Pattern Recognition:
Recurrent Neural Networks (RNN)

Annisa Nurul Azhar[1]
[1]Institut Teknologi Bandung, Masters Student at Informatics, Bandung, Indonesia
23519025@std.stei.itb.ac.id

Abstrak. *Recurrent Neural Networks* (RNN) adalah salah satu topologi *deep learning* yang banyak digunakan saat ini terutama untuk pemrosesan data-data sekuens dan *time-series*. Terdapat beberapa variasi dari RNN yang juga sering digunakan seperti *Gated Recurrent Unit* (GRU) dan *Long-Short Term Memory* (LSTM). Pada makalah ini akan dibahas mengenai RNN secara umum beserta implementasi dan penerapannya dalam berbagai *task*.

1. Pendahuluan

Recurrent Neural Network (RNN) merupakan suatu kelompok dari *Artificial Neural Network* (ANN) yang dapat memproses input sekuens menggunakan *internal state* yang dimilikinya. Pada *neural network* biasa, keluaran yang dihasilkan hanya dipengaruhi oleh pembobotan inputnya. Sementara itu, pada RNN, keluarannya juga dipengaruhi oleh konteks yang didapatkan dari masukan-masukan atau keluaran-keluaran sebelumnya yang direpresentasikan dengan *hidden state vector* [1].

Akan tetapi, salah satu permasalahan yang muncul ketika menggunakan RNN untuk data sekuens adalah *vanishing gradient problem*. *Vanishing gradient problem* adalah ketika nilai *gradient* menjadi semakin kecil hingga mendekati nol seiring dengan meningkatnya kompleksitas *network* (jumlah *layer* yang bertambah) [2]. Hal ini mengakibatkan *network* menjadi sulit untuk dilatih. *Vanishing gradient problem* dapat terjadi ketika *gap* atau jarak antara informasi yang relevan dengan titik dimana informasi tersebut diperlukan terlalu jauh sehingga RNN tidak mampu untuk menghubungkan keduanya (pembelajaran *network* semakin sulit karena nilai *gradient* semakin mendekati nol seiring dengan propagasi mundur ke *layer-layer* awal) [3]. Oleh karena itu, muncul salah satu

variasi dari RNN yaitu *Long Short-Term Memory* atau LSTM.

LSTM memiliki *control flow* yang hampir mirip dengan RNN. Perbedaannya terletak pada operasi-operasi yang dilakukan pada sel LSTM. LSTM memiliki *cell state* dan juga beberapa *gate* yang tidak terdapat pada RNN. *Cell state* dapat membawa informasi-informasi yang relevan sepanjang pemrosesan sekuens sehingga informasi-informasi pada *timestep-timestep* awal dapat terhubung dengan *timestep-timestep* selanjutnya [4] dan tidak terjadi *vanishing gradient problem*. *Gate* pada LSTM berfungsi untuk menentukan informasi mana saja yang relevan dan perlu disimpan serta informasi yang dapat dilupakan melalui proses pembelajaran. Terdapat tiga jenis *gate* dalam LSTM yaitu *input gate*, *forget gate*, dan *output gate*.

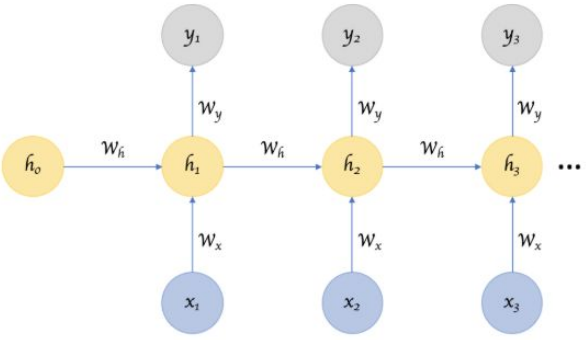


Fig. 1. *Recurrent Neural Network* (RNN)
Sumber: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

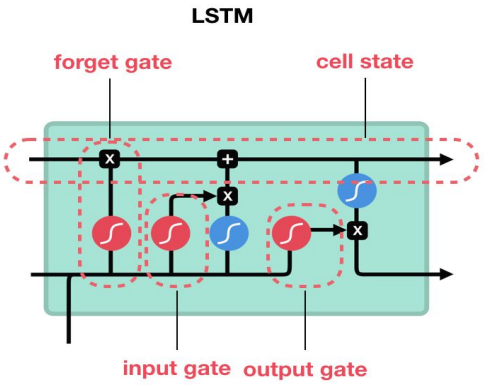


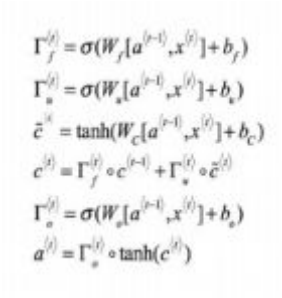
Fig. 2. *Long-Short Term Memory* (LSTM)
Sumber:

https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

2. *Hands-On Recurrent Neural Networks* (RNN) *and Long-Short Term Memory* (LSTM)

Tujuan dari task ini adalah mengimplementasikan proses-proses yang dilakukan di dalam RNN serta LSTM mulai dari menerima sekuens input hingga dapat menghasilkan keluaran. Berikut merupakan daftar fungsi yang diimplementasi beserta penjelasannya.

Fungsi	Penjelasan
rnn_cell_forward	implementasi dari operasi-operasi untuk *single time-step* dari satu sel RNN. Menerima input xt (data input pada *timestep* t), a_prev (*hidden state* pada *timestep* t-1), serta *parameters* yang berisi matriks bobot pengali input (Wxa), *hidden state* (Waa), serta penghubung antara *hidden state* (Wya) ke *output* dan bias input (ba) dan bias *hidden state-output* (by). Keluaran dari fungsi ini adalah a_next (*hidden state* selanjutnya), yt_pred (prediksi pada *timestep* t), dan cache (nilai *tuple* yang diperlukan untuk proses *backward pass*). Berikut operasi yang dilakukan untuk *single time-step* pada satu RNN sel. $a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$ $\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$
rnn_forward	implementasi proses propagasi maju dari RNN. Prosesnya yaitu setiap sel memiliki input yang berupa *hidden state* dari sel sebelumnya sebelumnya a(t-1) dan data input pada *timestep* t. Kemudian dilakukan operasi-operasi seperti pada *rnn_cell_forward* untuk setiap timestep. Keluaran mencakup a (*hidden state* untuk setiap *timestep*), y_pred (prediksi untuk setiap *timestep*), dan cache.
lstm_cell_forward	implementasi *single forward step* pada satu sel LSTM. Menerima input xt, a_prev, c_prev

	(*memory state* pada *timestep* t-1) dan *parameters* yang berisi matrix bobot dan bias dari *forget gate*, *update gate*, *output gate*, *first "tanh"*, serta matrix bobot dan bias penghubung *hidden state* ke *output*. Keluaran mencakup a_next, c_next (*memory state* selanjutnya), yt_pred, dan cache. Berikut operasi-operasi untuk *single forward step* pada satu sel LSTM. 
lstm_forward	implementasi proses propagasi maju pada LSTM. Masukan berupa data input untuk setiap *timestep* dan *initial hidden state*. Kemudian, dilakukan operasi-operasi seperti pada lstm_cell_forward untuk tiap *timestep*. Keluaran yang dihasilkan mencakup a (*hidden state* tiap *timestep*), y (prediksi dari tiap *timestep*), dan caches.
rnn_cell_backward	implementasi proses *backward pass* untuk sebuah sel RNN (*single timestep).* Masukan berupa da_next (*gradient loss* terhadap *next hidden state*) serta cache yang berisi nilai-nilai yang merupakan keluaran dari rnn_cell_forward. Keluaran yang dihasilkan yaitu *gradients* yang mencakup dx (*gradients* dari data input), da_prev (*gradients* dari *hidden state* sebelumnya), dWax (*gradient* dari input ke *hidden weights*), dWaa (*gradient* dari *hidden weights* ke *output*), dan dba (*gradient* dari bias)
rnn_backward	implementasi proses propagasi mundur pada RNN. Prosesnya yaitu menghitung *gradient* pada tiap *timestep* sehingga dapat dipropagasi mundur ke elemen sebelumnya. Untuk itu, proses

	komputasi *gradient* dimulai dari *timestep* paling akhir sampai ke *timestep* pertama, dimana untuk tiap *timestep*, *overall* nilai *gradient* dba, dWaa, dWax di-*increment*. Masukan berupa da (*gradients* untuk semua *hidden states*) dan caches yang berisi nilai dari keluaran rnn_forward. Keluaran yaitu *gradients* yang mencakup dx, da0 (*gradient* dari *hidden state* awal), dWax, dWaa, dan dba.
lstm_cell_backward	implementasi *backward pass* untuk satu sel LSTM (*single timestep*). Masukan yaitu da_next (*gradient* dari *hidden state* selanjutnya), dc_next (*gradient* dari *state* sel selanjutnya), cache (nilai dari keluaran lstm_cell_forward), dan dbo (*gradient* dari bias *output gate*). Keluaran yang dihasilkan adalah *gradients* yang mencakup dxt, da_prev, dc_prev, dWf (*gradient* matriks bobot *forget gate*), dWi (*gradient* matriks bobot *update gate*), dWc (*gradient* matriks bobot *memory gate*), dWo (*gradient* matriks bobot *output gate*), serta *gradient* dari bias pada tiap-tiap *gate* (dbf, dbi, dbc, dbo).

3. *Character Level Language Model*

Tujuan dari *task* ini adalah membangun model bahasa pada level karakter dengan menggunakan RNN. Model bahasa ini akan digunakan untuk menghasilkan nama-nama baru untuk spesies dinosaurus baru berdasarkan nama-nama dinosaurus yang telah ada sebelumnya. Tahap-tahap yang dilakukan antara lain sebagai berikut.

3.1. Praproses

Pada tahap praproses, dilakukan pembacaan dataset (dinos.txt). Selanjutnya, semua karakter diubah menjadi *lowercase* karena kapitalisasi tidak dipertimbangkan. Jumlah seluruh karakter dan jumlah karakter unik pada dataset kemudian dihitung untuk digunakan dalam pembangunan model. Selain itu, dibuat *dictionary* yang memetakan tiap karakter ke sebuah nilai indeks antara 0 sampai 26 (char_to_ix) dan *dictionary* lainnya untuk memetakan kembali tiap nilai indeks

ke karakter yang sesuai (ix_to_char)

3.2 Pembangunan Model

Pada tahap ini, dilakukan implementasi dari struktur model yang diinginkan yaitu melakukan inisialisasi parameter, melakukan *optimisasi* secara berulang hingga konvergen, kemudian menghasilkan keluaran berupa *learned parameters*. Adapun tahap optimisasi yang dilakukan mencakup propagasi maju untuk menghitung *loss function*, propagasi mundur untuk menghitung *gradient* terhadap *loss function*, melakukan *clipping* terhadap *gradient* untuk menghindari *exploding gradient*, kemudian menggunakan *gradient* tersebut untuk melakukan *update* terhadap *parameter* dengan menggunakan *gradient descent update rule*.

Metode *clipping gradient* yang dilakukan pada *task* ini sangat sederhana yaitu dengan menggunakan *simple element-wise clipping procedure* dimana tiap elemen dari *gradient* disesuaikan (dipotong) sehingga berada pada range -10 sampai 10. Oleh karena itu, elemen yang bernilai lebih dari 10 akan diubah nilainya menjadi 10 dan elemen yang bernilai kurang dari -10 akan diubah nilainya menjadi -10. Sementara elemen yang nilainya berada pada *range* tersebut dibiarkan apa adanya.

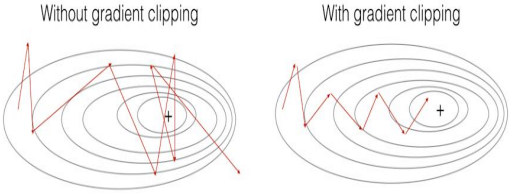


Fig. 3. *Gradient Clipping*
Sumber: *Character-Level Language Modeling - Dinosaurus Land by* Coursera

3.3 Pelatihan Model

Pada tahap ini, diberikan dataset yang berisi nama dinosaururs (dinos.txt) kemudian tiap satu nama dinosaurus pada dataset digunakan sebagai satu *training example*. Selanjutnya untuk tiap 100 *steps* dari *stochastic gradient descent*, dilakukan *sampling* sebanyak 10 nama dinosaurus yang dihasilkan untuk melihat performa dari model.

Dapat dilihat pada Fig. 4 bahwa nama-nama dinosaurus yang di-*generate* semakin *plausible* seiring dengan bertambahnya iterasi yang berarti peningkatan performa model tiap iterasi.

```
Iteration: 0, Loss: 23.087336

Nkzxwtdmfqoeyhsqwasjkjvu
Kneb
Kzxwtdmfqoeyhsqwasjkjvu
Neb
Zxwtdmfqoeyhsqwasjkjvu
Eb
Xwtdmfqoeyhsqwasjkjvu


Iteration: 2000, Loss: 27.884160

Liusskeomnolxeros
Hmdaairus
Hytroligoraurus
Lecalosapaus
Xusicikoraurus
Abalpsamantisaurus
Tpraneronxeros


Iteration: 4000, Loss: 25.901815

Mivrosaurus
Inee
Ivtroplisaurus
Mbaaisaurus
Wusichisaurus
Cabaselachus
Toraperlethosdarenitochusthiamamumamaon


Iteration: 6000, Loss: 24.608779

Onwusceomosaurus
Lieeaerosaurus
Lxussaurus
Oma
Xusteonosaurus
Eeahosaurus
Toreonosaurus
```

Fig. 4. *Generated dinosaurus names* menggunakan *character-level* LM

4. *Stock Price Prediction*

Tujuan dari task ini adalah untuk memprediksi harga saham dengan menggunakan LSTM. Dataset untuk *training* yang digunakan adalah dataset harga saham TATA Global (NSE-TATAGLOBAL.csv). Sementara untuk *testing*, digunakan dataset tatatest.csv. Berikut merupakan tahap-tahap yang dilakukan untuk menghasilkan prediksi harga saham TATA Global.

4.1 Praproses

Pada tahap praproses, seperti biasa dilakukan pembacaan data terlebih dahulu. Terdapat delapan kolom pada data *training* yaitu *Date*, *Open*, *High*, *Low*, *Last*,

Close, *Total Trade Quantity*, dan *Turnover* (*Lacs*). Akan tetapi, pada *task* ini yang akan dipertimbangkan hanya kolom *Open* yaitu harga saham pada saat pembukaan bursa di hari tersebut. Selanjutnya data akan di-*scaling* dengan MinMaxScaler sehingga data harga saham berada pada *range* 0 sampai 1.

Sebelum dilatih dengan LSTM, data harus diubah terlebih dahulu sehingga memenuhi ketentuan sebagai masukan dari LSTM yaitu 3D *array*. Pertama, *generate* data dengan 60 *timesteps* kemudian konversikan menjadi 3D *array* (*num_samples*, 60 *timesteps*, 1 *feature* untuk tiap step).

4.2 Pembangunan Model

Model LSTM dibangun dengan memanfaatkan *library* Keras. Arsitektur dari model LSTM yang dibangun yaitu terdiri dari 4 *layer* LSTM dengan *dropout layers* (*dropout rate*: 0.2) dan 1 *layer* Dense untuk menghasilkan 1 unit keluaran. Tiap-tiap *layer* LSTM memiliki 50 *units*. Untuk *optimizer*, yang digunakan adalah Adam dan untuk *loss function* digunakan *mean squared error*.

4.3 Pelatihan Model dan Evaluasi

Model kemudian dilatih selama 100 epoch dengan *batch size* sebesar 32. Selanjutnya, model dievaluasi terhadap data uji yang telah disiapkan sebelumnya. Gambar Fig. 5 berikut merupakan visualisasi dari performa model terhadap data uji.

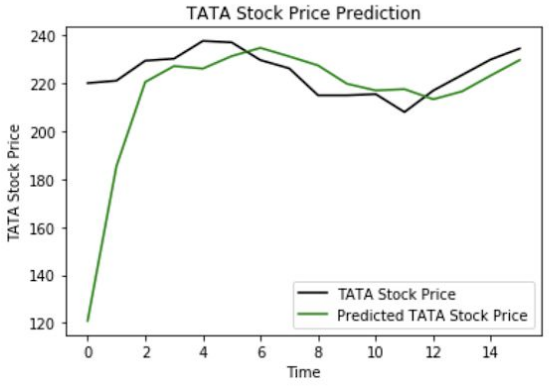


Fig. 5. Visualisai hasil prediksi harga saham

5. *Individual Project*

Untuk *individual project*, topik yang dipilih adalah melakukan prediksi terhadap jumlah pasien di RSUD Kota Bandung pada bulan tertentu. Dataset didapatkan dari Bandung Open Data yaitu data mengenai 10 Kasus Penyakit Tertinggi di Kota Bandung berdasarkan Jenis Kelamin. Tujuan dari *project* ini adalah untuk membantu pihak rumah sakit dalam menyiapkan stok alat-alat medis, obat-obatan, serta fasilitas kesehatan lainnya. Berikut merupakan tahap-tahap yang dilakukan pada *individual project*.

5.1 *Data Gathering* dan *Data Preparation*

Pada tahap ini dilakukan pengumpulan data dari Bandung Open Data. Dalam *website* tersebut data-datanya masih terpisah berdasarkan bulan dan tahunnya oleh karena itu perlu digabungkan terlebih dahulu. Pada data asli dari Bandung Open Data, terdapat beberapa atribut seperti tahun, bulan, nama penyakit, nama klinik, kode kamar, serta jumlah kasus pada pasien perempuan dan laki-laki. Tetapi karena fokus pada *project* ini adalah jumlah pasien, atribut yang digunakan hanya tahun, bulan, dan jumlah kasus gabungan. Data kemudian diurutkan dari yang paling lama (Januari 2016) hingga yang terbaru (Juni 2019). Data yang telah diproses dapat dilihat pada *file* patients.csv. Untuk pembagian data latih dan data uji, 20% data terbaru digunakan sebagai data uji dan sisanya sebagai data latih. Setelah itu, sama seperti pada *task Stock Price Prediction*, digunakan MinMaxScaler untuk melakukan *scaling* terhadap data dan mengubah data menjadi 3D *array* (*num_samples*, 1 *timestep*, 1 *feature* untuk tiap *step*).

5.2 Pembangunan Model

Untuk *individual project*, model yang dibangun hanya akan terdiri dari satu *layer* LSTM dengan 256 unit dan satu *layer* Dense sebagai *output layer*. Karena jumlah data yang masih sedikit, model dibuat sesederhana mungkin untuk menghindari *overfitting*. *Optimizer* yang digunakan adalah Adam dan *loss function* yang digunakan yaitu *mean squared error*.

5.3 Pelatihan Model dan Evaluasi

Model dilatih selama 100 epoch dengan *batch size* sebesar 1. Model kemudian dievaluasi terhadap 20% data terbaru yang sebelumnya telah diambil untuk digunakan sebagai data uji. Fig. 6 berikut merupakan gambaran performa model terhadap data uji. Seperti dapat dilihat, prediksi yang diberikan masih cukup jauh

dibandingkan dengan jumlah sebenarnya. Hal ini dapat disebabkan karena data yang tersedia belum cukup banyak. Dengan penambahan data tentunya model dapat belajar lebih banyak sehingga prediksi yang dihasilkan lebih akurat.

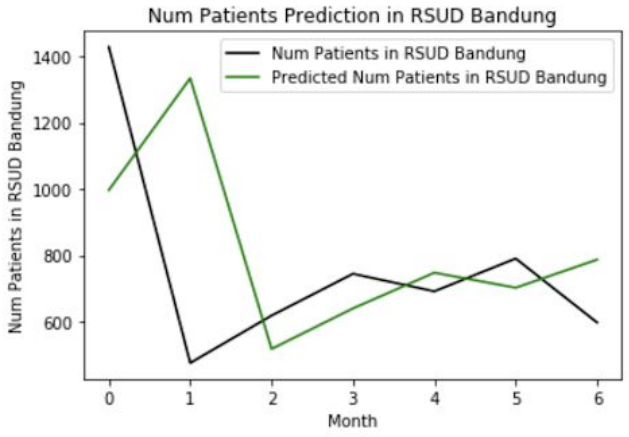


Fig. 5. Visualisai hasil prediksi jumlah pasien di RSUD Kota Bandung

6. Kesimpulan

Penggunaan RNN maupun LSTM untuk pemrosesan data sekuens merupakan hal yang tepat karena baik RNN maupun LSTM memiliki *chain-like architecture* yang memang sesuai dengan data sekuens. Seperti telah dibahas di atas, RNN dan LSTM dapat diterapkan untuk berbagai *task* pada domain yang berbeda-beda seperti pembangunan *language model*, prediksi harga saham, hingga prediksi jumlah pasien di rumah sakit.

Referensi

1. Venkatachalam, M. *Recurrent Neural Networks*. URL: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce
2. Wang, C. *The Vanishing Gradient Problem*. URL: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484
3. Colah. *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
4. Nguyen, M. *Illustrated Guide to LSTM's and GRU's: A step by step explanantion*. URL: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
5. *Hands-On Recurrent Neural Network*. Coursera.
6. *Character-Level Language Modeling - Dinosaurus Land*. Coursera.
7. TATA Global Stock Market Price
8. Bandung Open Data (data.bandung.go.id)