_i. Sebuah *gate* tidak serta merta selesai setelah melalui sebuah *layer* melainkan harus diterapkan kepada nilai yang ingin diatur. Pada *gate* W_i, hal ini dilakukan dengan menghitung *pointwise product* i_t bersama dengan keluaran *layer W*.

Selain dipengaruhi nilai i_t, penghitungan nilai c_t juga dipengaruhi oleh nilai f_t. Berbeda dengan nilai i_t yang digunakan sebagai ukuran seberapa besar nilai masukan x_t yang harus di-ingat, nilai f_t digunakan sebagai ukuran seberapa besar nilai c_{t-1} yang harus di-ingat. Nilai f_t, dihitung berdasarkan *gate* W_f, nilai h_{t-1}, dan

nilai x_t. Semakin besar f_t menandakan *cell state* akan di-ingat atau disimpan lebih baik, dan sebaliknya, semakin kecil nilai f_t maka semakin banyak informasi yang dilupakan, oleh karena itu *gate* W_f disebut juga sebagai *forget gate*.

Kedua nilai f_t dan i_t, digunakan untuk menghitung *cell state* yang baru. Persamaan 6, menunjukan bagaimana nilai i_t dihitung, dengan b_i merupakan bias. Dan, Persamaan 7, menunjukan bagaimana nilai f_t dihitung, dengan b_t merupakan bias.

$$i_t = \sigma\left(W_i \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_i\right)$$

Persamaan 6

$$f_t = \sigma\left(W_f \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f\right)$$

Persamaan 7

Untuk menghitung nilai c_t digunakanlah Persamaan 8. Pada Persamaan 8, terlihat bahwa operasi *pointwise product* f_t terhadap c_{t-1} berguna untuk mengatur seberapa besar c_{t-1} yang harus di-ingat. Dan, terlihat juga bahwa operasi *pointwise product* i_t terhadap $\tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$ berguna untuk mengatur seberapa besar pengaruh x_t terhadap c_t.

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

Persamaan 8

4.4 Penghitungan Output Value

Penghitungan keluaran langkah ke-*t* secara langsung dipengaruhi oleh *cell state* pada langkah tersebut, yaitu nilai c_t, dan nilai o_t. Nilai o_t digunakan untuk mengatur seberapa besar pengaruh nilai c_t mempengaruhi penghitungan c_{t+1} pada langkah selanjutnya melalui nilai h_t. Dapat dicermati pada Persamaan 6 dan Persamaan 7, bahwa nilai h_{t-1} mempengaruhi penghitungan i_t dan f_t, yang berarti secara tidak langsung mempengaruhi penghitungan nilai c_t. Dan nilai h_{t-1} juga mempengaruhi penghitungan c_t secara langsung pada Persamaan 8.

Nilai h_t dihitung dengan Persamaan 9, di mana, nilai o_t bersama dengan nilai c_t yang di-distribusikan dengan fungsi *hyperbolic tangent* akan dioperasikan dengan sebuah operator *pointwise product*. Persamaan 9 menunjukan bagaimana nilai o_t mempengaruhi nilai h_t.

$$h_t = o_t \otimes \tanh c_t$$

Persamaan 9

Sedangkan nilai o_t sendiri dihitung seperti halnya dengan nilai i_t dan nilai f_t, menggunakan sebuah *gate*, dalam kasus ini *gate* yang digunakan adalah *output gate*, dinotasikan dengan W_o. Persamaan 10, menunjukan bagimana o_t dihitung, dengan b_o sebagai *bias*.

$$o_t = \sigma\left(W_o\begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_o\right)$$

Persamaan 10

5 Studi Kasus LSTM: Harga Saham

5.1 Permasalahan

Pada permasalahan ini digunakan dua buah *dataset* berbeda yang berisi harga-harga saham harian. Dua *dataset* yang digunakan adalah harga saham perusahaan Tata, dan Dow Jones Industrial Average (DJI). Keluaran yang di-inginkan adalah prediksi harga harian selanjutnya.

Dataset untuk harga saham Tata dibagi menjadi dua buah *file*, yaitu "NSE-TATAGLOBAL.csv", yang merupakan *training set* dan *file* "tatatest.csv", yang merupakan *test set*. *Dataset* untuk harga DJI juga dibagi menjadi dua buah *file*, yaitu "DJI-train.csv", yang merupakan *training set* dan "DJI-test.csv", yang merupakan *test set*.

5.2 Garis Besar Pengerjaan

Dua buah model LSTM terpisah akan di-*train* mengunakan masing-masing *training set*. Selanjutnya, *test set* masing-masing *dataset* akan digunakan sebagai pembanding antara harga-harga yang diprediksi dan harga-harga yang sebenarnya. Pada kasus ini, *library* Keras digunakan untuk mengimplementasikan model LSTM. Langkah *preprocessing* dan *network architecture* yang digunakan akan dijelaskan pada bagian-bagian selanjutnya. Selain itu, ditentukan pula *regularization* yang digunakan adalah *drop-out regularization*. Untuk percobaan ini digunakan 1000 *epoch* dengan 2035 *training data* per *epoch* untuk harga saham Tata, dan 800 *training data* per *epoch* untuk harga DJI.

5.3 Preprocessing

Baik *training set* maupun *test set*, sebelum digunakan, terlebih dahulu dilakukan *preprocessing*. Harga saham harian akan diubah menjadi nilai dengan rentang 0 sampai dengan 1. Hal ini dilakukan sebagai normalisasi terhadap masukan yang

dapat mempercepat model untuk konvergen, dengan kata lain, performa yang lebih baik untuk banyak *epoch* yang sama.

5.4 Network Architecture

Network architecture yang digunakan untuk menyelesaikan masalah ini terdiri dari 5 buah *layer* yang di mana 4 *layer* pertama masing-masing mengeluarkan keluaran berdimensi 50. Dan untuk setiap 4 *layer* LSTM tersebut, akan dilakukan *drop-out regularization* setelahnya dengan 20% masukan yang akan dibuang secara acak. *Layer* ke-5 merupakan *fully-connected neural network* yang berisi 1 buah *unit*, yang digunakan sebagai *output layer*.

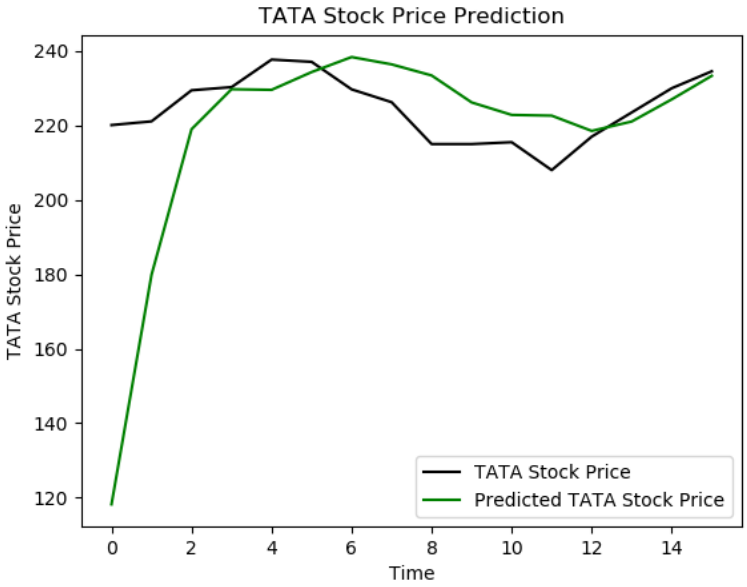
5.5 Regularization

Drop-out regularization digunakan untuk mengurangi resiko terjadinya *exploding gradient* dan *vanishing gradient* pada *dataset* yang sangat panjang. Walaupun, LSTM lebih tahan terhadap kedua masalah tersebut dari pada RNN, namun dengan *regularization* diharapkan *gradient* tidak menjadi terlalu besar ataupun terlalu kecil. *Gradient* yang terlalu besar akan mengakibatkan model sulit untuk konvergen. Sedangkan *gradient* yang terlalu kecil akan mengakibatkan model lambat untuk konvergen atau membutuhkan lebih banyak *epoch* untuk mencapi performa yang di-inginkan.

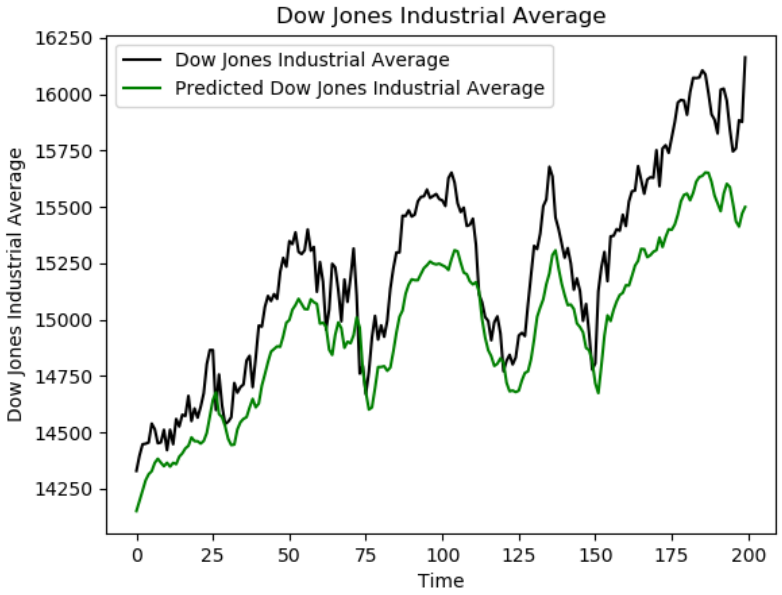
5.6 Hasil

Gambar 6 menunjukan perbandingan harga hasil prediksi model dengan harga yang sebenarnya untuk *dataset* harga saham Tata. Pada ujicoba terhadap harga saham Tata, prediksi model tidak terlalu dapat menyamai tren dari kenaikan ataupun penurunan harga. Namun, harga yang diprediksi masih cukup dekat dengan harga yang sebenarnya. Pada akhir grafik pada Gambar 6 terlihat bahwa harga hasil prediksi dan harga sebenarnya hampir sama.

Pada ujicoba terhadap harga DJI, Gambar 7 menunjukan bagaimana perbandingan antara harga hasil prediksi dengan harga yang sebenarnya. Pada Gambar 7, terlihat bahwa harga hasil prediksi tidak dapat menyamai harga yang sebenarnya, namun model dengan baik dapat memprediksi tren *bullish* atau *bearish* pada harga saham. Dengan kata lain, model dapat memprediksi tren naik turunnya harga sehingga dapat menjadi indikator beli atau jual.



Gambar 6



Gambar 7

6 Kesimpulan

Dalam memproses data sekuensial dengan menggunakan *artificial neural network*, arsitektur khusus dibutuhkan guna menggunakan kembali informasi yang telah dihitung pada urutan sebelumnya.

Pada proses *training* data sekuensial resiko terjadinya *exploding gradient* dan *vanishing gradient* semakin besar sering banyaknya langkah yang dilakukan. Pada RNN, digunakan teknik *gradient clipping* guna menghindari kedua masalah tersebut. Walaupun teknik ini berhasil menjaga nilai *gradient* untuk tetap pada rentang yang di-inginkan, pada data sekuensial yang sangat panjang, teknik ini akan membuat RNN tidak dapat menyimpan informasi yang didapatkan dari perhitungan dilakukan jauh sebelumnya.

RNN dengan *gradient clipping* berhasil membangkitkan kembali urutan karakter yang berada pada *training data* dengan melakukan *samping* terhadap distribusi urutan karakter hasil *training* dengan menggunakan fungsi *softmax*. Keberhasilan RNN untuk membangkitkan kembali urutan karakter membuktikan bahwa untuk data sekuensial yang relatif tidak terlalu panjang, model berhasil menggunakan kembali informasi yang telah dihitung pada langkah (*time-step*) sebelumnya.

Arsitekur LSTM, yang merupakan perkembangan dari RNN, dengan menggunakan tiga buah nilai f_t, o_t, dan i_t, mampu menjaga rentang *gradient* tanpa kehilangan informasi yang dianggap penting untuk digunakan kembali dilangkah yang jauh setelahnya. LSTM dengan menggunakan *forget gate*, menghasilkan nilai f_t, menggunakan *output gate* untuk menghasilkan o_t, dan *input gate* untuk menghasilkan i_t, berhasil membuat model yang mampu memprediksi tren harga pasar saham.

References

1. Slide Perkuliahan IF5180. RNN Basic
2. Slide Perkuliahan IF5180. RNN Hands On
3. Slide Perkuliahan IF5180. Pasar Modal Basic
4. Slide Perkuliahan IF5180. Pasar Modal Basic Prediction LSTM
5. Bahan Perkuliahan IF5180. Character Level Modeling – Dinosaurus Land
6. "Understanding LSTM Networks." *Understanding LSTM Networks -- Colah's Blog*, https://colah.github.io/posts/2015-08-Understanding-LSTMs