Laporan Mata Kuliah Pengenalan Pola Mengenai RNN dan LSTM

Giffari Alfarizy[1]

[1]Teknik Informatika, STEI, Institut Teknologi Bandung, Indonesia
23519024@std.stei.itb.ac.id

Abstrak. Recurrent Neural Networks (RNN) sangat efektif untuk Natural Language Processing dan pekerjaan sekuens lainnya karena RNN memiliki "memori". Keberhasilan RNN didukung dengan penggunaan Long Short-Term Memory (LSTM) yang bekerja lebih baik dari versi standarnya. Menggunakan metode RNN dan LSTM bekerja dengan baik dalam menghasilkan nama dinosaurus baru dari kumpulan nama-nama dinosaurus yang ada dan dapat memprediksi pergerakan saham TATA dan Asus.

1 Pendahuluan

Recurrent Neural Networks (RNN) sangat efektif untuk Natural Language Processing dan pekerjaan sekuens lainnya karena RNN memiliki "memori". RNN dapat membaca input x(t) (misalnya kata-kata) satu per satu, dan mengingat beberapa informasi/konteks melalui hidden layer activations yang dilewati dari suatu time-step menuju time-step berikutnya.

Dalam beberapa tahun ini, banyak aplikasi RNN yang berhasil dalam beberapa persoalan seperti speech recognition, language modeling translation, dan image captioning. Keberhasilan ini didukung dengan penggunaan Long Short-Term Memory (LSTM) yang merupakan jenis khusus dari RNN yan bekerja untuk banyak pekerjaan dengan hasil yang lebih baik daripada versi standarnya[1].

2 Dasar Teori

2.1 Recurrent Neural Networks (RNN)

RNN adalah salah satu bagian dari keluarga Neural Network untuk memproses data yang bersambung (sequential data). 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[2].

Tahap-tahap implemetasi RNN adalah sebagai berikut:

1. Implementasi perhitungan yang diperlukan untuk satu time-step RNN
2. Implementasi perulangan terhadap TxTx time-step untuk memproses keseluruhan input satu per satu.

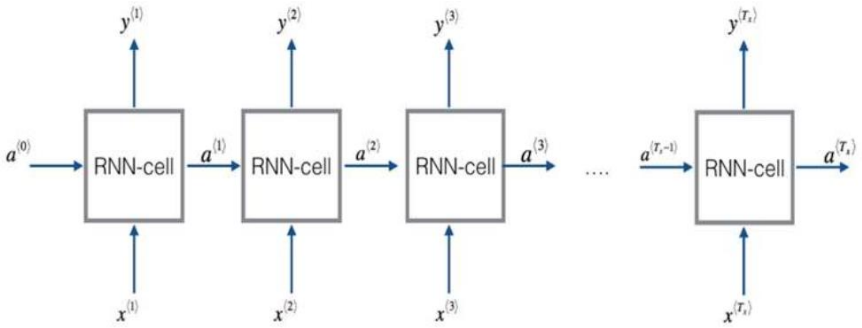

Fig. 2.1.1. Basic RNN dasar yang diimplementasikan dalam percobaan ini

Recurrent Neural Network dapat dipandang sebagai perulangan dari sebuah single cell. Pertama dilakukan implementasi komputasi untuk satu time-step. Gambar berikut mendeskripsikan operasi untuk setiap satu time-step dalam sebuah cell RNN.

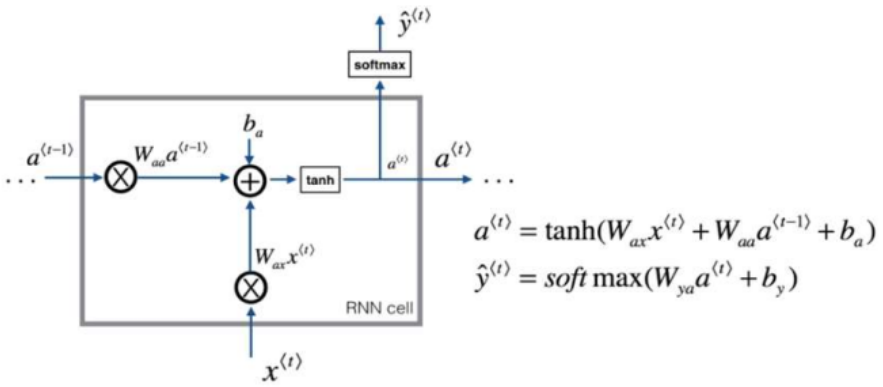

Fig. 2.1.2. Operasi pada satu time-step dari sebuah RNN cell

2.2 Long Short-Term Memory Network

Long Short-Term Memory networks - biasanya disebut "LSTM" - adalah jenis khusus RNN, yang mampu mempelajari long-term. Mereka Diperkenalkan oleh

Hochreiter & Schmidhuber (1997), disempurnakan dan dipopulerkan oleh banyak pengikutnya. LSTM bekerja sangat baik pada berbagai macam masalah dan sekarang banyak digunakan.

LSTM dirancang secara eksplisit untuk menghindari masalah long-term dependency. Mengingat informasi untuk jangka waktu yang lama adalah behaviour standar LSTM

3 Hasil Program

Berikut adalah hasil dari masing-masing program.

3.1 RNN_HandsOn

```
a_next[4] =  [ 0.59584544  0.18141802  0.61311866  0.99808218  0.85016201  0.99980978
 -0.18887155  0.99815551  0.6531151   0.82872037]
a_next.shape =  (5, 10)
yt_pred[1] =  [0.9888161  0.01682021 0.21140899 0.36817467 0.98988387 0.88945212
 0.36920224 0.9966312  0.9982559  0.17746526]
yt_pred.shape =  (2, 10)
```

Fig. 3.1.1. Hasil keluaran rnn_cell_forward

```
a[4][1] =  [-0.99999375  0.77911235 -0.99861469 -0.99833267]
a.shape =  (5, 10, 4)
y_pred[1][3] =  [0.79560373 0.86224861 0.11118257 0.81515947]
y_pred.shape =  (2, 10, 4)
caches[1][1][3] =  [-1.1425182  -0.34934272 -0.20889423  0.58662319]
len(caches) =  2
```

Fig. 3.1.2. Hasil keluaran rnn_forward

```
a_next[4] =  [-0.66408471  0.0036921   0.02088357  0.22834167 -0.85575339  0.00138482
  0.76566531  0.34631421 -0.00215674  0.43827275]
a_next.shape =  (5, 10)
c_next[2] =  [ 0.63267805  1.00570849  0.35504474  0.20690913 -1.64566718  0.11832942
  0.76449811 -0.0981561  -0.74348425 -0.26810932]
c_next.shape =  (5, 10)
yt[1] =  [0.79913913 0.15986619 0.22412122 0.15606108 0.97057211 0.31146381
 0.00943007 0.12666353 0.39380172 0.07828381]
yt.shape =  (2, 10)
cache[1][3] =  [-0.16263996  1.03729328  0.72938082 -0.54101719  0.02752074 -0.30821874
  0.07651101 -1.03752894  1.41219977 -0.37647422]
len(cache) =  10
```

Fig. 3.1.3. Hasil keluaran lstm_cell_forward

```
a[4][3][6] =  0.17211776753291672
a.shape =  (5, 10, 7)
y[1][4][3] =  0.9508734618501101
y.shape =  (2, 10, 7)
caches[1][1][1] =  [ 0.82797464  0.23009474  0.76201118 -0.22232814 -0.20075807  0.18656139
  0.41005165]
c[1][2][1] =  -0.8555449167181981
len(caches) =  2
```

Fig. 3.1.4. Hasil keluaran lstm_forward

```
gradients["dxt"][1][2] = -0.4605641030588796
gradients["dxt"].shape = (3, 10)
gradients["da_prev"][2][3] = 0.08429686538067718
gradients["da_prev"].shape = (5, 10)
gradients["dWax"][3][1] = 0.3930818739219303
gradients["dWax"].shape = (5, 3)
gradients["dWaa"][1][2] = -0.2848395578696067
gradients["dWaa"].shape = (5, 5)
gradients["dba"][4] = [0.80517166]
gradients["dba"].shape = (5, 1)
```

Fig. 3.1.5. Hasil keluaran rnn_cell_backward

```
gradients["dx"][1][2] = [-2.07101689 -0.59255627  0.02466855  0.01483317]
gradients["dx"].shape = (3, 10, 4)
gradients["da0"][2][3] = -0.31494237512664996
gradients["da0"].shape = (5, 10)
gradients["dWax"][3][1] = 11.264104496527777
gradients["dWax"].shape = (5, 3)
gradients["dWaa"][1][2] = 2.303333126579893
gradients["dWaa"].shape = (5, 5)
gradients["dba"][4] = [-0.74747722]
gradients["dba"].shape = (5, 1)
```

Fig. 3.1.6. Hasil keluaran rnn_backward

```
gradients["dxt"][1][2] = 3.2305591151091884
gradients["dxt"].shape = (3, 10)
gradients["da_prev"][2][3] = -0.06396214197109239
gradients["da_prev"].shape = (5, 10)
gradients["dc_prev"][2][3] = 0.7975220387970015
gradients["dc_prev"].shape = (5, 10)
gradients["dWf"][3][1] = -0.1479548381644968
gradients["dWf"].shape = (5, 8)
gradients["dWi"][1][2] = 1.0574980552259903
gradients["dWi"].shape = (5, 8)
gradients["dWc"][3][1] = 2.304562163687667
gradients["dWc"].shape = (5, 8)
gradients["dWo"][1][2] = 0.3313115952892109
gradients["dWo"].shape = (5, 8)
gradients["dbf"][4] = [0.18864637]
gradients["dbf"].shape = (5, 1)
gradients["dbi"][4] = [-0.40142491]
gradients["dbi"].shape = (5, 1)
gradients["dbc"][4] = [0.25587763]
gradients["dbc"].shape = (5, 1)
gradients["dbo"][4] = [0.13893342]
gradients["dbo"].shape = (5, 1)
```

Fig. 3.1.7. Hasil keluaran lstm_cell_backward

3.2 Latihan Mandiri – Character-Level Language Modeling

```
Iteration: 0, Loss: 23.087336

Nkzxwtdmfqoeyhsqwasjkjvu
Kneb
Kzxwtdmfqoeyhsqwasjkjvu
Neb
Zxwtdmfqoeyhsqwasjkjvu
Eb
Xwtdmfqoeyhsqwasjkjvu
```

Fig. 3.2.1. Hasil generate nama dinosaurus pada iterasi 0 pada Character-Level Language Modeling

```
Iteration: 2000, Loss: 27.884160

Liusskeomnolxeros
Hmdaairus
Hytroligoraurus
Lecalosapaus
Xusicikoraurus
Abalpsamantisaurus
Tpraneronxeros
```

Fig. 3.2.2. Hasil generate nama dinosaurus pada iterasi 2000 pada Character-Level Language Modeling

```
Iteration: 4000, Loss: 25.901815

Mivrosaurus
Inee
Ivtroplisaurus
Mbaaisaurus
Wusichisaurus
Cabaselachus
Toraperlethosdarenitochusthiamamumamaon
```

Fig. 3.2.3. Hasil generate nama dinosaurus pada iterasi 4000 pada Character-Level Language Modeling

```
Iteration: 6000, Loss: 24.608779

Onwusceomosaurus
Lieeaerosaurus
Lxussaurus
Oma
Xusteonosaurus
Eeahosaurus
Toreonosaurus
```

Fig. 3.2.4. Hasil generate nama dinosaurus pada iterasi 6000 pada Character-Level Language Modeling

```
Iteration: 8000, Loss: 24.070350

Onxusichepriuon
Kilabersaurus
Lutrodon
Omaaerosaurus
Xutrcheps
Edaksoje
Trodiktonus
```

Fig. 3.2.5. Hasil generate nama dinosaurus pada iterasi 8000 pada Character-Level Language Modeling

```
Iteration: 10000, Loss: 23.844446

Onyusaurus
Klecalosaurus
Lustodon
Ola
Xusodonia
Eeaeosaurus
Troceosaurus
```

Fig. 3.2.6. Hasil generate nama dinosaurus pada iterasi 10000 pada Character-Level Language Modeling

```
Iteration: 12000, Loss: 23.291971

Onyxosaurus
Kica
Lustrepiosaurus
Olaagrraiansaurus
Yuspangosaurus
Eealosaurus
Trognesaurus
```

Fig. 3.2.7. Hasil generate nama dinosaurus pada iterasi 12000 pada Character-Level Language Modeling

```
Iteration: 14000, Loss: 23.382339

Meutromodromurus
Inda
Iutroinatorsaurus
Maca
Yusteratoptititan
Ca
Troclosaurus
```

Fig. 3.2.8. Hasil generate nama dinosaurus pada iterasi 14000 pada Character-Level Language Modeling

```
Iteration: 16000, Loss: 23.288447

Meuspsangosaurus
Ingaa
Iusosaurus
Macalosaurus
Yushanis
Daalosaurus
Trpandon
```

Fig. 3.2.9. Hasil generate nama dinosaurus pada iterasi 16000 pada Character-Level Language Modeling

```
Iteration: 18000, Loss: 22.823526

Phytrolonhonyg
Mela
Mustrerasaurus
Peg
Ytronorosaurus
Ehalosaurus
Trolomeehus
```

Fig. 3.2.10. Hasil generate nama dinosaurus pada iterasi 18000 pada Character-Level Language Modeling

```
Iteration: 20000, Loss: 23.041871

Nousmofonosaurus
Loma
Lytrognatiasaurus
Ngaa
Ytroenetiaudostarmilus
Eiafosaurus
Troenchulunosaurus
```

Fig. 3.2.11. Hasil generate nama dinosaurus pada iterasi 20000 pada Character-Level Language Modeling

```
Iteration: 22000, Loss: 22.728849

Piutyrangosaurus
Midaa
Myroranisaurus
Pedadosaurus
Ytrodon
Eiadosaurus
Trodoniomusitocorces
```

Fig. 3.2.12. Hasil generate nama dinosaurus pada iterasi 22000 pada Character-Level Language Modeling

```
Iteration: 24000, Loss: 22.683403

Meutromeisaurus
Indeceratlapsaurus
Jurosaurus
Ndaa
Yusicheropterus
Eiaeropectus
Trodonasaurus
```

Fig. 3.2.13. Hasil generate nama dinosaurus pada iterasi 24000 pada Character-Level Language Modeling

```
Iteration: 26000, Loss: 22.554523

Phyusaurus
Liceceron
Lyusichenodylus
Pegahus
Yustenhtonthosaurus
Elagosaurus
Trodontonsaurus
```

Fig. 3.2.14. Hasil generate nama dinosaurus pada iterasi 26000 pada Character-Level Language Modeling

```
Iteration: 28000, Loss: 22.484472

Onyutimaerihus
Koia
Lytusaurus
Ola
Ytroheltorus
Eiadosaurus
Trofiashates
```

Fig. 3.2.15. Hasil generate nama dinosaurus pada iterasi 28000 pada Character-Level Language Modeling

```
Iteration: 30000, Loss: 22.774404

Phytys
Lica
Lysus
Pacalosaurus
Ytrochisaurus
Eiacosaurus
Trochesaurus
```

Fig. 3.2.16. Hasil generate nama dinosaurus pada iterasi 30000 pada Character-Level Language Modeling

```
Iteration: 32000, Loss: 22.209473

Mawusaurus
Jica
Lustoia
Macaisaurus
Yusolenqtesaurus
Eeaeosaurus
Trnanatrax
```

Fig. 3.2.17. Hasil generate nama dinosaurus pada iterasi 32000 pada Character-Level Language Modeling

```
Iteration: 34000, Loss: 22.396744

Mavptokekus
Ilabaisaurus
Itosaurus
Macaesaurus
Yrosaurus
Eiaeosaurus
Trodon
```

Fig. 3.2.18. Hasil generate nama dinosaurus pada iterasi 32000 pada Character-Level Language Modeling

3.3 PasarModal_basic_prediction_LSTM – TATAGLOBAL

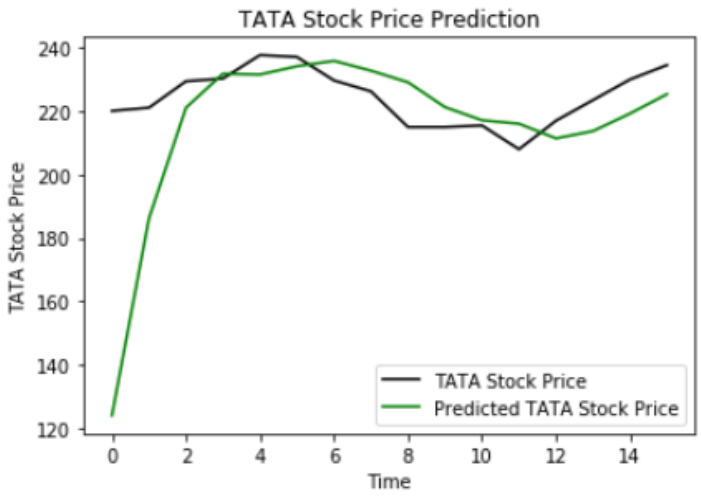

Fig. 3.3. Hasil prediksi harga saham TATA (garis hijau) berbanding harga saham TATA sebenarnya (garis hitam)

3.4 PasarModal_basic_prediction_LSTM – ASUSTEK COMPUTER INC

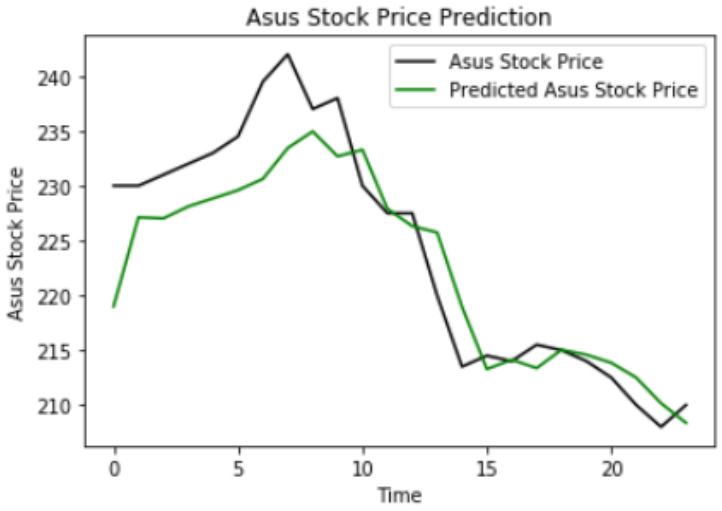

Fig. 3.4. Hasil prediksi harga saham Asus (garis hijau) berbanding harga saham Asus sebenarnya (garis hitam)

4 Simpulan

Dari seluruh rangkaian percobaan yang dilakukan dapat disimpulkan bahwa RNN menggunakan LSTM dapat membuat nama dinosaurus baru dari nama-nama dinosaurus yang telah ada dan dapat memprediksi pergerakan harga saham TATA maupun Asus.

Referensi

1. Olah, Christopher. "Understanding LSTM Networks". Tersedia di: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (diakses pada 06/12/19 pukul 11.04 WIB)
2. Gema, Aryo Pradipta. "Recurrent Neural Network (RNN) dan Gated Recurrent Unit (GRU)". Tersedia di: https://socs.binus.ac.id/2017/02/13/rnn-dan-gru/ (diakses pada 06/12/19 pukul 11.20)