Universitas Pertamina

SEQUENCE MODEL

DEEP LEARNING

MEREDITA SUSANTY

Penerbit Universitas Pertamina

Sequence Model Deep Learning

Penulis: Meredita Susanty

ISBN: 978-623-94030-7-2

Editor: Prasasya Kirana

Penerbit: Universitas Pertamina

Cetakan Pertama, 2021
Edisi Pertama, 2021

Jl Teuku Nyak Arief Simprug
Kebayoran Lama, Jakarta Selatan 12120
Telepon : 021-29044308
Website : https://universitaspertamina.ac.id/
Email : info@universitaspertamina.ac.id

Cetakan Pertama, 2021
Edisi Pertama, 2021

Susanty, Meredita

Sequence Model Deep Learning

—Jakarta: Penerbit Universitas Pertamina,2021
1 jil., 155 hlm., 17,6 x 25 cm

ISBN. 978-623-94030-7-2

1. Ilmu Komputer
2. Kecerdasan Buatan
I. Judul
II. Susanty, Meredita

Tentang Penulis

Meredita Susanty adalah dosen tetap program studi Ilmu Komputer Universitas Pertamina sejak tahun 2016. Meredita meraih gelar sarjana komputer dari Universitas Gadjah Mada dan gelar master dari University of Nottingham. Di program studi Ilmu Komputer Meredita mengajar Pengantar Teknologi Informasi dan Algoritma, Dasar Pemrograman dan Rekayasa Perangkat Lunak.

Prakata

Jika seratus tahun lalu listrik mengubah setiap industri besar mulai dari transportasi, manufaktur, kesehatan, komunikasi, dan banyak bidang lainnya. Saat ini perkembangan *deep learning* mengakibatkan transformasi di berbagai bidang seperti dalam melakukan pencarian melalui situs web, melakukan pemasaran daring, melakukan pembacaan X-ray di bidang kesehatan, melakukan personalisasi pendidikan, hingga otomotif dengan munculnya kendaraan tanpa awak.

Karena pengaruhnya di berbagai sektor, kemampuan dan penguasaan disiplin ilmu ini sangat dicari dalam bidang teknologi dan informasi. Buku ini diharapkan dapat membantu para pemula untuk memahami cara kerja salah satu model dalam deep learning dimana urutan data perlu diperhatikan. Karena urutan data berpengaruh penting, model sekuens akan sedikit lebih kompleks dibandingkan dengan model jaringan saraf tiruan atau *convolutional neural network* yang banyak digunakan dalan pengenalan citra. Dengan ilustrasi arsitektur model dan tahapan perhitungan yang dilakukan dalam model diharapkan pembaca dapat lebih mudah memahami cara kerja setiap model yang dibahas dalam buku ini.

Buku ini ditujukan untuk pembaca yang kurang familiar dengan Kalkulus sehingga dasar teori perhitungan tidak dibahas mendalam. Bagi pembaca yang memiliki pemahaman yang baik tentang kalkulus atau ingin mendalami teori dasar kalkulus yang mendasari perhitungan pada tiap model, dapat mengacu ke referensi yang disediakan di bagian akhir setiap bab.

Buku ini diharapkan dapat membantu pembaca dari berbagai kalangan untuk menambah pengetahuan dan membentuk pemahaman mengenai model sekuens yang menggunakan arsitektur *recurrent neural network* dan variannya seperti *gated recurrent unit* dan *long short-term memory*.

Daftar Isi

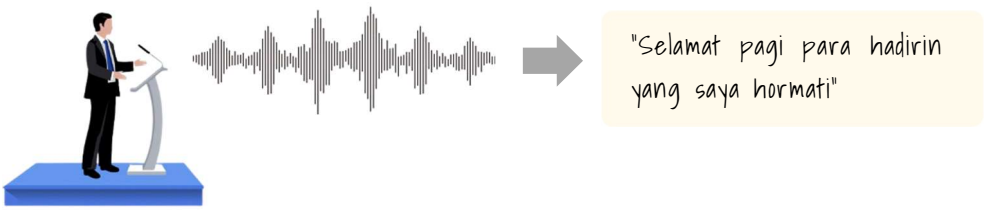
BAB 1

Sequence Model

Tujuan:

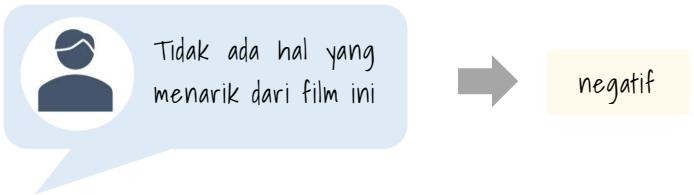
- Memahami berbagai masalah yang dapat diselesaikan menggunakan *sequence model*.
- Mengetahui berbagai notasi yang digunakan untuk menjelaskan training set untuk data sekuensial.
- Memahami cara merepresentasikan kata dalam kalimat pada *sequence model*.

Kita akan mempelajari *sequence model*, salah satu area yang paling menarik dalam *deep learning*. Salah satu model yang akan dipelajari adalah *Recurrent Neural Network* (RNN) yang banyak digunakan dalam pengenalan suara, pemrosesan bahasa alami, dan banyak area lainnya. Sebelum membahas detail mengenai RNN, kita akan melihat beberapa contoh penggunaan *sequence model*.

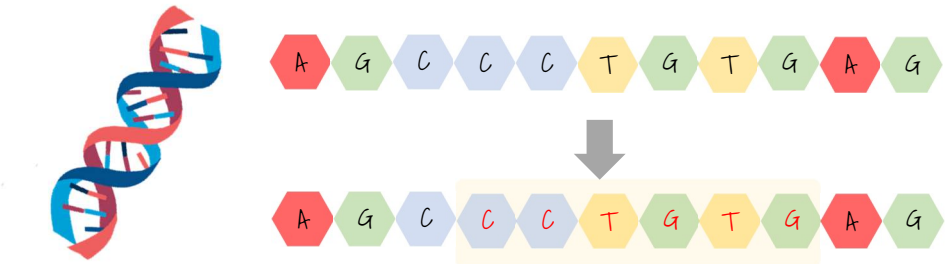


Di area pengenalan suara, kita diberikan input berupa audio kemudian memetakannya menjadi transkripsi dalam bentuk teks sebagai output. Pada contoh ini, baik input maupun output adalah sekuens data dimana input berupa klip audio saat dijalankan akan mengeluarkan suara per satuan waktu dan output adalah urutan kata.

Contoh lainnya adalah dalam membuat musik. Input bisa saja berupa himpunan kosong atau sebuah integer yang menyatakan aliran musik yang ingin dihasilkan atau beberapa not awal yang diinginginkan. Dari input tersebut, outputnya berupa sekuens. Pada kasus ini, hanya output yang berupa sekuens.



Penggunaan lain *sequence model* adalah untuk melakukan klasifikasi sentimen dengan input berupa kalimat seperti "Tidak ada hal yang menarik dari film ini". Dari kalimat tersebut, dilakukan pengelompokan apakah masuk sebagai sentimen yang positif/baik atau negatif/buruk. Berkebalikan dengan contoh sebelumnya, pembuatan musik, hanya input yang berupa sekuens pada contoh klasifikasi sentimen.



Sequence model juga sangat berguna di bidang bioinformatika untuk melakukan analisis sekuens DNA. DNA biasanya direpresentasikan dengan huruf A, C, G, dan T. Dari untaian atau sekuens DNA diberi label bagian mana saja yang mungkin berinteraksi dengan protein.

Harry Potter and Hermione Granger invented a new spell.

Harry Potter dan Hermione Granger menemukan sebuah mantra sihir baru.

Pada mesin menerjemah, *sequence model* menerima input berupa kalimat dalam suatu bahasa dan memberikan output kalimat dalam bahasa lain. Misalnya input adalah kalimat bahasa Inggris dan outputnya adalah kalimat dalam bahasa Indonesia. Disini, baik input maupun output berbentuk sekuens

Bermain golf

Dalam *video activity recognition,* atau pengenalan kegiatan dalam video, kita mungkin diberikan urutan gambar yang membentuk suatu video dan diminta untuk mengenali aktivitas yang ada dalam rangkaian frame video tersebut.

Harry Potter dan Hermione Granger menemukan mantra sihir baru.

Harry Potter dan Hermione Granger menemukan mantra sihir baru.

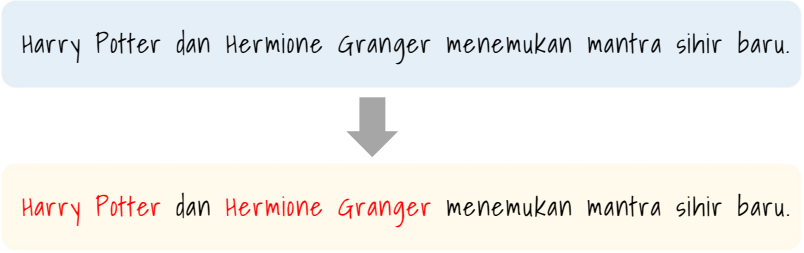
Dalam bidang pemrosesan bahasa alami ada ***name-entity recognition*** **(NER)** atau pengenalan entitas-entitas yang ada dalam sebuah kalimat. Input berupa kalimat dan outputnya adalah kata-kata yang merupakan entitas tertentu misalnya orang, lokasi, organisasi.

Semua contoh yang diberikan, dapat diselesaikan menggunakan pendekatan *supervised learning* dengan label data X untuk input dan Y untuk output pada *training set*. Namun, pada contoh diatas dapat dilihat bahwa masing-masing permasalahan memiliki bentuk sekuens yang berbeda. Pada beberapa contoh, baik input X dan output Y berupa sekuens. Dalam kasus sedemikian panjang sekuens X dan Y mungkin saja berbeda namun bisa juga sama. Pada beberapa contoh lainnya, hanya input X atau hanya output Y yang berupa sekuens. Dari beragam contoh diatas, kita akan mempelajari bagaimana *sequence model* dapat digunakan pada berbagai situasi.

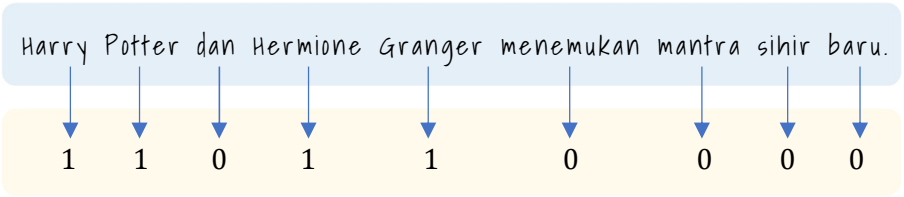
Notasi

Kita sudah melihat berbagai aplikasi dimana *sequence model* dapat digunakan. Sekarang kita akan mendefinisikan notasi yang akan digunakan di buku ini untuk membangun *sequence model*.

Sebagai ilustrasi, misal kita ingin membuat sebuah *sequence model* yang menerima input sebagai berikut; ”Harry Potter dan Hermione Granger menemukan mantra sihir baru”. Yang diharapkan adalah *sequence model* dapat secara otomatis menunjukkan dimana letak nama orang pada kalimat tersebut. Permasalahan ini dikenal dengan ***name-entity recognition*** **(NER)**. NER digunakan oleh mesin pencarian (*search engine*) untuk mengindekskan semua artikel berita yang muncul dalam 24 jam terakhir berdasarkan nama orang yang disebutkan di dalam artikel berita tersebut. Nama pada sistem ini bisa berupa nama orang, nama perusahaan atau organisasi, waktu, nama tempat atau lokasi, nama negara, nama mata uang, dan jenis nama lainnya pada berbagai jenis teks.

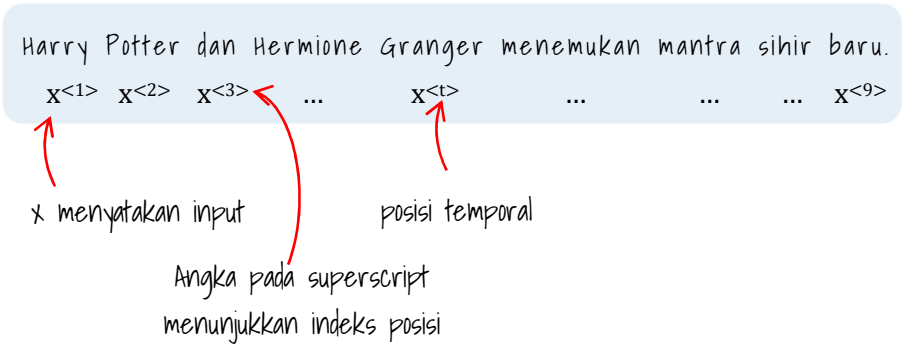


Dengan input X, kita menginginkan sebuah model yang memiliki satu output untuk setiap kata pada input dan target output Y memberi tahu apakah setiap kata merupakan nama orang atau bukan.

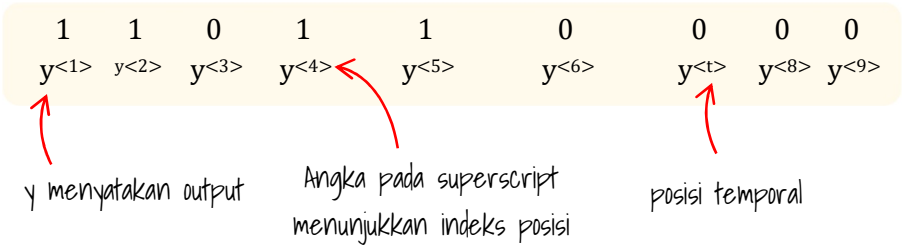


Secara teknikal solusi sebelumnya bukanlah representasi output terbaik. Ada representasi output yang lebih canggih. Representasi ini dapat memberitahukan bukan hanya kata mana saja yang merupakan nama orang tapi juga dimana nama orang tersebut dimulai dan berakhir pada kalimat yang diberikan. Misalnya pada contoh diatas, Harry potter berada pada posisi satu hingga dua.

Agar lebih mudah, pada bagian ini kita hanya akan menggunakan representasi output yang lebih sederhana. Pada contoh diatas, input adalah kalimat yang terdiri dari sembilan kata. Dengan demikian kita akan memiliki sembilan fitur untuk merepresentasikan kesembilan kata ini, indeks untuk menunjukkan posisi pada sekuens. Kita akan menggunakan notasi x dan *superscript* berupa kurung siku dengan indeks 1, 2, 3 dan seterusnya di antara pasangan kurung siku yang menunjukkan posisi. Untuk menunjukkan posisi sementara pada sekuens kita bisa menggunakan notasi t sebagai indeks.

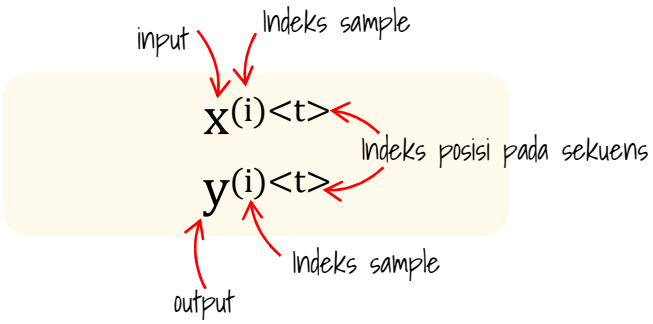


Notasi serupa juga digunakan untuk output. Kita akan menggunakan notasi y dan *superscript* berupa kurung siku dengan indeks 1, 2, 3 dan seterusnya di antara pasangan kurung siku yang menunjukkan posisi.



Untuk menunjukkan panjang sekuens, kita akan menggunakan T dengan subscript x atau y sebagai sekuensnya. Pada contoh diatas panjang kata yang menjadi input adalah sembilan sehingga Tx = 9. Outputnya juga memiliki panjang sembilan sehingga Ty = 9. Pada kasus ini panjang input dan output sama (Tx = Ty) . Contoh lain pada saat kita menerjemahkan "*Harry Potter and Hermione Granger invented a new spell*" menjadi "Harry Potter dan Hermione Granger menemukan sebuah mantra sihir baru", Tx tidak sama dengan Ty

Umumnya kita akan menggunakan lebih dari satu sekuens input dalam training set. Untuk itu, gunakan notasi tanda kurung dengan indeks 1,2,3 dan seterusnya diantara pasangan tanda kurung untuk menunjukkan sampel pada indeks tersebut. Notasi i digunakan untuk menunjukkan indeks sample tertentu dalam training set.



Pada training set ada beberapa pasangan sample input dan output sebagai berikut:

(<"Harry Potter memenangkan pertandingan Quidditch", 1 1 0 0 0>,

<"Harry Potter bertemu Hermione Granger kemarin sore", 1 1 0 1 1 0 0 >, <"Ron Weasley tinggal di asrama yang sama dengan Harry Potter", 1 1 0 0 0 0 0 1 1 >

Tuliskan apa yang diacu oleh notasi dibawah ini:

$x^{(2)}$ .....

$Tx^{(2)}$ .....

$Tx^{(3)}$ .....

$x^{(2)<3>}$ .....

$y^{(3)<2>}$ .....

$Ty^{(1)}$ .....

$y^{(2)}$ .....

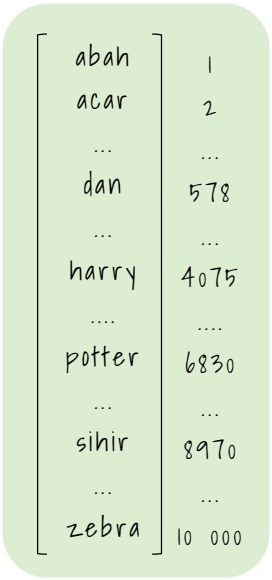
Representasi Kata

Dalam buku ini kita akan banyak menggunakan contoh dari bidang pemrosesan bahasa alami untuk menunjukkan cara kerja berbagai model sekuens. Karena itu kita perlu tahu bagaimana merepresentasikan sebuah kata dalam sekuens. Misalnya kata Harry atau $x^{<1>}$ pada contoh sebelumnya.

X : Harry Potter dan Hermione Granger menemukan mantra sihir baru.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<9>}$

Untuk merepresentasikan sebuah kata dalam kalimat, yang pertama kali dilakuan adalah membuat kosakata (*vocabulary*). Ada juga yang menyebutnya kamus (*dictionary*). Kosakata atau kamus adalah daftar kata-kata yang akan digunakan dalam representasi. Misalkan kita memiliki sebuah kamus yang berisi kata seperti dibawah ini:

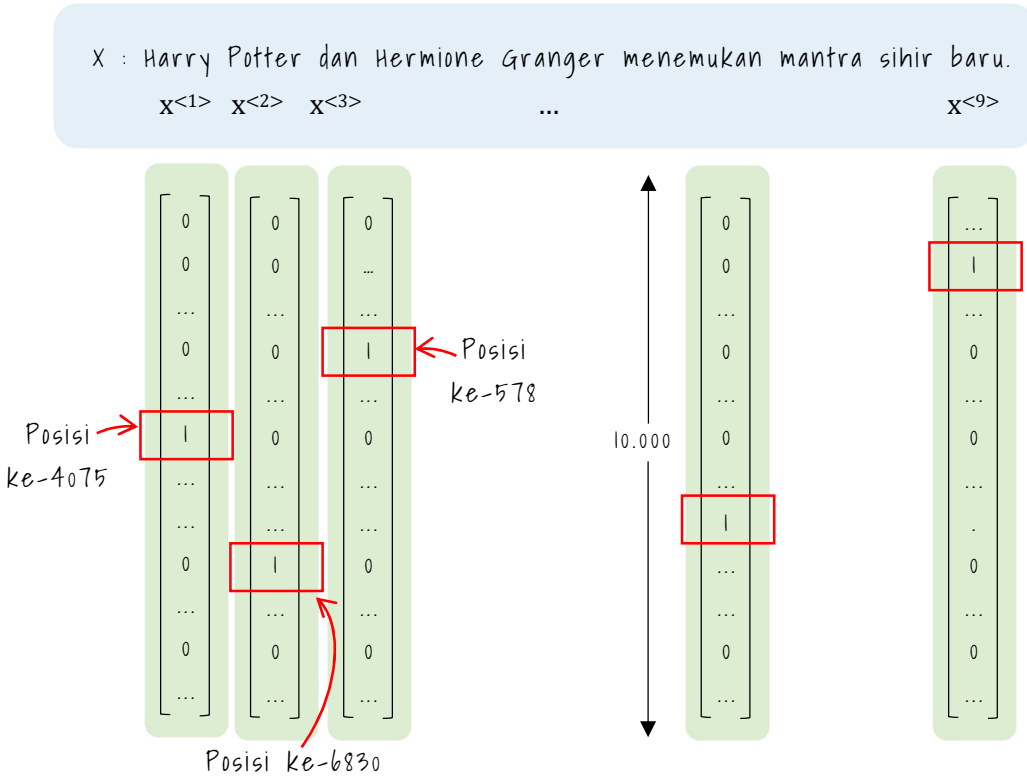


Abah adalah kata pertama, acar kata kedua, dan kata ke-578, harry kata ke-4075 hingga zebra adalah kata ke-10.000 dalam kamus

ukuran kamus adalah 10.000

Pada contoh ini kita menggunakan ukuran kamus yang relatif kecil. ukuran kamus yang biasa digunakan pada aplikasi komersil bisa mencapai jutaan kata. Umumnya ukuran yang digunakan antara 30.000 hingga 50.000

Setelah memiliki kamus, selanjutnya kita akan menggunakan representasi *one-hot* untuk merepresentasikan setiap kata. Misalnya, $x^{<1>}$ yang merepresentasikan kata Harry akan berupa sebuah vektor dengan semua nilai adalah 0 kecuali pada posisi 4075 yang akan bernilai 1. Kemudian $x^{<2>}$ yang merepresentasikan kata Potter akan berupa sebuah vektor dengan semua nilai adalah 0 kecuali pada posisi 6830 yang akan bernilai 1. Semua kata akan berupa vektor 10.000 dimensi jika kamus berisi 10.000 kata. Maka dalam representasi ini, $x^{<t>}$ untuk setiap nilai t dalam kalimat akan berupa *one-hot vector*. Disebut *one-hot* karena hanya ada tepat satu nilai 1 dan sisanya adalah 0 untuk setiap kata dalam kalimat. Pada contoh diatas, kita akan memiliki sembilan one-hot vector yang merepresentasikan sembilan kata dalam kalimat.



Tujuan yang ingin dicapai adalah dengan representasi x ini kita mampu mempelajari pemetaan menggunakan *sequence model* untuk menghasilkan target output y.

Pada bab berikutnya kita juga akan membahas lebih detail bagaimana jika kita menemukan kata yang tidak ada dalam kamus. Pada kasus demikian, kita akan membuat sebuah token baru atau sebuah kata palsu yang disebut *unknown word* menggunakan notasi UNK.

BAB 3

Gated Recurrent Unit

Tujuan:

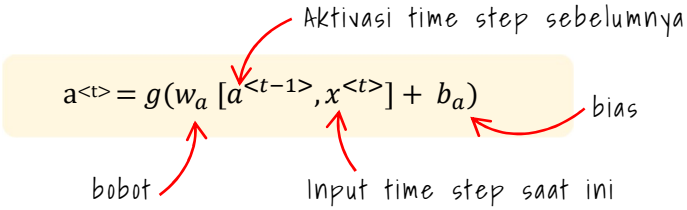
- Paham perbedaan *recurrent neural network* dan *gated recurrent unit*
- Paham cara kerja *gated recurrent unit* sederhana
- Paham bagaimana *gated recurrent unit* mengatasi keterbatasan pada *recurrent neural network*

Pada bab ini kita akan membahas *gated recurrent unit* (GRU) yang merupakan modifikasi terhadap *hidden layer* pada arsitektur RNN. Modifikasi ini membuat GRU lebih baik dalam menangkap hubungan antara input yang letaknya berjauhan (*long-range connection*). Selain itu, GRU juga mengatasi permasalahan *vanishing gradient* yang terjadi pada arsitektur RNN.

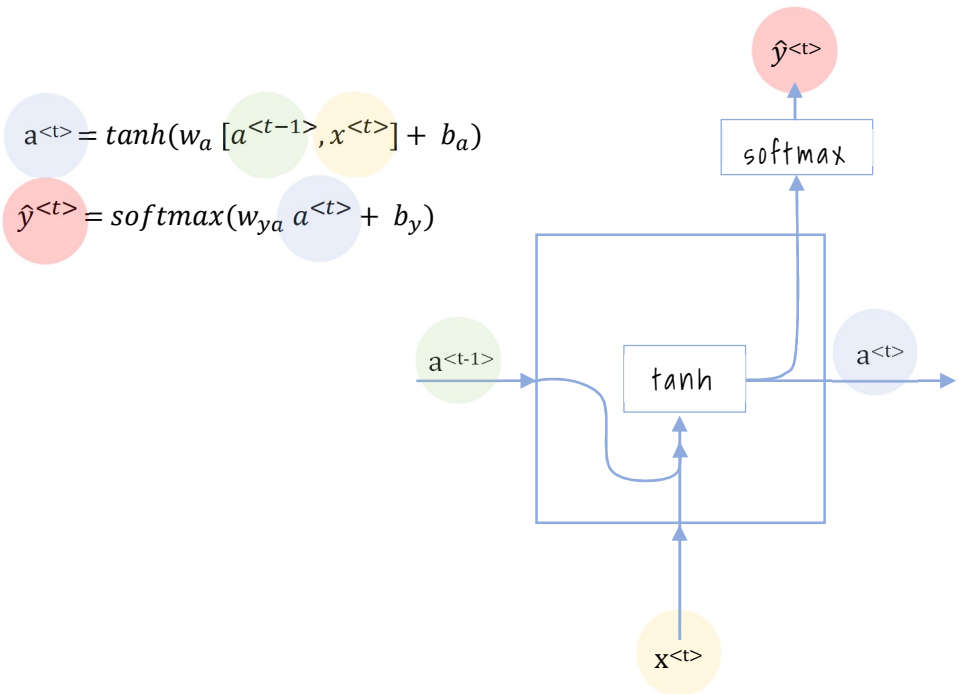
Pada sub-bab berikutnya kita akan membahas perbedaan mendasar antara arsitektur GRU dan RNN. Kemudian membahas cara kerja GRU yang disederhanakan. Setelah memahami cara kerja GRU yang disederhanakan, kita akan membahas cara kerja GRU secara keseluruhan.

5.1 Perbedaan GRU dan RNN

Pada bab sebelumnya kita sudah membahas cara menghitung fungsi aktivasi pada *time step* tertentu pada arsitektur RNN. Fungsi aplikasi dihitung terhadap hasil perhitungan parameter bobot w_a dikalikan aktivasi *time step* sebelumnya dan input pada *time step* saat ini ditambah parameter bias.



Jika digambarkan, perhitungan ini akan menjadi seperti ilustrasi dibawah ini. RNN unit digambarkan sebagai kotak yang menerima input berupa nilai aktivasi dari *time step* sebelumnya ($a^{<t-1>}$) dan input untuk *time step* saat ini ($x^{<t>}$). Keduanya digunakan dalam perhitungan dengan parameter bobot dan bias. Setelah perhitungan linear ini, jika g adalah fungsi aktivasi tanh, maka setelah menghitung nilai tanh terhadap hasil perhitungan linear didapatkan nilai aktivasi untuk *time step* saat ini ($a^{<t>}$). Output ini ($a^{<t>}$) juga mungkin digunakan untuk softmax unit atau unit lain yang digunakan untuk menghasilkan output $\hat{y}^{<t>}$. Ilustrasi ini hanya untuk sebuah RNN unit pada *hidden layer* dari arsitektur RNN.



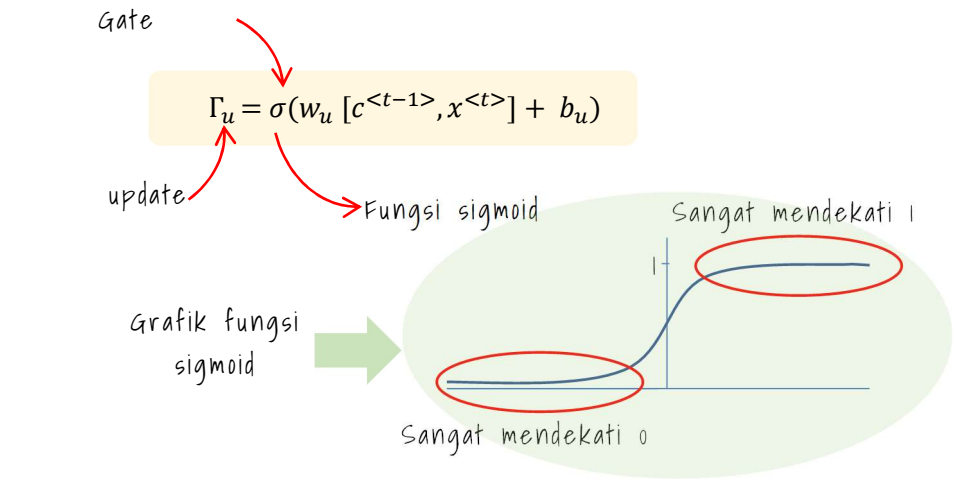
Kita akan menggunakan gambar yang serupa untuk mengilustrasikan GRU unit. Untuk menjelaskan GRU,kita akan menggunakan kalimat dalam Bahasa Inggris yang kita gunakan pada bab sebelumnya. Kalimat pertama adalah " the cat, which climb the tree, was stuck there for three hours" dan kalimat kedua "the cats, which climb the tree, were stuck there for three hours". Jika subjeknya tunggal maka menggunakan kata kerja bentuk tunggal (cat – was) dan jika subjeknya jamak maka menggunakan kata kerja bentuk jamak (cat – were). GRU unit akan memiliki sebuah variabel tambahan yakni variabel C. C merupakan singkatan dari *cell* yang mewakili *memory cell*. *Memory cell* mencatat sedikit informasi (ingatan). Pada contoh kalimat diatas, mengingat apakah kata *cat* adalah tunggal atau jamak sehingga pada bagian kalimat yang lebih lanjut dapat menyesuaikan kata kerja dengan bentuk subjeknya.

Pada time step t, memory cell akan menyimpan nilai aktivasi pada time step t. Pada setiap time step, kita akan mengubah nilai $c^{<t>}$ dengan nilai $\tilde{c}^{<t>}$ (dibaca c tilda). $\tilde{c}^{<t>}$ adalah kandidat pengganti nilai $c^{<t>}$. Dihitung menggunakan fungsi aktivasi tanh terhadap hasil perhitungan parameter bobot w_c dikalikan *memory cell time step* sebelumnya dan input pada *time step* saat ini ditambah parameter bias.

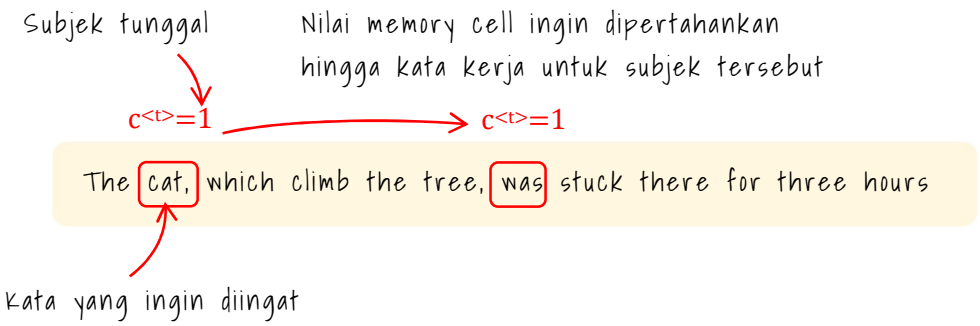
$$c^{<t>} = a^{<t>}$$

$$\tilde{c}^{<t>} = tanh(w_c\,[c^{<t-1>}, x^{<t>}] + b_c)$$

Ide utama dari GRU adalah kita akan memiliki suatu gerbang yang dinotasikan dengan Gamma (Γ). Γ_u adalah *update gate* akan menyimpan nilai antara 0 hingga 1 yang didapatkan dari perhitungan fungsi sigmoid terhadap hasil perhitungan parameter bobot w_u dikalikan *memory cell time step* sebelumnya dan input pada *time step* saat ini ditambah parameter bias. Untuk memudahkan pemahaman mengenai cara kerja GRU, kita akan menganggap nilai selalu 0 atau 1 walaupun pada kenyataannya bisa berupa nilai lain diantara 0 dan 1.

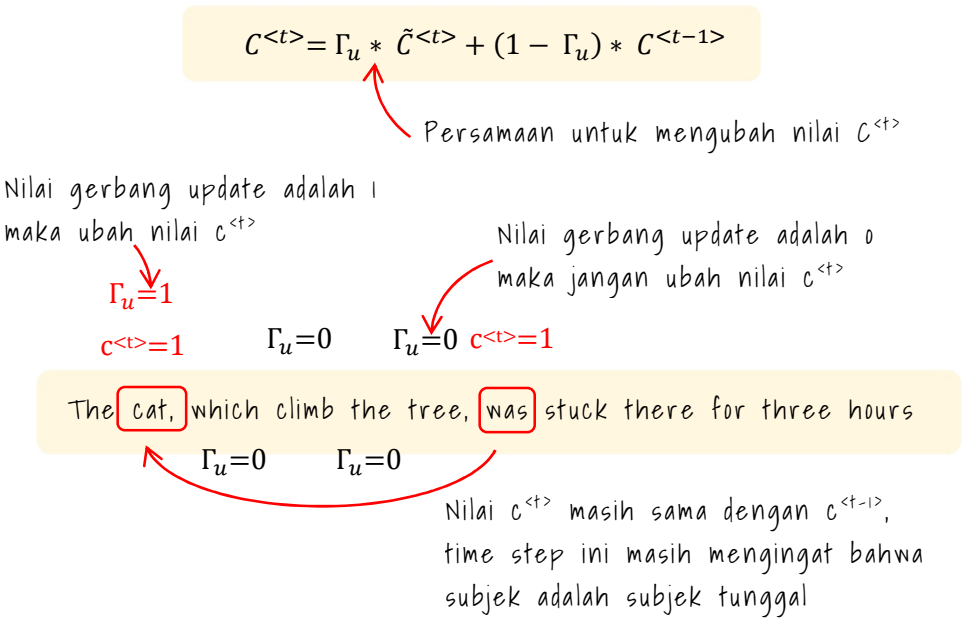


Bagian penting lain dari GRU adalah persamaan untuk memutuskan apakah kita akan mengubah nilai $c^{<t>}$ menjadi $\tilde{c}^{<t>}$. Kita akan menggunakan contoh kalimat "the cat, which climb the tree, was stuck there for three hours" untuk mengilustrasikan bagaimana keputusan ini diambil. Misal memory cell $c^{<t>}$ kita beri nilai 0 atau 1 untuk menyatakan apakah kata yang ingin diingat atau subjek dari suatu kalimat merupakan subjek tunggal atau jamak. Dalam contoh ini, jika subjek tunggal kita akan membuat nilai $c^{<t>}$ bernilai 1, jika subjek jamak nilai $c^{<t>}$ adalah 1. GRU unit akan mengingat nilai $c^{<t>}$ hingga menemukan kata kerja untuk subjek tersebut dimana nilai $c^{<t>}$ tetap 1. Karena nilai $c^{<t>}$ adalah 1, hal ini memberikan informasi bahwa subjek adalah subjek tunggal sehingga kata kerja yang dipilih juga kata kerja dalam bentuk tunggal.

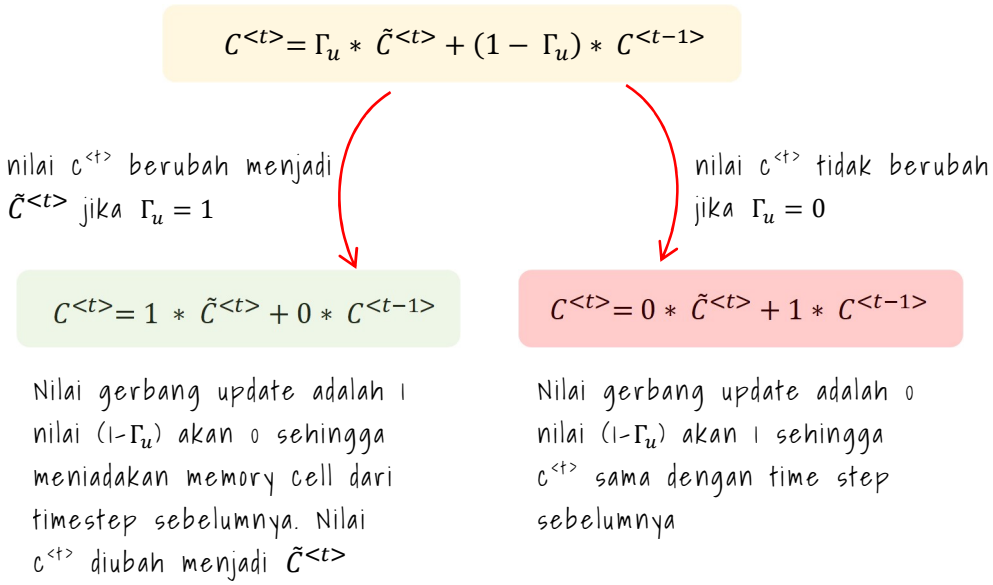


Dalam proses tersebut, tugas gerbang Γ_u adalah memutuskan kapan kita akan mengubah nilai $c^{<t>}$. Pada saat kita melihat frasa "the cat", kita tahu bahwa ini adalah sebuah konsep baru dan "cat" merupakan subjek dari

kalimat. Maka, kita ingin mengubah nilai $c^{<t>}$. Kemudian, setelah selesai membentuk kalimat "the cat was full" kita tidak perlu mengingat bentuk subjek itu lagi. Persamaan yang akan kita gunakan untuk GRU adalah $C^{<t>} = \Gamma_u * \tilde{C}^{<t>} + (1 - \Gamma_u) * C^{<t-1>}$. Jika gerbang update bernilai 1, maka ini menyatakan ubah nilai $c^{<t>}$ menjadi nilai kandidat $\tilde{c}^{<t>}$. Untuk kata-kata yang berada di tengah "which climb the tree", kita harus membuat Γ_u bernilai 0 sehingga tidak mengubah nilai $C^{<t>}$. Dengan kata lain, $C^{<t>}$ tetap menyimpan nilai yang sebelumnya.



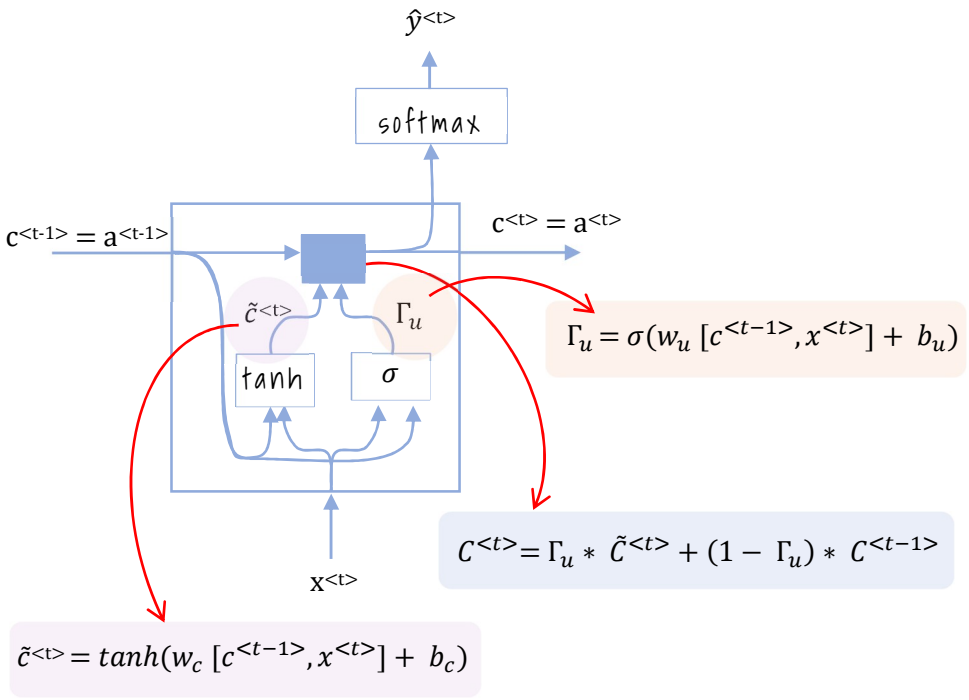
Pada persamaan $C^{<t>} = \Gamma_u * \tilde{C}^{<t>} + (1 - \Gamma_u) * C^{<t-1>}$, jika nilai Γ_u adalah 0, maka $(1 - \Gamma_u)$ bernilai 1. Nilai kandidat akan menjadi 0 dan kita akan menggunakan nilai *memory cell* pada time step sebelumnya sebagai nilai *memory cell* untuk time step saat ini. Saat kita membaca kata dari kiri ke kanan, saat gerbang update bernilai 0 ini memberitahukan kita untuk tetap menggunakan nilai memory cell yang lama atau menyuruh kita untuk mengingat nilai yang sebelumnya. Dengan demikian bahkan hingga kita menemukan kata "was" nilai $C^{<t>}$ masih sama dengan nilai $C^{<t-1>}$ sehingga kita masih ingat bahwa subjek kalimat adalah subjek tunggal.



Jika digambarkan, perhitungan ini akan menjadi seperti ilustrasi dibawah ini. GRU unit menerima input berupa nilai *memory cell* dari *time step* sebelumnya ($c^{<t-1>}$) dimana nilai *memory cell* sama dengan nilai aktivasi dari *time step* sebelumnya ($a^{<t-1>}$) dan input untuk *time step* saat ini ($x^{<t>}$). Kedua input ini kemudian digabungkan menggunakan fungsi aktivasi tanh dan menghasilkan nilai $\tilde{c}^{<t>}$ yang merupakan kandidat untuk menggantikan nilai $c^{<t>}$. Menggunakan parameter yang berbeda, kedua input ini digunakan untuk menghitung nilai gerbang update menggunakan fungsi sigmoid. Nilai kandidate dan gerbang update kemudian digabungkan menggunakan operasi $C^{<t>} = \Gamma_u * \tilde{C}^{<t>} + (1 - \Gamma_u) * C^{<t-1>}$. Operasi ini menggunakan gerbang update Γ_u, kandidat nilai $\tilde{c}^{<t>}$dan memory cell time step sebelumnya $c^{<t-1>}$ sebagai input dan menghasilkan nilai untuk *memory cell* pada time step saat ini. Jika diperlukan, output dari persamaan ini juga digunakan untuk melakukan prediksi nilai output $\hat{y}^{<t>}$ menggunakan fungsi softmax atau fungsi lainnya.

Sejauh ini kita sudah membahas GRU unit dan perbedaannya dengan RNN unit. Perlu diingat, yang kita bahas sejauh ini adalah bentuk sederhana dari GRU. Hal yang menarik dari GRU adalah saat kita membaca kata dari kiri ke kanan, gerbang update memutuskan apakah ini saat yang tepat untuk mengubah nilai *memory cell* menjadi 1, tidak

mengubah nilai *memory cell* hingga pada suatu kata dimana kita perlu menggunakan *memory cell* yang sudah diatur diawal kalimat tadi.



Karena nilai gerbang update sangat mudah dibuat menjadi 0 asalkan hasil perhitungan $w_u\ [c^{<t-1>}, x^{<t>}] + b_u$ bernilai negatif yang cukup besar. Dengan pembulatan numerik, maka nilai gerbang update akan menjadi 0 atau sangat dekat dengan 0. Jika ini terjadi, maka akan $C^{<t>}$ pada persamaan $C^{<t>} = \Gamma_u * \tilde{C}^{<t>} + (1 - \Gamma_u) * C^{<t-1>}$ akan bernilai $C^{<t-1>}$. Hal ini sangat berguna untuk mengingat nilai untuk suatu *cell*. Karena gamma bisa bernilai sangat mendekati 0, misalnya 0,000001 atau lebih kecil lagi GRU tidak mengalami permasalahan *vanishing gradient* seperti RNN. Karena jika gamma sangat mendekati 0, pada dasarnya nilai $C^{<t>}$ sama dengan $C^{<t-1>}$. Selain itu, nilai $C^{<t>}$ tidak banyak berubah bahkan di banyak kata. Hal ini sangat membantu dalam hal *vanishing gradient* dan memungkinkan *network* mempelajari ketergantungan antar kata yang bahkan terletak sangat jauh (*long-range dependencies*). Pada contoh diatas, kata ”the cat” dan ”was” berkaitan, walaupun dipisahkan banyak kata diantaranya ”which climb the tree”.