BAB II

LANDASAN TEORI

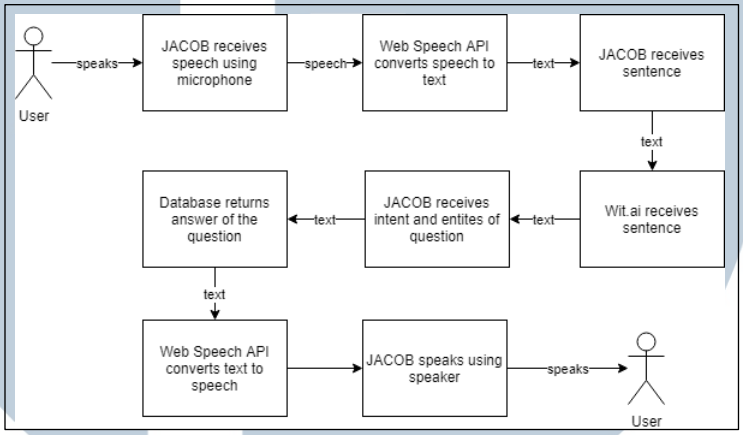
2.1 JACOB

JACOB merupakan sebuah aplikasi *chatbot* berbasis suara yang digunakan untuk memperoleh informasi seputar program studi Informatika *Dual Degree* pada Marketing Universitas Multimedia Nusantara. Informasi program studi Informatika berupa dosen, kurikulum, mata kuliah, fasilitas, peminatan, dan karir. JACOB menerima masukan dan memberikan keluaran berupa suara dalam bahasa Inggris. *Voice chatbot* JACOB dikembangkan berbasis web agar mudah diakses diberbagai *platform* seperti yang diteliti oleh Steven Wijaya (2019a). Aplikasi *voice chatbot* ini dikembangkan dengan menggunakan *database* MySQL dan platform Wit.ai. Basis pengetahuan JACOB disimpan di dalam *database* MySQL (Wijaya, 2019a).

Wit.ai adalah *cloud platform Natural Language Understanding* (NLU) milik Facebook, seperti DialogFlow milik Google. Perbedaan wit.ai dengan tool NLU lain adalah Wit.ai lebih berfokus dalam mencari arti dalam suatu kata (Canonico dan Russis, 2018). Platform Wit.ai menggunakan integrasi *webhook* dengan mengirimkan informasi melalui *web service*. Kemudian, dilakukan penentuan konteks percakapan berupa *intent* dan *entities* oleh platform Wit.ai. Dalam penentuan *intent* dan *entities*, Wit.ai menggunakan dan mempelajari dari contoh kalimat serupa yang pernah digunakan sebelumnya (Rahman dkk., 2017).

Dalam cara kerja JACOB yang ditunjukkan dalam Gambar 2.1, JACOB menerima masukan berupa suara dari pengguna. Kemudian, suara yang didengar diubah menjadi teks menggunakan *library* Web Speech API, berupa *Speech*

Recognition. Kalimat yang diterima oleh JACOB dikirimkan ke Wit.ai dan Wit.ai menentukan konteks kalimat. Hasil yang didapat dari Wit.ai dicocokkan dengan *database* untuk mendapatkan respon yang sesuai. Langkah terakhir, JACOB mengembalikan respon suara dengan menggunakan *library* Web Speech API, berupa *Speech Synthesis* untuk mengubah teks menjadi suara.



Gambar 2.1 Alur kerja *voice chatbot* JACOB (Wijaya, 2019a)

Di dalam aplikasi JACOB terdapat tiga jenis pengguna, yaitu pengguna biasa, admin, dan super admin. Pengguna biasa hanya dapat melakukan komunikasi atau interaksi tanya jawab dengan aplikasi JACOB. Jenis pengguna admin dan super admin memiliki hak untuk menambahkan jawaban yang digunakan sebagai basis pengetahuan JACOB. Super admin memiliki satu hak tambahan, yaitu dapat menambahkan admin baru. Fitur pada sistem admin yang hanya bisa diakes oleh admin dan superadmin terdiri dari lihat daftar jawaban, pengubahan pertanyaan, lihat daftar pertanyaan, pengubahan pertanyaan, lihat *entities*, pengubahan *entities*, lihat log percakapan, dan hapus log percakapan, *login*, *logout*, lihat halaman *about* (Wijaya, 2019a).

2.2 Kecerdasan Buatan

Kecerdasan buatan atau *Artificial Intelligence* diciptakan pada tahun 1950-an oleh John McCarthy dari Dartmouth College (Halper, 2017). *Artificial Intelligence* (Winston, 1993) adalah "*the study of the computations that make it possible to perceive, reason, and act*". Oleh sebab itu, komputer atau teknologi yang mengimplementasikan kecerdasan buatan dapat memiliki wawasan dan dapat bertingkah laku mendekati manusia. Berdasarkan tingkah laku manusia (Russell dan Norvig, 2010) terdapat empat kelompok atau kategori dalam kecerdasan buatan, yaitu sistem yang berperilaku seperti manusia, sistem yang berpikir seperti manusia, sistem yang berpikir rasional, dan sistem yang berperilaku secara rasional. Sistem yang cerdas tersebut telah melewati proses untuk memperoleh kemampuan dari (Russell dan Norvig, 2010):

- *natural language processing* untuk dapat melakukan komunikasi;
- *knowledge representation* untuk menyimpan informasi yang diketahui atau didengar;
- *automated reasoning* untuk membuat kesimpulan baru dari informasi yang tersimpan;
- *machine learning* untuk beradaptasi dengan keadaan baru dan untuk mendeteksi dan meramalkan kemungkinan pola.

2.2.1 Chatbot

Chatbot merupakan sebuah *conversational software agent*, yang dapat berinteraksi dengan pengguna menggunakan bahasa alami. Salah satu penggunaan *chatbot* adalah sebagai media untuk melakukan tanya jawab sehingga *chatbot* dapat dikatakan sebagai *Question Answering System* (QAS). *Question Answering*

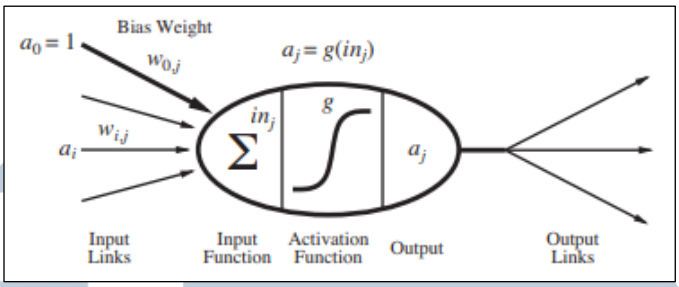
System (QAS) merupakan sistem yang menerima pertanyaan pengguna dalam bahasa alami dan memberikan jawaban yang tepat. Tujuan utama dari QAS adalah untuk mendapatkan jawaban dari pertanyaan daripada mendapatkan satu dokumen penuh (Reddy & Madhavi, 2017). Terdapat dua jenis *chatbot*, yaitu *chatbot* yang menetapkan beberapa aturan dan *chatbot* yang menggunakan *machine learning* atau *artificial intelligence* (Burns, 2017).

2.2.2 Jaringan Saraf Tiruan

Jaringan saraf tiruan (*Artificial Neural Network* atau ANN) adalah sebuah model komputasi yang terinspirasi dari struktur jaringan saraf biologis. Pembelajaran dan penelitian mengenai jaringan saraf tiruan telah dilakukan sejak pertengahan abad ke-20. Jaringan saraf tiruan (JST) dapat digambarkan seperti *graph* dengan sebuah neuron menunjukkan *node* atau unit dan sebuah *link* di antara kedua neuron menunjukkan *edge* (Shalev-Shwartz & Ben-David, 2014). Unit digambarkan dengan simbol lingkaran dan untuk simbol kotak yang terdiri dari kumpulan unit disebut dengan *layer*. Setiap *link* memiliki bobot $w_{i,j}$ yang menyatakan kekuatan dari koneksi tersebut (Russell dan Norvig, 2010).

A. Model Sel Saraf (Neuron)

Model sel saraf (neuron) merupakan sebuah sel sama seperti pada jaringan saraf biologis. Satu neuron terdapat fungsi masukan (*input* atau *summing function*), fungsi aktivasi, dan nilai aktivasi (a_j). Nilai aktivasi berupa keluaran yang dihasilkan dari suatu neuron (Russell dan Norvig, 2010).



Gambar 2.2 Model jaringan saraf tiruan (Russell dan Norvig, 2010)

Tiap neuron menerima nilai masukan dari keluaran neuron sebelumnya yang terkoneksi. Ketika neuron menerima berbagai input, setiap sinyal input akan dikalikan dengan bobot tiap *link* atau *edge* kemudian dijumlahkan oleh *summing function*. Setelah dijumlahkan, variabel akan dihitung dalam fungsi aktivasi dan akan dibandingkan dengan nilai *threshold* (batas ambang). Jika hasil keluaran memiliki nilai yang lebih besar dari batas ambang, maka neuron tersebut akan diaktifkan. Jika tidak, maka neuron tersebut akan mengirimkan keluaran melalui bobot-bobot keluaran ke semua neuron yang berhubungan dengannya melalui *link*. Proses tersebut disebut dengan istilah *feed-forward* (ReNom, 2018).

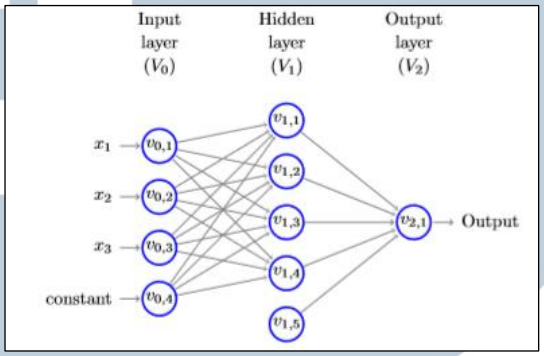
B. Arsitektur Jaringan Saraf Tiruan

Dalam jaringan saraf tiruan terdapat beberapa jenis arsitektur jaringan (Russell dan Norvig, 2010).

a. Single Layer dan Multi Layer Feed-forward Network

Jaringan *feed-forward* hanya memiliki satu arah berbentuk seperti *graph* asiklik berarah. Setiap neuron menerima masukan dari neuron-neuron sebelumnya dan mengirimkan hasilnya ke neuron setelahnya. Proses tersebut dilakukan tanpa adanya perulangan (*loops*). Jaringan ini terbagi menjadi dua, yaitu *single layer* dan *multi layer*. Perbedaan antara jaringan *single layer* dan

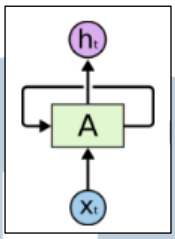
multi layer adalah pada jumlah *layer*-nya. Pada jaringan *single layer* hanya memiliki *layer* dari unit masukan dan *layer* dari unit keluaran sehingga setiap unit atau neuron langsung terhubung dari jaringan *input* ke *output*. Jaringan *multi layer* memiliki tambahan satu atau lebih *layer* dari unit tersembunyi (*hidden unit*) yang terletak di antara *layer* dari unit masukan dan unit keluaran. Salah satu contoh algoritma untuk jaringan ini adalah *back-propagation*.



Gambar 2.3 Jaringan layar jamak (Shalev-Shwartz dan Ben-David, 2014)

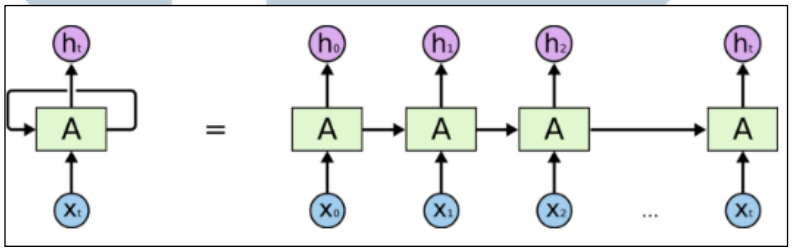
b. Recurrent Network

Berbeda dengan jaringan *feed-forward*, jaringan *recurrent* memberikan hasil kelurannya sebagai nilai *input*. Hal ini berarti bahwa tingkat aktivasi jaringan membentuk sistem dinamis yang dapat mencapai keadaan stabil atau sebaliknya. Selain itu, respon jaringan terhadap masukan yang diberikan tergantung pada keadaan awal, yang mungkin tergantung pada masukan sebelumnya. Jaringan ini dapat mendukung *short-term memory* sehingga model jaringan ini lebih menarik sebagai model otak (Russell dan Norvig, 2010).



Gambar 2.4 Jaringan *recurrent* dengan *looping* (Olah, 2017)

Berdasarkan Gambar 2.4, $\boldsymbol{A}$ menerima masukan dari $\boldsymbol{x_t}$ dan memberikan keluaran berupa nilai $\boldsymbol{h_t}$. Sebuah *loop* mengizinkan informasi dikirimkan dari satu jaringan ke jaringan lainnya. Menurut Cristhoper Olah (2017), jaringan saraf tiruan *recurrent* dapat digambarkan sebagai banyak salinan dari jaringan yang sama dengan masing-masing jaringan mengirimkan pesan ke jaringan selanjutnya.



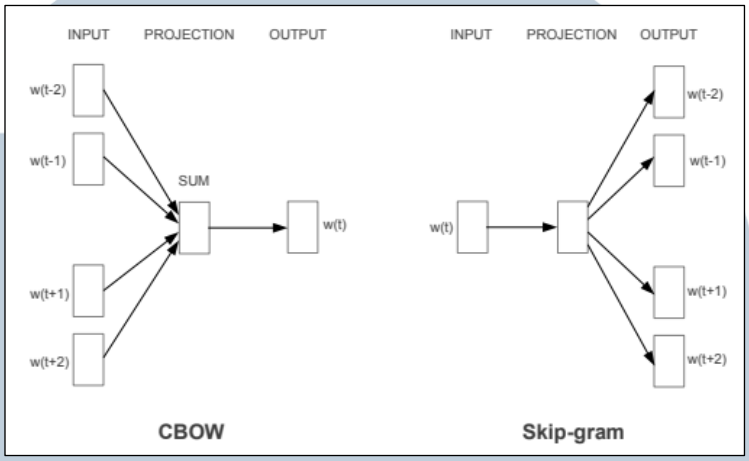
Gambar 2.5 Jaringan recurrent jika tidak dalam bentuk loop (Olah, 2017)

Dengan adanya bentuk jaringan yang seperti rantai ini mengungkapkan bahwa jaringan jaringan saraf sangat terikat dengan urutan (*sequence*). *Recurrent Neural Network* (RNN) dapat belajar menggunakan informasi sebelumnya.

C. Word Embedding

Word embedding adalah representasi kata yang dapat dibuat dari *raw text*, atau dokumen, berdasarkan konteks kalimat. Representasi untuk setiap kata berupa vektor yang biasanya berisi bilangan riil. Salah satu kegunaan dari

representasi vektor kata (*word vectors*) adalah nilainya dapat dihitung untuk mencari kemiripan kata (Leeuwenberg dkk., 2016).



Gambar 2.6 *Continuous bag-of-word* (kiri), dan *skip-gram* (kanan) (Mikolov dkk., 2013a)

Mikolov dkk. (2013a) mengembangkan dua arsitektur model untuk menentukan nilai vektor dari sebuah kata dengan arsitektur *continuous bag-of-words model* (CBoW) dan *skip gram model* (SG) yang ditunjukkan pada Gambar 2.6. Berdasarkan nilai vektor dapat dihitung kemiripan kata dan juga kemiripan antara pasangan kata (Leeuwenberg dkk., 2016). Kedua jenis model vektor kata dapat dibangun dengan menggunakan *feedforward neural network* sederhana dan dilihat berdasarkan konteksnya. CBoW dioptimalkan untuk memprediksi kata dari kata-kata disekitarnya atau konteksnya, sedangkan SG dioptimalkan untuk memprediksi konteks dari kata sekarang dan memprediksi kata-kata disekitarnya (Mikolov dkk., 2013a).

Dalam upaya mendapatkan hasil vektor kata yang baik terhadap jaringan CBoW dan SG terdapat dua jenis metode pelatihan, yaitu *hierarchical softmax* dan *negative sampling*. Dalam *hierarchical softmax*, bobot diperbaharui berdasarkan *maximization of log-likelihood*. Dalam *negative sampling*, bobot

diperbaharui berdasarkan pada apakah kata target yang diambil dari set latih, atau dari *random distribution* (Mikolov dkk., 2013b).

2.3 Parafrasa Kalimat

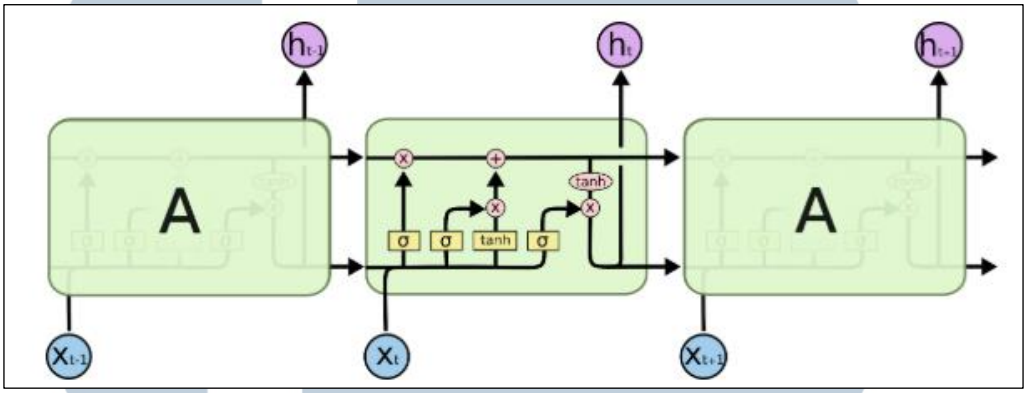
Parafrasa berarti memiliki arti yang sama, tetapi menggunakan kalimat yang berbeda. Parafrasa dapat dilakukan dengan cara mengubah kata dengan sinonimnya dan mengganti kalimat menjadi kalimat aktif atau pasif (Vila dkk., 2011). Selain itu, parafrasa dapat dilakukan dengan mengganti dengan antonim, mengubah kata ganti orang, mengganti *pronoun* dengan frasa *noun*, mengganti kata menjadi lebih umum atau lebih spesifik, dan lain-lain (Bhagat dan Hovy, 2013).

Kehadiran parafrasa dalam *Natural Language Processing* (NLP) memiliki potensi besar untuk peningkatan sistem dalam pemahaman dan *generation*, seperti tanya jawab, peringkasan, atau mesin terjemahan (Vila dkk., 2011; Bhagat dan Hovy, 2013). Salah satu alasan sistem pengenalan parafrasa susah dikembangkan adalah karena parafrasa sulit dikenali. Meskipun interpretasi yang ketat dari istilah "parafrasa" cukup sempit karena memerlukan makna yang persis sama. Dalam literatur linguistik, parafrasa paling sering ditandai dengan perkiraan kesamaan arti di seluruh kalimat atau frasa (Bhagat dan Hovy, 2013).

2.4 Long Short-Term Memory (LSTM)

LSTM adalah salah satu jenis dari arsitektur *Recurrent Neural Network* (RNN) yang dirancang untuk menghindari masalah dependensi jangka panjang. Jaringan ini diperkenalkan pertama kali oleh Hochreiter dan Schmidhuber pada tahun 1997. Dalam satu neuron, jaringan LSTM memiliki empat lapisan yang saling berinteraksi, yaitu tiga lapisan sigmoid dan satu lapisan tanh. Ketika suatu

nilai dihitung menggunakan fungsi aktivasi sigmoid, nilai yang dikembalikan berada di rentang nol sampai satu. Untuk fungsi aktivasi tanh mengembalikan nilai antara negatif satu sampai satu (Olah, 2017). Untuk ilustrasi jaringan LSTM ditunjukkan pada Gambar 2.7.



Gambar 2.7 Jaringan LSTM (Olah, 2017)

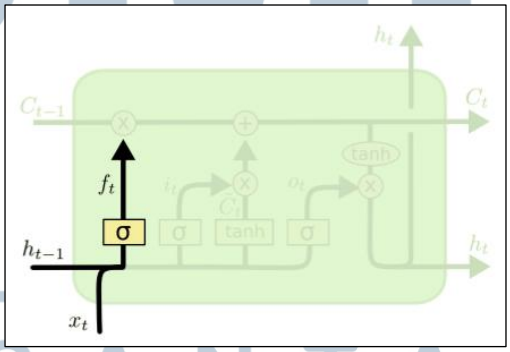
Keterangan mengenai notasi yang digunakan pada jaringan LSTM (lihat Gambar 2.7) adalah sebagai berikut (Olah, 2017).

- *Neural network layer* merupakan jaringan saraf yang dipelajari dan dilambangkan dengan kotak berwarna kuning.
- *Vector transfer* merupakan garis yang membawa seluruh vektor, dari keluaran satu neuron ke masukan lainnya. Dinotasikan dengan anak panah tanpa cabang ataupun garis penggabungan.
- *Pointwise operation* adalah operasi yang dilakukan seperti penambahan vektor. Dinotasikan dengan lingkaran berwarna merah muda.
- *Concatenate* merupakan proses penggabungan yang menandakan rangkuman. Dilambangkan dengan garis yang mengalami penggabungan.
- *Copy* merupakan proses penyalinan nilai untuk dibawa ke lokasi yang berbeda. Dilambangkan dengan garis yang bercabang.

- x_t merupakan notasi yang menyatakan masukan untuk tiap unit pada *time step* t dan nilainya berupa vektor.
- h_t merupakan notasi yang menyatakan *hidden state* yang dihasilkan oleh tiap unit pada *time step* t dan nilainya berupa vektor.
- C_t merupakan notasi yang menyatakan *cell state* yang dihasilkan oleh tiap unit pada *time step* t dan nilainya berupa vektor.

LSTM memiliki kemampuan untuk menghapus atau menambah informasi ke *cell state*, yang diatur oleh struktur yang disebut *gate*. Dalam LSTM terdiri atas tiga *gate*, yaitu *forget gate*, *input gate*, dan *output gate*. *Gate* tersusun dari lapisan jaringan saraf sigmoid dan operasi *pointwise multiplication*. Pada lapisan sigmoid (*forget gate layer*) dihasilkan nilai antara nol sampai satu (Olah, 2017). Jika mendekati nilai nol maka informasi dilupakan dan jika mendekati nilai satu maka informasi masih relevan sehingga diteruskan ke *cell state* (Gers dkk., 1999; Olah, 2017). Ketika hasil dari fungsi aktivasi sigmoid mendekati nol dapat dikatakan bahwa *gate* tertutup sehingga informasi yang sudah tidak relevan tidak dilanjutkan ke *cell state* (Gers, 1999). Fungsi tiga *gate* yang terdapat pada LSTM adalah untuk melindungi dan mengontrol *cell state* (Olah, 2017).

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f) \quad \ldots(2.1)$$



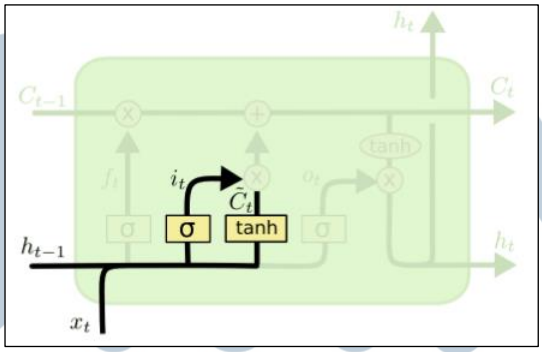
Gambar 2.8 Ilustrasi Rumus (2.1) (Olah, 2017)

Pada Rumus (2.1) terdapat beberapa notasi yang dipakai dalam *forget gate*. Notasi tersebut adalah f_t merupakan notasi untuk nilai yang dihasilkan pada *forget gate*; σ merupakan fungsi aktivasi sigmoid; W_f merupakan matriks bobot pada *forget gate*; dan b_f merupakan nilai bias pada *forget gate* (Olah, 2017).

Berdasarkan penjelasan Rumus (2.1) dan Gambar 2.8 terdapat penggabungan antara *hidden state* sebelumnya dan *input* saat ini, yang berupa nilai vektor. Kemudian, hasil dari penggabungan dua vektor melewati *forget gate* yang terdiri lapisan sigmoid. *Forget gate* digunakan untuk *reset* informasi pada *memory blocks* sehingga dalam lapisan sigmoid ditentukan apakah informasi diteruskan atau dilupakan (Gers dkk., 1999; Olah, 2017). Ketika informasi sudah tidak relevan, informasi tersebut dilupakan, tetapi tidak akan secara eksplisit dilupakan sampai *forget gate* telah belajar untuk lupa (Gers dkk., 1999).

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i) \quad \ldots(2.2)$$

$$\acute{C}_t = tanh(W_C . [h_{t-1}, x_t] + b_C) \quad \ldots(2.3)$$



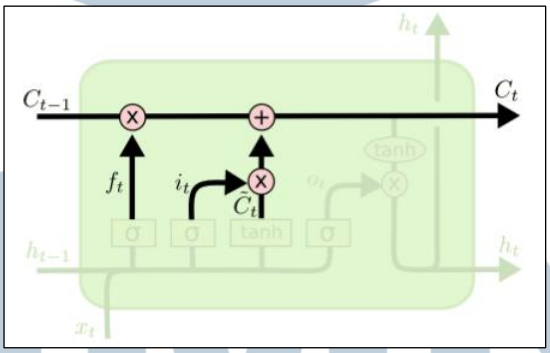
Gambar 2.9 Ilustrasi Rumus (2.2) dan (2.3) (Olah, 2017)

Pada Rumus (2.2) terdapat beberapa notasi yang dipakai dalam *input gate*. Notasi tersebut adalah i_t merupakan notasi untuk nilai yang dihasilkan pada *input gate*; σ merupakan fungsi aktivasi sigmoid; W_i merupakan matriks bobot pada *input gate*; dan b_i merupakan nilai bias pada *input gate*. Berdasarkan Rumus (2.3),

notasi yang digunakan adalah tanh yang merupakan fungsi aktivasi *tangent hyperbolic* (Olah, 2017).

Berdasarkan ilustrasi pada Gambar 2.9, penjelasan untuk Rumus (2.2) dan (2.3) merupakan rumus ketika kedua vektor *hidden state* sebelumnya dan *input* melewati *input gate*. Pada *input gate* menggunakan fungsi aktivasi sigmoid dan tanh. Pada lapisan tanh membuat nilai vektor *candidate* ($\acute{C}_t$), yang dapat ditambahkan ke dalam *state*. Kemudian, hasil dari kedua fungsi aktivasi digabungkan dengan menggunakan operasi *pointwise multiplication* untuk memutuskan informasi yang akan diperbaharui ke dalam *state* (Olah, 2017). *Input gate* bertugas untuk melindungi informasi pada *cell state* dari gangguan *input* yang tidak relevan (Hochreiter dan Schmidhuber, 1997).

$$C_t = f_t * C_{t-1} + i_t * \acute{C}_t \quad \ldots(2.4)$$

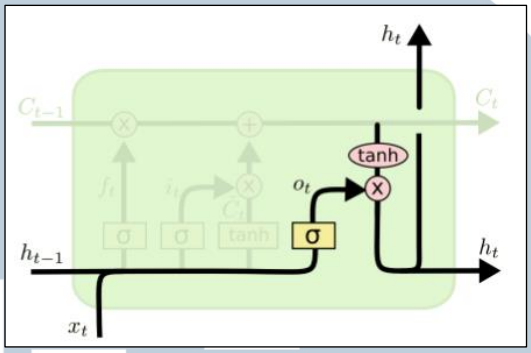


Gambar 2.10 Ilustrasi Rumus (2.4) (Olah, 2017)

Rumus (2.4) dan ilustrasi pada Gambar 2.10 menjelaskan hasil dari *forget gate* yang telah didapatkan pada Rumus (2.1) dilakukan *pointwise multiplication* dengan nilai *cell state* sebelumnya. Nilai dari dua operasi *pointwise multiplication* yang terjadi pada *forget gate* dan *input gate* kemudian dijumlahkan, sehingga mendapatkan nilai *cell state* terbaru (Olah, 2017).

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad \ldots(2.5)$$

$$h_t = o_t * \tanh(C_t) \quad \ldots(2.6)$$



Gambar 2.11 Ilustrasi Rumus (2.5) dan (2.6) (Olah, 2017)

Pada Rumus (2.5) terdapat beberapa notasi yang dipakai dalam *output gate*. Notasi tersebut adalah o_t merupakan notasi untuk nilai yang dihasilkan pada *output gate*; σ merupakan fungsi aktivasi sigmoid; W_o merupakan matriks bobot pada *output gate*; dan b_o merupakan nilai bias pada *output gate* (Olah, 2017).

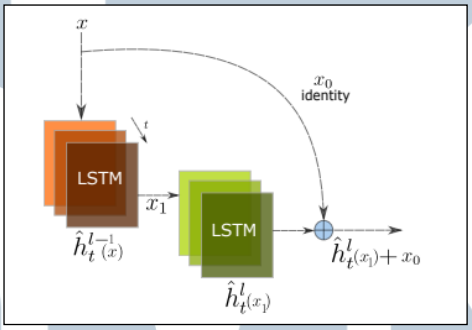
Rumus (2.5) merupakan proses yang terjadi pada *output gate* dan di dalam *output gate* terdapat perhitungan fungsi aktivasi sigmoid terhadap penggabungan nilai *hidden state* sebelumnya dan *input*. Kemudian dilanjutkan dengan operasi *pointwise multiplication* terhadap nilai dari *output gate* dan nilai fungsi aktivasi tanh terhadap nilai *cell state* yang baru sehingga dihasilkan nilai *hidden state* baru, sesuai dengan Gambar 2.11 (Olah, 2017). *Output gate* memiliki tugas untuk menjaga unit lainnya dari gangguan informasi yang tidak relevan yang tersimpan pada *cell state* saat ini (Hochreiter dan Schmidhuber, 1997).

2.4.1 Stacked Residual LSTM

Model *Stacked Residual* LSTM adalah model yang dikembangkan oleh Prakash dkk. (2016) dengan tujuan untuk menghasilkan parafrasa pada kalimat. Ide awal penggunaan residu pada model ini dikarenakan He dkk. (2015) telah

menunjukkan bahwa penambahan residu *x* secara eksplisit ke fungsi yang dipelajari memungkinkan pelatihan *deeper network* terjadi tanpa adanya *overfitting* data (Prakash dkk., 2016). *Overfitting* ditandai dengan *training error* yang sangat kecil dan *validation error* yang sangat besar pada waktu yang bersamaan (Bilbao dan Bilbao, 2017). Jaringan *deep learning* yang dikembangkan oleh He dkk. (2015) bernama ResNet (Prakash dkk., 2016).

Ketika menggunakan *stacking multiple layer*, jaringan sering mengalami *degradation problem*. *Degradation problem* terjadi karena adanya penurunan tingkat konvergensi kesalahan pada saat pelatihan yang rendah, dan masalah ini berbeda dengan masalah *vanishing gradient*. Penambahan *residual connections* dapat membantu mengatasi masalah tersebut (Prakash dkk., 2016).



Gambar 2.12 Sebuah unit *Stacked Residual* LSTM (Prakash dkk., 2016)

Pada penelitian Prakash dkk. (2016) menggunakan *stacked* LSTM dengan jumlah *layer* sebanyak empat. Penambahan *residual connections* dilakukan dengan *pointwise addition* dan ditambahkan setiap dua *layer* seperti yang dapat dilihat pada Gambar 2.12. Dimensi masukan (x_t) dan keluaran (h_t) harus memiliki panjang yang sama. Fungsi $\hat{\mathrm{h}}$ yang dipelajari untuk *layer* dengan *residual connection* ditunjukkan pada Rumus (2.7) berikut.

$$\hat{\mathrm{h}}_t^{(l)} = f_h^l\left(h_t^{(l-1)}, h_{t-1}^{(l)}\right) + x_{l-n} \quad \ldots(2.7)$$

Berdasarkan Rumus (2.7), *layer* masukan pertama pada *time step* t dilalui dari *hidden state layer* sebelumnya $h_t^{(l)}$, dengan l dinotasikan sebagai *layer*. Fungsi ĥ untuk *layer* l diperbaharui dengan nilai x_{l-n} dan x_i merupakan masukan untuk *layer* $i + 1$. *Residual connection* ditambahkan setelah setiap n *layers*. Jika n lebih besar dari 3 maka dibutuhkan komputasi yang tinggi. Selain itu, Jika n = 1 maka akan menjadi jaringan LSTM biasa dengan *bias* yang tergantung pada masukan x (Prakash dkk., 2016).

Implemetasi *code* untuk *pretrained model Stacked Residual* LSTM terdapat pada platform web GitHub (Prakash, 2018). Berdasarkan implementasi tersebut, semua kata diubah menjadi *one-hot vector* saat melakukan proses pembelajaran. Variabel dalam pelatihan yang digunakan adalah *learning rate* sebesar 0,001, jumlah *epoch* yang digunakan 10.000, *batch size* sebesar 32, dan *optimizer* yang digunakan adalah Adam. Jumlah unit LSTM pada setiap *layer* sama, yaitu 256 (Prakash, 2018).

Dalam upaya menghasilkan parafrasa yang optimal digunakan algoritma *beam search*. Selama pelatihan, *loss function* yang digunakan adalah *perplexity*. *Perplexity* mengukur ketidakpastian dalam model bahasa, sesuai dengan banyak bit rata-rata yang dibutuhkan untuk meng-*encode* setiap kata pada model bahasa. Semakin kecil nilai *perplexity* menandakan semakin baik model tersebut (Prakash dkk., 2016).

2.5 Text Summarization

Ringkasan adalah sebuah teks yang dibuat dari satu atau lebih teks, yang berisi informasi penting dari teks asli dalam bentuk yang lebih singkat. Kelebihan dari menampilkan ringkasan adalah mengurangi waktu membaca. Metode

peringkasan teks terdiri atas dua, yaitu peringkasan ekstraktif dan abstraktif. Metode peringkasan ekstraktif dilakukan dengan memilih kalimat penting dari kalimat asli sehingga menjadi kalimat yang lebih singkat. Ringkasan yang menggunakan metode peringkasan abstraktif mengerti konsep utama dari dokumen dan kemudian membuat kalimat dengan menggunakan bahasa alami yang jelas (Babar, 2013).

Terdapat dua kelompok peringkasan teks, yaitu induktif dan informatif. Peringkasan induktif hanya menampilkan ide utama teks kepada pengguna. Panjang kalimat dari ringkasan jenis ini sebesar lima sampai sepuluh persen dari teks utama. Peringkasan dengan jenis informatif memiliki panjang kalimat sebesar 20 sampai 30 persen dari teks utama (Babar, 2013). Dalam penelitian yang dilakukan oleh Victoria McCargar (2005) menggunakan jumlah kalimat sebesar 25% dari teks sebagai ringkasan.

2.5.1 TextRank

Algoritma TextRank merupakan algoritma peringkat berbasis *graph* (*graph-based ranking algorithm*) yang dapat digunakan untuk menentukan *vertex* yang penting dalam sebuah graf. Biasanya model ini digunakan untuk pemungutan suara dan rekomendasi. Algoritma ini menerapkan algoritma PageRank untuk menentukan peringkat pada kalimat (Mihalcea dan Tarau, 2004). Hasil dari algoritma PageRank adalah nilai probabilitas untuk setiap kalimat. Kalimat yang memiliki probabilitas terbesar dapat dikatakan sebagai kalimat yang paling relevan dan kalimat penting dalam sebuah teks (Ferreira dkk., 2016). Rumus untuk algoritma PageRank ditunjukkan pada Rumus (2.8) berikut (Li dan Zhao, 2016).

$$PR(p) = (1 - d) + d \sum_{i=1}^{n} \frac{PR(T_i)}{C(T_i)} \quad \ldots(2.8)$$

Berdasarkan Rumus (2.8), notasi $PR(p)$ merupakan nilai (*score*) untuk *page* p; Ti merupakan *page* lain yang mengarah ke *page* p dengan i dimulai dari angka 1 sampai n; d merupakan probabilitas pengguna untuk pergi ke sebuah *page* secara acak dengan nilai antara 0 sampai 1; $C(Ti)$ merupakan jumlah *link* yang keluar dari *page* Ti; $PR(Ti)/C(Ti)$ merupakan nilai PageRank dengan *page* Ti yang terhubung dengan *page* p (Li dan Zhao, 2016). Notasi d disebut *damping factor* dengan nilai biasanya sebesar 0,85 (Mihalcea dan Tarau, 2004).

Algoritma TextRank dapat digunakan untuk memberikan peringkat pada teks, yaitu ektraksi kata kunci (*keyword extraction task*) dan ekstraksi kalimat (*sentence extraction task*). Dalam melakukan dua tugas tersebut sepenuhnya dilakukan dengan cara *unsupervised* dan tidak ada pelatihan. Dalam ekstraksi kata kunci dilakukan pemilihan *keyphrases* yang mewakili kalimat. Dalam ekstraksi kalimat dilakukan pencarian kalimat yang paling penting dalam sebuah teks sehingga dapat dipakai untuk membuat ringkasan atau disebut juga *extractive summary* (Mihalcea dan Tarau, 2004; Ferreira dkk., 2016). Rumus untuk algoritma TextRank ditunjukkan pada Rumus (2.9) berikut (Li dan Zhao, 2016).

$$TR(V_i) = (1 - d) + d \sum_{V_j \in In(V_i)} \frac{W_{ji}}{\sum_{V_k \in Out(V_j)} W_{jk}} TR(V_j) \quad \ldots(2.9)$$

Berdasarkan Rumus (2.9), notasi $TR(V_i)$ merupakan nilai (*score*) untuk *vertex* V_i. *Edge* menghubungkan dua *vertex* i dan j dengan bobot W_{ij}. $In(V_i)$ merupakan kumpulan *vertex* yang mengarah ke *vertex* V_i; $Out(V_i)$ merupakan kumpulan *vertex* yang pergi dari *vertex* V_i. Notasi d pada algoritma TextRank sama seperti pada algoritma PageRank (Li dan Zhao, 2016).

Penerapan algoritma peringkat berbasis graf untuk teks bahasa alami terdiri dari beberapa langkah utama, yaitu (1) identifikasi kata, kalimat, dll yang paling berperan dan tambahkan ke dalam *vertex* pada sebuah graf, (2) identifikasi relasi yang menghubungkan antar kata, kalimat, atau yang lain, dan gunakan relasi ini untuk menggambarkan *edge* antara *vertex* dalam sebuah graf, (3) lakukan perulangan menggunakan algoritma sampai ditemukan konvergensi, dan (4) urutkan *vertex* berdasarkan nilai akhir dan lakukan pemilihan dengan peringkat tertinggi. Konvergensi dicapai ketika *error rate* untuk *vertex* di sebuah *graph* telah berada di bawah nilai *threshold*. *Error rate* dihitung berdasarkan perbedaan antara *score* saat ini ($TR^{k+1}(V_i)$) dengan *score* sebelumnya ($TR^{k}(V_i)$) (Mihalcea dan Tarau, 2004).

2.5.2 Cosine Similarity

Metode *Cosine Similarity* menghitung kemiripan antara dua buah vektor dengan mengukur nilai *cosine* sudut antara dua vektor tersebut. Dalam perhitungan nilai menggunakan metode ini menunjukkan jika hasil semakin dekat dengan angka satu maka semakin mirip dua vektor tersebut (Bigdeli dan Bahmani, 2008). Rumus *Cosine Similarity* ditunjukkan pada Rumus (2.10).

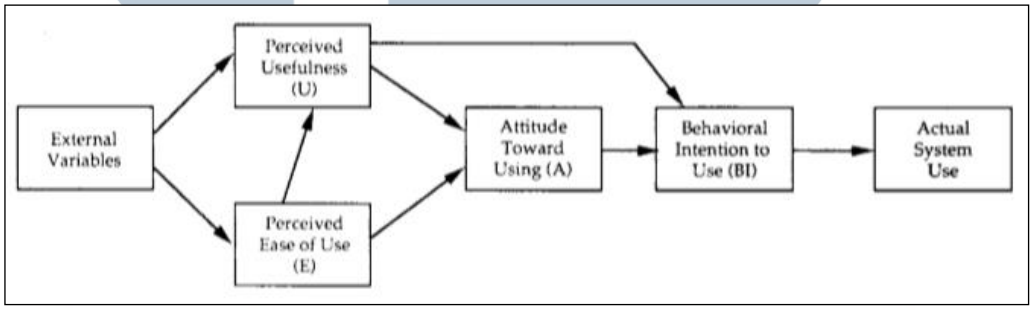
$$sim(i,j) = \cos(\vec{i} \cdot \vec{j}) = \frac{\vec{i}.\vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2} \quad \ldots(2.10)$$

Berdasarkan Rumus (2.10), lambang . pada $\vec{i}.\vec{j}$ menyatakan *dot-product* antara dua vektor. Lambang $||\vec{i}||_2$ merupakan notasi untuk panjang vektor yang dilakukan dengan penjumlahan dari kuadrat tiap nilai pada vektor tersebut. Metode ini dapat digunakan untuk menghitung tingkat kemiripan antara dua kata

atau dua kalimat. Contohnya seperti penelitian yang dilakukan oleh Gokul, Akhil, dan Shiva (2017) untuk mencari kemiripan kalimat dengan bahasa Malayalam.

2.6 Technology Acceptance Model (TAM)

Technology Acceptance Model (TAM) mulanya dikenalkan oleh Davis pada tahun 1989 dan digunakan untuk memprediksi kemungkinan teknologi baru digunakan dalam suatu kelompok atau organisasi. TAM merupakan hasil dari adaptasi *Theory of Reasoned Action* (TRA). TRA adalah sebuah model yang dipelajari dari psikologi sosial yang berkaitan dengan faktor penentu perilaku yang dilakukan secara sadar (Davis dkk., 1989). Konsep utama dan struktur TAM dapat ditunjukkan dalam Gambar 2.13.



Gambar 2.13 *Technology Acceptance Model* (Davis dkk., 1989)

Hasil evaluasi TAM ditentukan berdasarkan dua variabel persepsi ketika pengguna akan menggunakan sistem informasi baru, yaitu *perceived userfulness* dan *perceived ease of use*. *Perceived usefulness* merupakan derajat bahwa pengguna percaya jika sistem tertentu dapat meningkatkan performa kerja. *Perceived ease of use* merupakan derajat bahwa pengguna percaya dengan menggunakan sistem tertentu dapat bebas dari usaha. TAM mendalilkan bahwa *actual technology usage* ditentukan oleh *behavioral intention to use*, yang dilihat sebagai sikap pengguna terhadap penggunaan teknologi (*attitude toward using*)

dan manfaat yang dirasakan (*perceived usefulness*). Dalam TAM, *perceived usefulness* juga dipengaruhi oleh *perceived ease of use* karena semakin mudah sistem digunakan maka semakin berguna (Tang dan Chen, 2011).

Berdasarkan evaluasi yang dikembangkan oleh Davis (1989), beberapa pertanyaan evaluasi yang dapat dijadikan acuan dalam penelitian ini dengan masing-masing variabel ditampilkan sebanyak dua dari 14 pertanyaan adalah sebagai berikut.

a. Perceived ease of use
    a) Saya sering bingung ketika menggunakan sistem *electronic mail*.
    b) Saya membuat kesalahan secara berkala ketika menggunakan *electronic mail*.

b. Perceived usefulness
    a) Perkerjaan saya lebih sulit dikerjakan tanpa adanya *electronic mail*.
    b) Penggunaan *electronic mail* memberikan kontrol yang lebih baik terhadap pekerjaan saya.

Skala Likert adalah skala yang dirancang untuk mengukur sikap secara ilmiah yang dapat diterima dan divalidasi (Joshi dkk., 2015). Cara untuk mengukurnya adalah dengan menetapkan variabel penelitian yang kemudian dibentuk dalam beberapa pertanyaan atau pernyataan. Jawaban dari pertanyaan atau pernyataan memiliki bobot nilai dari positif hingga negatif.