2. LANDASAN TEORI

2.1 Tinjauan Pustaka

2.1.1 Admisi dan Humas Universitas Kristen Petra

Universitas Kristen Petra memiliki beberapa layanan untuk membantu masyarakat dalam memperoleh informasi seputar UK Petra. 2 layanan diantaranya adalah Admisi dan Humas Universitas Kristen Petra. Pihak admisi lebih diperuntukan dalam menerima pertanyaan-pertanyaan terkait ujian-ujian untuk mahasiswa yang ingin masuk ke Universitas Kristen Petra. Ujian ini dapat berupa ujian tulis untuk beberapa jalur penerimaan mahasiswa (jalur umum / jalur prestasi / jalur Kerjasama) dan ujian khusus untuk beberapa program studi yang terkait. Pihak Humas / Hubungan masyarakat Universitas Kristen Petra merupakan pihak yang menerima pertanyaan-pertanyaan yang bersifat umum. Pertanyaan ini bisa berupa biaya studi, jalur-jalur penerimaan, informasi umum mengenai Universitas Kristen Petra, informasi program studi yang dimiliki UK Petra, dan sebagainya.

2.1.2 *Artificial Intelligence*

Artificial intelligence atau yang biasa dikenal dengan nama kecerdasan buatan, merupakan salah satu bidang ilmu yang mempelajari tentang otomasi sistem. Otomasi sistem ini memiliki arti dimana sistem akan dapat bekerja secara otomatis dan tidak perlu bantuan dari manusia. Sistem ini juga akan berusaha untuk bekerja dan memiliki fungsi layaknya seorang manusia. Terobosan ilmu *AI* akhir-akhir ini menunjukan perkembangan dan improvisasi sistem dalam memiliki kemampuan layaknya manusia. Terobosan ini meliputi kemampuan sistem dalam memahami gambar *(image recognition, text recognition)*, suara *(voice recognition)*, dan pemilihan keputusan (Jarrahi, Askay, Eshraghi, & Smith, 2022). Perkembangan terus menerus dalam bidang pengetahuan ini akan sangat membantu manusia dalam pekerjaannya. Selain itu, era modernisasi akan berkembang secara pesat apabila teknologi *artificial intelligence* dapat digunakan secara maksimal dengan tujuan yang tepat.

2.1.3 *Chatbot*

Chatbot merupakan salah satu sistem *AI* yang bekerja dalam bidang pemahaman arti *text* dan suara, serta melakukan pemilihan jawaban layaknya seorang manusia. Fungsi dari sistem ini adalah untuk membantu manusia dalam memberikan jawaban tentang suatu *problem* secara instan dan akurat. *Chatbot* mulai dikembangkan sejak tahun 1966 oleh *Massachusetts*

Institute of Technology (MIT) dan diberi nama Eliza (Ahmed et al., 2022). Sejak itu, perkembangan *chatbot* menjadi semakin luas dan maju. *Chatbot* mulai digunakan dalam berbagai bidang, seperti kebutuhan rumah sakit, bank, dan sebagainya. Penggunaan sistem ini membantu perusahaan dalam menghemat biaya pekerja serta memberikan layanan yang selalu siap setiap saat.

Kekurangan yang dimiliki *chatbot* saat ini adalah kurangnya kemampuan *chatbot* dalam mendeteksi arti pertanyaan manusia. Pada awalnya, *chatbot* dapat bekerja dengan adanya *knowledge base* yang diberikan secara manual oleh peneliti. Isi dari *knowledge base* ini juga berdasarkan *pattern matching*, sehingga peneliti harus memberikan banyak kemungkinan kalimat yang mungkin ditanyakan oleh pengguna (Adamopoulou, 2020). Apabila pertanyaan pengguna memiliki susunan pertanyaan yang berbeda dari *knowledge base* namun memiliki arti yang sama, maka *chatbot* dengan *knowledge base* berbasis *pattern matching* tidak akan dapat memahami arti pertanyaan *user*. Oleh karena itu, digunakan pendekatan *machine learning*. Algoritma *machine learning* memungkinkan *chatbot* untuk dapat memahami konteks pertanyaan dari pengguna tanpa harus mengikuti aturan-aturan kompleks tentang susunan kalimat. Hal ini membuat *chatbot* menjadi lebih pintar dalam memberikan jawaban yang berhubungan dengan pertanyaan walaupun susunan pertanyaannya tidak jelas.

Namun, kehadiran *machine learning* saja tidak akan membantu *chatbot* memberikan jawaban yang lebih baik. *Chatbot* dapat bekerja dan belajar dengan adanya *dataset*. Seberapa pintar suatu *chatbot* tentu akan berhubungan erat dengan kualitas dan kuantitas dari *dataset* yang disediakan (Meyer, 2021). Apabila *dataset* yang diberikan kepada *chatbot* lengkap, rinci, dan jelas maka performa *chatbot* akan ikut meningkat. Namun apabila kualitas *dataset* yang diberikan kurang baik, maka kualitas *chatbot* akan ikut menurun walaupun algoritma *machine learning* yang digunakan canggih. Masalah ini menjadi semakin besar mengingat pertanyaan-pertanyaan yang disampaikan oleh *user* tidak selalu berupa aturan / *rule* yang jelas, melainkan berupa arti tersirat dan sentimen. Untuk *chatbot* bisa menangkap arti-arti tersebut, diperlukan data yang banyak sebagai *training.*

2.1.4 *Natural Language Processing*

Komputer tidak memiliki kemampuan dalam memahami arti bahasa manusia yang diberikan oleh penggunanya. Oleh karena itu, diperlukan suatu bidang di dalam *Artificial Intelligence* bernama *natural language processing* untuk memproses bahasa tersebut. *NLP* adalah suatu bidang yang berkaitan dengan pemahaman bahasa alami manusia dan

menggunakannya untuk melakukan interaksi antar manusia dan komputer (Gudivada & Arbabifard, 2018). Pengaplikasian *NLP* dapat digunakan untuk pemahaman teks dan suara manusia (*Natural Language Understanding)* serta pemberian jawaban berdasarkan pemahaman sistem (*Natural Language Generation)*. Penggunaan *NLP* ini juga sudah meluas, mulai dari penerjemah bahasa, melakukan *summary* paragraf, *chatbot,* hingga *speech recognition*.

2.1.5 *Text Preprocessing*

Agar suatu sistem dapat membuat suatu model yang efektif untuk digunakan, sebelumnya data perlu dilakukan *preprocessing*. Data yang digunakan untuk penyusunan *chatbot* adalah data teks, sehingga diperlukan *text preprocessing*. Ada beberapa macam *text preprocessing* yang dapat dilakukan, yaitu *tokenization, lower casing, stop words removal, stemming,* dan *lemmatization* (Harshith, 2019).

2.1.5.1 *Tokenization*

Tokenization adalah proses pemisahan suatu paragraf atau kalimat menjadi bagian lebih kecil, yang akan disebut sebagai *token*. *Tokenization* biasanya dibagi menjadi 2 bidang, yaitu *Word Tokenization* dan *Sentence Tokenization* (Kapadia, 2019). *Word Tokenization* bertugas untuk membagi kalimat menjadi kata / *token. Sentence Tokenization* bertugas untuk membagi suatu paragraf menjadi beberapa kalimat.

2.1.5.2 *Lower Casing*

Seperti dengan namanya, semua kalimat atau kumpulan kata yang digunakan dalam penelitian nantinya akan diubah menjadi huruf kecil. Proses ini dilakukan agar sistem tidak membedakan suatu kata hanya karena perbedaan huruf besar atau huruf kecil. Hal ini akan berguna dalam mengurangi fitur-fitur yang sama dan *noise* saat dilakukan penelitian kedepannya.

2.1.5.3 *Stop Words Removal*

Stop words removal adalah proses menghapus kata-kata penghubung dan kata akhir yang biasanya tidak memiliki arti yang signifikan didalamnya. Di bahasa Indonesia, contoh *stopwords* antara lain adalah “dan”, “atau”, “jika”, “maka”, “dengan”, dan lainnya (Hosea, 2020). Kata-kata tersebut sering kali ada di dalam kalimat dan apabila tidak dihapus di *text preprocessing*, akan dapat mengubah hasil akhir dari suatu model.

2.1.5.4 *Stemming*

Stemming adalah suatu proses penghapusan imbuhan awal dan akhir dari suatu kata. Tujuan dari proses ini adalah untuk mengurangi redundansi *dataset* hanya karena perbedaan imbuhan. Fungsi ini akan berguna dalam pembuatan *dataset* yang efektif saat dilakukan *training* kepada model.

2.1.5.5 *Lemmatization*

Fungsi dari *lemmatization* mirip seperti *stemming*, hanya saja *lemmatization* berusaha untuk mengembalikan suatu kata dengan imbuhan menjadi kata dasarnya. Tujuan dari proses ini sama seperti *stemming*, yaitu untuk memperoleh data berdasarkan kata dasarnya saja dan tidak dibeda-bedakan hanya karena imbuhan yang berbeda.

2.1.6 *Feature Extraction*

Feature Extraction merupakan suatu teknik pengambilan suatu ciri / fitur dari sebuah data mentah. Data yang dimaksud dapat berupa gambar, suara, maupun teks. Penggunaan teknik ini akan berdampak besar pada peningkatan akurasi dari model yang akan dibuat. Biasanya, akan diperlukan suatu tahap yang dinamakan *word embeddings.* Tahap ini adalah suatu proses yang akan mengkonversi sebuah teks menjadi angka. Tahapan ini diperlukan karena sebagian besar algoritma *machine learning* dan arsitektur *deep learning* tidak mampu melakukan proses analisis pada data berupa *string*, sehingga diperlukan angka sebagai input. Namun, cara untuk memperoleh angka tersebut bermacam-macam dan hasil yang diperoleh juga berbeda-beda tiap caranya. Cara-cara tersebut mencakup *Tokenizer, Bag of Words, Term Frequency – Inverse Document Frequency,* dan *Word2Vec*.

2.1.6.1 *Tokenizer*

Tokenizer merupakan suatu cara untuk mengubah tiap kata di dalam kalimat berupa *string* menjadi suatu angka integer. Tiap angka merepresentasikan suatu kata unik. Sehingga hasil yang diperoleh nantinya berupa suatu rangkaian integer dari kalimat yang diproses tadi. Dengan begitu, komputer dapat memahami data yang sedang digunakan dengan lebih baik. Sebelum lanjut ke tahap berikutnya, tiap *sequence* data juga akan diterapkan *padding*. *Padding* merupakan suatu cara untuk menambahkan suatu *default value* kepada tiap *sequence* data agar semua *sequence* memiliki panjang data yang sama. Hal ini dilakukan agar data tersebut dapat diproses ke tahap berikutnya yaitu *training* model. Tahap ini membutuhkan data yang akan digunakan untuk memiliki panjang / ukuran yang sama.

2.1.6.2 Bag of Words

Bag of Words merupakan suatu cara untuk mengubah suatu kalimat menjadi suatu vektor angka yang merepresentasikan jumlah kemunculan / frekuensi suatu kata (Qader, Ameen, & Ahmed, 2019). Model *Bag of Words* adalah suatu cara representasi sederhana yang biasanya digunakan dalam *Natural Language Processing* dan *Information Retrieval (IR)*. Dalam konteks data berupa kalimat, tiap data akan direpresentasikan sebagai suatu tas yang berisi semua kata-kata. Namun, nilai yang akan membedakan dari satu data dengan data lainnya adalah frekuensi kata yang muncul. Nilai ini hanya akan menghitung seberapa banyak kata-kata yang muncul dari suatu data, dan mengabaikan tata Bahasa dan urutan kata.

Namun, *bag of words* juga memiliki beberapa pengaturan yang dapat disesuaikan dengan kebutuhan. Pengaturan pertama adalah *n-gram*. *N-gram* merupakan suatu cara yang menunjukan seberapa banyak *sequence* sepanjang *n* yang muncul di seluruh data. Apabila *n* berkisar antara 1 dengan 2, maka akan dicari total frekuensi kata sepanjang 1 dan 2 yang ada di semua data. Tujuan dari cara ini adalah untuk mengatasi masalah *bag of words* yang mengabaikan urutan kata. Kadang, urutan dari kata pertama dan kata kedua apabila di acak tidak akan memiliki arti yang jelas. Namun dengan *n-gram,* urutan kata tersebut dapat disimpan dan akan diperoleh suatu arti baru dari *sequence* kata tersebut.

Pengaturan kedua yang dapat diubah adalah *min_df*. Pengaturan ini bertujuan untuk mengurangi total jumlah data. Caranya adalah dengan membuang data-data yang frekuensi kemunculannya ada dibawah nilai min_df. Hal itu berarti kata-kata yang berada di bawah frekuensi min_df merupakan sebuah kebetulan dan bukan suatu pola. Dengan begitu, probabilitas kata tersebut akan muncul lagi akan menjadi kecil, membuat data tersebut menjadi tidak penting / *outlier*.

2.1.6.3 *Term Frequency – Inverse Document Frequency*

Term Frequency – Inverse Document Frequency adalah suatu cara untuk mengukur derajat kepentingan suatu istilah terhadap sebuah kumpulan dokumen / korpus. Perhitungan ini seringkali digunakan untuk mencari fitur terpenting / fitur paling relevan dari sebuah kumpulan dokumen. Oleh karena itu, algoritma ini biasanya digunakan untuk pengolahan data-data yang berjumlah banyak. Karena perhitungan algoritma ini biasanya berdasarkan kumpulan kata kunci dan bukan sebuah kalimat, maka perlu dilakukan *text preprocessing* terlebih dahulu (*tokenization, stemming, stopwords removal*). Setelah kumpulan data telah berbentuk kumpulan kata dasar, maka baru bisa dilakukan perhitungan bobot *TF-IDF*. Perhitungan bobot

ini biasanya berdasarkan perhitungan total kata yang muncul pada dokumen dibanding dengan total kata pada korpus. Hasil akhir dari cara ini adalah kumpulan angka *float* dari tiap data. Rumus 2.1 hingga 2.3 diperoleh dari (Yunus, 2020).

Berikut adalah rumus untuk perhitungan *TF*:

$$tf_{ij} = \frac{f_d(i)}{\max_{j \in d} f_d(j)} \tag{2.1}$$

Keterangan:

- $f_d(i)$ = Frekuensi kemunculan *term i* pada dokumen *j*
- $\max_{j \in d} f_d(j)$ = Total *term* pada dokumen *j*

Berikut adalah rumus untuk perhitungan *IDF:*

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2.2}$$

Keterangan:

- N = Jumlah total dokumen dalam korpus
- $|\{d \in D : t \in d\}|$ = Jumlah dokumen yang mengandung *term t*

Berikut adalah rumus untuk perhitungan *TF-IDF:*

$$tfidf = tf_{ij} * idf(t, D) \tag{2.3}$$

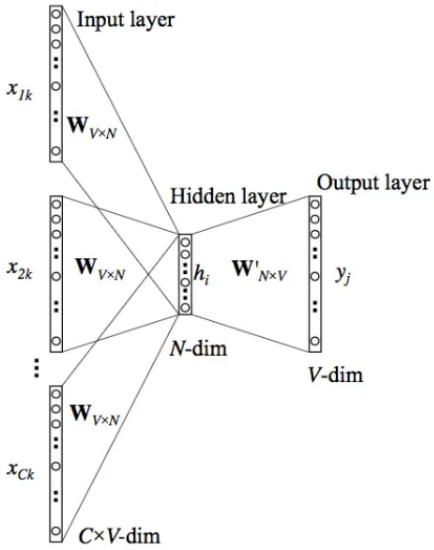
Keterangan:

- tf_{ij} = Frekuensi kemunculan *term i* pada dokumen *j*
- $idf(t, D)$ = Mengurangi bobot *term*

2.1.6.4 *Word2Vec*

Word2Vec merupakan salah satu cara *word embedding* untuk memperoleh nilai representasi vektor dari tiap kata dengan menggunakan *framework* dari *neural networks*. Model ini akan menghasilkan suatu *vector space* berisi tiap kata yang ada didalam *dataset*. Dari itu, akan dihitung nilai vektor dari tiap kata dengan dilihat jarak dari suatu kata dengan kata lainnya. Ada 2 arsitektur utama dari *word2vec*, yaitu *skip gram* dan *Continuous Bag of Words* (Mikolov

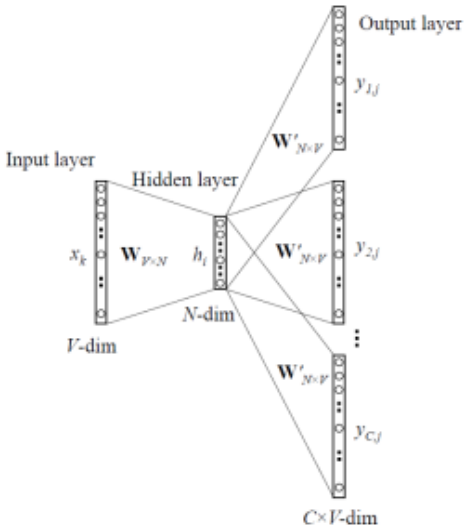
et al, 2013). Di kedua arsitektur tersebut, *word2vec* mempertimbangkan nilai suatu kata dengan melihat kata-kata yang mengelilingi kata tersebut.



Gambar 2.1 Arsitektur *Continuous Bag of Words*

Sumber: Goyal, M. (2023, Mar 24). *CBOW* Model.

Continuous Bag of Words menghitung nilai dari suatu kata berdasarkan kata-kata yang ada di sekelilingnya, dan urutan dari kata tidak akan mengubah hasil (seperti cara kerja *Bag of Words)*. Gambar 2.1 menunjukkan bahwa *input layer* nya berupa semua kata yang ada berada di sekitar suatu kata, dan *output layer* nya berupa suatu kata.



Gambar 2.2 Arsitektur *Skip Gram*

Sumber: Goyal, M. (2023, Mar 24). *Skip Gram* Model.

Skip Gram menggunakan suatu kata untuk menilai kata-kata yang berada di sekitar kata tersebut. Gambar 2.2 menunjukkan bahwa *input layer* merupakan suatu kata yang sedang diteliti dan *output layer* merupakan kata-kata yang berada di sekitar kata tersebut. Kata-kata yang berada lebih dekat akan diberikan nilai yang lebih tinggi daripada yang lebih jauh. Setelah model *word2vec* sudah jadi, kata-kata yang ada di *vector space* tadi akan ditempatkan sedemikian rupa sehingga kata-kata yang memiliki makna mirip akan berdekatan. Hasil akhir yang diperoleh merupakan representasi nilai vektor dari tiap kata.

Selain *word2vec*, terdapat juga suatu cara bernama *GloVe. GloVe* atau yang disebut *Global Vectors* merupakan suatu bentuk *unsupervised learning algorithm* untuk memperoleh representasi vektor dari kumpulan kata. Pembuatan model ini dilakukan dengan menggunakan data global dari suatu korpus (dalam konteks ini, korpus Bahasa Indonesia), yang nantinya akan digabungkan ke *dataset* yang akan diteliti. Dengan begitu, representasi nilai vektor dari tiap kata juga akan berbeda apabila hanya menggunakan dari *dataset*.

2.1.7 *Machine Learning*

Machine learning merupakan salah.0 satu pengaplikasian dari bidang *Artificial Intelligence* yang berfokus dalam pembelajaran mandiri dari suatu sistem. Pembelajaran mandiri ini memiliki arti bahwa manusia tidak berperan apa-apa saat sistem sedang melakukan pembelajaran. Manusia hanya memprogram sistem di awal dan nantinya sistem akan berjalan dan belajar sendiri. Sistem akan menggunakan kumpulan data atau biasa disebut sebagai *dataset* untuk belajar dengan mandiri. *Dataset* ini bisa memiliki banyak bentuk dan sistem akan diprogram sedemikian rupa oleh manusia untuk dapat menerima dan memproeses data tersebut. *Machine learning* telah digunakan dalam banyak bidang, seperti *data security,* bidang keuangan, bidang kesehatan, deteksi penipuan, penjualan, dan lainnya.

Ada 3 macam bentuk *machine learning* secara umum, yaitu *unsupervised, supervised,* dan *reinforcement learning* (Coursera, 2022).

2.1.7.1 *Unsupervised Learning*

Unsupervised learning memiliki arti bahwa sistem tidak akan mendapatkan bantuan sama sekali dari *user*. *Dataset* yang digunakan juga tidak memiliki *label* yang dapat membantu sistem. Oleh karena itu, sistem akan melakukan analisis dengan data seadanya dan memberikan kesimpulan dari data tersebut. *Unsupervised learning* cocok digunakan saat ingin melakukan pengumpulan fitur yang mirip dari data yang ada. Hal ini biasanya disebut sebagai *clustering.*

2.1.7.2 *Supervised Learning*

Supervised learning memiliki arti bahwa sistem akan diberikan data yang sudah memiliki *label* / arti. Dengan begitu, sistem sudah memiliki informasi terlebih dahulu yang dapat membantu dalam pembuatan model. *Outcome* yang dihasilkan nantinya juga akan memiliki *label* yang sama seperti *dataset* yang diberikan.

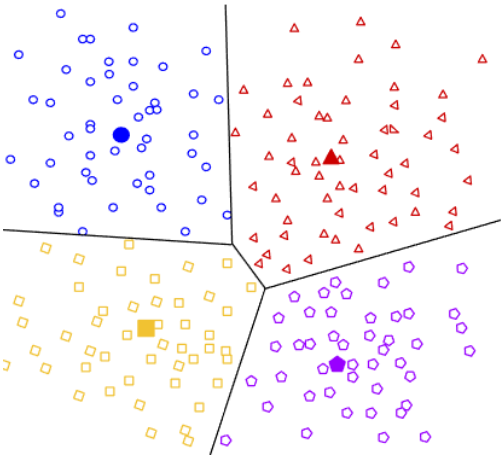
2.1.7.3 *Reinforcement Learning*

Reinforcement learning akan berjalan dengan *trial and error*. Sistem akan diberikan suatu hadiah saat *outcome* yang dihasilkan baik, dan hukuman saat *outcome* yang dihasilkan tidak baik. Dengan begitu, sistem nantinya akan belajar dengan mempertimbangkan hasil akhir yang diperoleh dan menyesuaikan cara kerjanya, seperti layaknya manusia belajar.

2.1.8 *Clustering*

Clustering merupakan metode yang digunakan untuk melakukan pengelompokkan data menjadi beberapa grup berdasarkan fitur-fitur yang mirip. *Clustering* merupakan salah satu metode dari *unsupervised learning.* Kelebihan dari metode ini adalah data yang diperlukan tidak perlu memiliki *label* untuk dapat dijalankan. Hasil yang diperoleh dari proses ini akan mempermudah sistem dalam melakukan penelitian kedepannya. Hal ini dikarenakan data yang digunakan untuk penelitian telah memiliki dimensi yang lebih kecil dengan fitur yang lebih mirip. *Clustering* biasanya digunakan untuk segmentasi pasar, sistem rekomendasi, pengenalan alur, dan banyak hal lainnya. Terdapat beberapa metode *clustering* umum*,* yaitu *centroid-based clustering, density-based clustering, distribution-based clustering,* dan *hierarchical clustering* (Xu & Tian, 2015).

2.1.8.1 *Centroid-based Clustering*

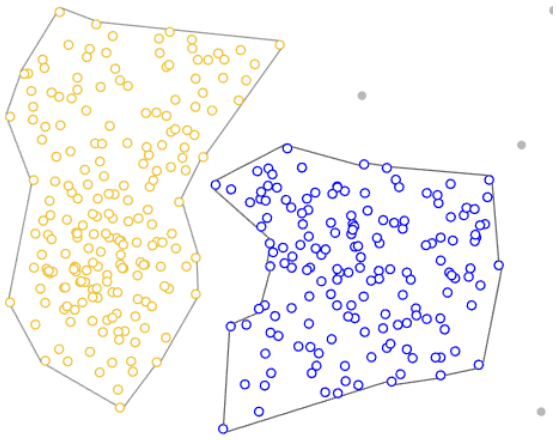


Gambar 2.3 *Centroid-based Clustering*

Sumber: Regunath, G. (2022, June 14). *Example of centroid-based clustering.*

Gambar 2.3 menunjukkan salah satu hasil dari *Centroid-based Clustering.* Metode ini melakukan *clustering* tanpa adanya susunan / hierarkis. Metode ini mencari titik-titik yang paling dekat dengan titik awal dan menganggap titik tersebut termasuk *cluster*nya. Hal ini dilakukan hingga semua fitur / titik telah memiliki *cluster*nya masing-masing. Kelemahan dari metode ini adalah sensitifitasnya terhadap letak titik awal dan *outlier*. Apabila penempatan titik awal tidak baik, maka hasil *clustering* bisa menjadi tidak baik.

2.1.8.2 *Density-based Clustering*

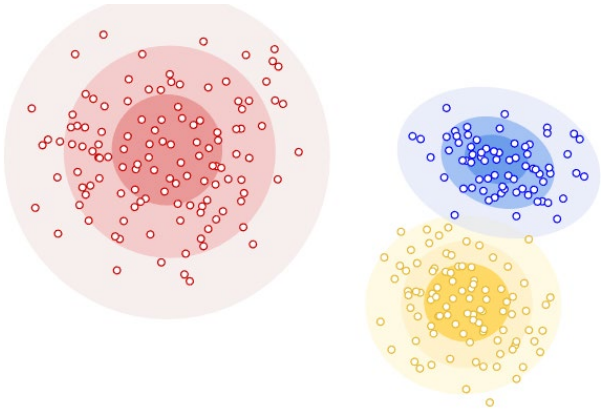


Gambar 2.4 *Density-based Clustering*

Sumber: Regunath, G. (2022, June 14). *Example of density-based clustering.*

Gambar 2.4 menunjukkan contoh hasil dari *Density-based Clustering.* Metode ini melihat densitas dari kumpulan titik / data dan memasukkannya kedalam suatu *cluster* apabila densitasnya berhubungan. Kekurangan dari metode ini adalah kesusahannya saat data memiliki berbagai macam densitas dan dimensi yang tinggi.

2.1.8.3 *Distribution-based Clustering*

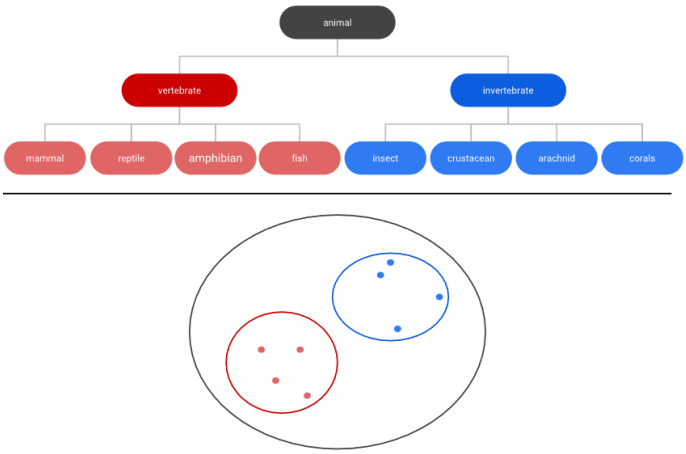


Gambar 2.5 *Distribution-based Clustering*

Sumber: Regunath, G. (2022, June 14). *Example of distribution-based clustering.*

Gambar 2.5 menunjukkan contoh hasil dari *Distribution-based Clustering.* Metode ini mengasumsikan data terdiri dari distribusi, layaknya distribusi *Gaussian*. Semakin jauh jarak titik dari pusat distribusi, probabilitas titik tersebut masuk didalam *cluster* tersebut mengecil.

2.1.8.4 *Hierarchical Clustering*



Gambar 2.6 *Hierarchical Clustering*

Sumber: Regunath, G. (2022, June 14). *Example of hierarchical tree clustering.*

Gambar 2.6 menunjukkan contoh dari hasil *hierarchical clustering.* Metode *clustering* ini akan menghasilkan suatu *cluster* berbentuk hierarki yang mirip seperti bentuk *tree*. Dengan

melihat jarak dan tingkatan, akan didapatkan *cluster* yang dapat divisualisasikan seperti sebuah *graph tree*.

2.1.9 *k-Means*

k-Means merupakan salah satu algoritma *clustering* yang menggunakan metode *centroid-based clustering*. Awalnya akan dipilih *centroids /* titik pusat secara random sebanyak k. k ini adalah parameter yang akan di *input* kan oleh *user*. Lalu akan dilakukan iterasi ke semua titik yang ada dan dilakukan perhitungan rata-rata ke *centroid*. Perhitungan ini akan dilakukan dengan menggunakan *Euclidean distance*. Hal ini dilakukan dengan tujuan untuk melakukan penyesuaian posisi *centroid-centroid* yang ada. Iterasi ini dilakukan hingga *centroid* tidak bergeser lagi. Saat *centroid* selesai bergerak, semua titik yang memiliki jarak terdekat dengan *centroid* A akan masuk ke dalam *cluster* A, demikian juga sisanya. Berikut adalah rumus untuk perhitungan *Euclidean distance*:

$$d_{Euclidean}(x, y) = \sqrt{\sum_{i} (x_i - y_i)^2} \tag{2.4}$$

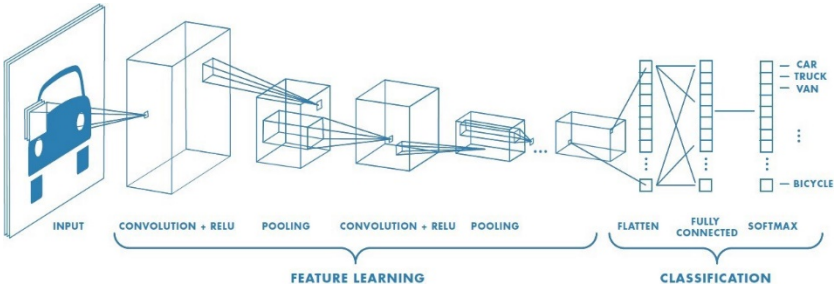
Keterangan:

- $d_{Euclidean}(x, y)$= Jarak suatu data ke *centroid*
- x_i = Posisi ke i suatu data
- y_i = Posisi ke i *centroid*

k-Means memiliki beberapa kelebihan daripada algoritma *clustering* lainnya, seperti kemudahan implementasi, kemampuan dalam meng-*handle* data yang banyak dan kompleks, dan mudah beradaptasi apabila terdapat data baru. Kekurangan dari algoritma ini adalah *user* harus memasukkan total *centroid* secara manual, hasil yang diperoleh sangat bergantung dengan kondisi *centroid* awal, diperlukan generalisasi apabila data memiliki ukuran dan densitas yang beragam, dan *outlier* dapat mengganggu *centroid* saat dihitung rata-ratanya.

2.1.10 *Deep Learning*

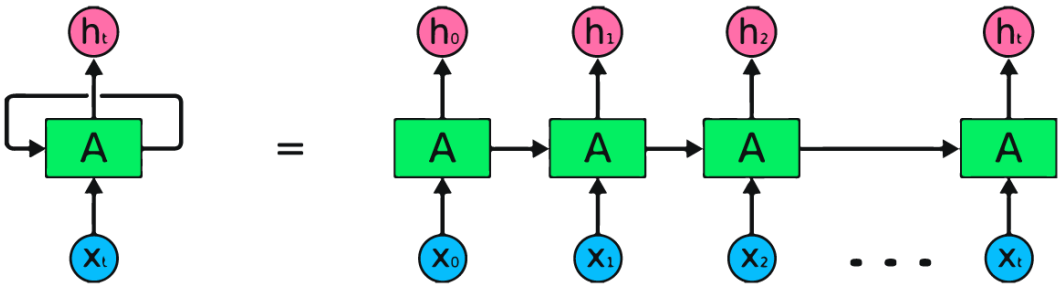
Deep learning merupakan salah satu cabang dari *Artificial Intelligence. Deep learning* juga merupakan salah satu cabang dari *machine learning* yang lebih berfokus kepada cara komputer mengimitasi cara manusia belajar. Tipe ini berfokus kepada penggunaan suatu sistem yang mirip seperti jaringan syaraf manusia. Sistem ini biasa disebut dengan *neural network.* Disini, digunakan beberapa *layer* untuk dilakukan *processing* data untuk memperoleh informasi dari data secara lebih dalam. Kegunaan *deep learning* ini biasanya digunakan untuk *semantic parsing, transfer learning, natural language processing, computer vision,* dan lain-lain (Guo et al., 2016).



Gambar 2.7 *Convolutional Neural Network*

Sumber: Saha, S. (2018, Dec 15). *Convolutional Neural Network.*

Metode *deep learning* umum biasanya menggunakan *Convolutional Neural Network* dan *Recurrent Neural Network*. *Convolutional Neural Network* sangat umum digunakan untuk melakukan *processing* data-data berupa gambar. *CNN* menggunakan proses konvolusi untuk mengambil data-data berupa *pixel* dari gambar untuk diproses selanjutnya kedalam *neural network*. Pada gambar 2.7 dapat terlihat bahwa *pixel-pixel* yang diperoleh akan dimasukkan dalam tahap konvolusi dan *pooling* untuk memperoleh fitur-fiturnya. Setelah itu, akan dilakukan klasifikasi berdasarkan fitur-fitur yang telah diperoleh tadi.



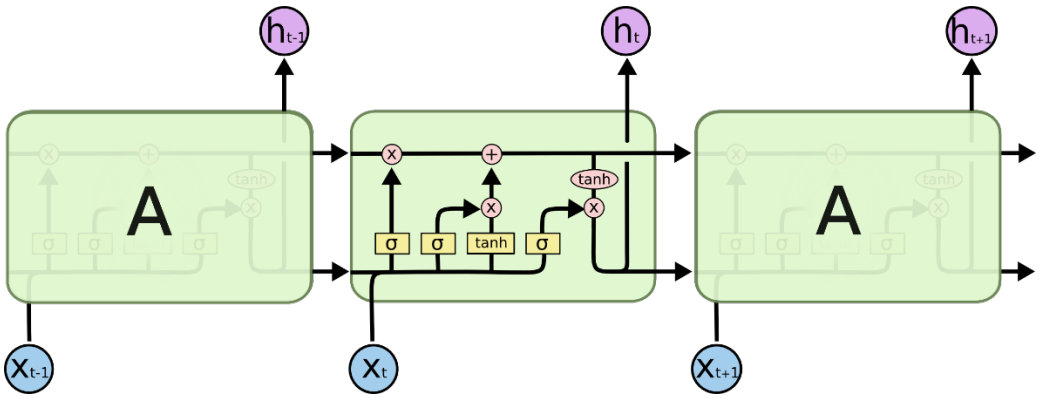
Gambar 2.8 *Recurrent Neural Network*

Sumber: Awan, A. (2022, Mar). *RNN.*

RNN digunakan untuk data yang memerlukan proses *sequential*, seperti *text* dan suara. Cara kerja dari algoritma ini adalah menyimpan informasi sebelumnya dan dilanjutkan ke proses selanjutnya. Gambar 2.8 menunjukkan bahwa data-data yang diperoleh dari *input layer* akan dilanjutkan ke *hidden layer*. Di *hidden layer,* akan dilakukan proses pengambilan informasi yang saling berhubungan / *recurrent*. Hasil informasi yang diperoleh di *layer* pertama *hidden layer*, akan dilanjutkan ke *layer* 2 untuk diperoleh informasi selanjutnya. Hal ini dilakukan berulang-ulang hingga data yang diproses telah habis. Namun, *RNN* memiliki suatu masalah apabila data yang diperoleh merupakan data *sequential* yang panjang, yaitu masalah *vanishing gradient.* Masalah ini terjadi saat *RNN* gagal dalam mengingat informasi-informasi yang diperoleh dari beberapa *step* sebelumnya (*long term dependency*). Masalah ini dapat mengakibatkan penurunan akurasi dari suatu prediksi *RNN*. Untuk menjawab masalah ini, akan digunakan salah satu jenis *RNN* bernama *Long Short Term Memory*.

2.1.11 *Bidirectional Long Short Term Memory*

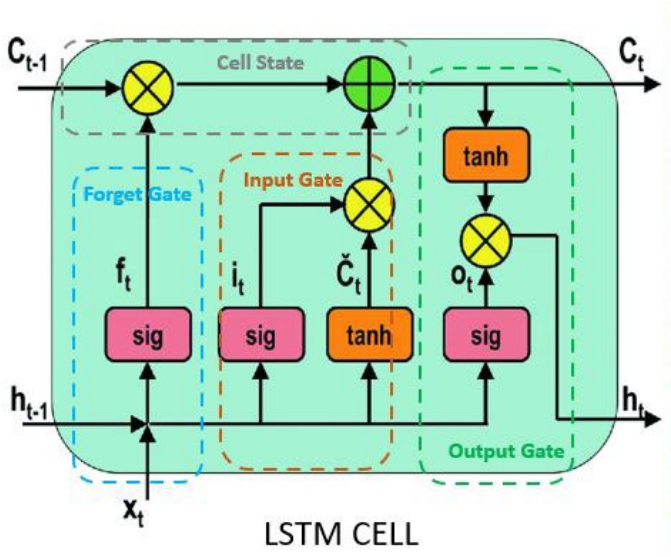
Long Short Term Memory merupakan salah satu jenis *Recurrent Neural Network* yang berfokus kepada masalah dimana arti kata sebelum-sebelumnya perlu disimpan untuk memprediksi selanjutnya (Knight, 2021). *Long Short Term Memory* memiliki suatu fitur bernama *Cell state* yang akan menyimpan arti kata awal yang mungkin telah dilupakan apabila menggunakan algoritma lain. Kata-kata yang disimpan tadi merupakan kata-kata yang memiliki makna penting, tidak hanya kata random yang ada di suatu kalimat yang sedang diteliti. Dengan begitu, walaupun kalimat yang sedang diteliti merupakan kalimat panjang, program masih tetap dapat menyimpan dan mengartikan arti sesungguhnya dari kalimat tersebut.



Gambar 2.9 *Long Short Term Memory*

Sumber: Olah, C. (2015, Aug 27). *Understanding LSTM Networks.*

Gambar 2.9 menunjukkan ilustrasi dari cara kerja *Long Short Term Memory.* Terdapat 2 *activation function* yang digunakan, yaitu *sigmoid* (σ) dan *tanh*. Fungsi aktivasi *sigmoid* bertugas untuk mentransformasikan nilai data yang awalnya memiliki *range* bebas menjadi 0 hingga 1. Tujuan dari aktivasi ini adalah untuk menghindari hilangnya data apabila bernilai 0 dan menormalisir nilai data. Fungsi dari aktivasi *tanh* adalah mengubah data menjadi range -1 hingga 1. Fungsi dari aktivasi ini sama seperti *sigmoid*, yaitu untuk menormalisir nilai data. Hal ini mencegah nilai data yang terlalu tinggi setelah melewati banyak transformasi operasi matematika.



Gambar 2.10 Tahapan *LSTM*

Sumber: Singhal, G. (2020, Sep 9). *Introduction to LSTM Units in RNN.*

Gambar 2.10 menunjukkan 3 *gate* yang dimiliki *Long Short Term Memory*. *Gate* pertama adalah *forget gate*. Dalam tahapan ini, informasi-informasi yang tidak terlalu penting akan

dihilangkan menggunakan fungsi aktivasi *sigmoid*. X_t merupakan *input* data pada *timestep* ke t. h_{t-1} merupakan *hidden state* yang diperoleh dari *timestep* sebelumnya (t-1). Berikut adalah rumus yang digunakan di *forget gate*:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_i) \quad (2.5)$$

Keterangan:

- x_t = *input* data di *timestep t*
- h_{t-1} = *hidden state* di *timestep t-1*
- b_i = *weight input*
- W_f = *weight matrix hidden state*

Tahap kedua adalah *input gate*, dimana proses ini akan memilah dan menentukan informasi mana yang akan diperbarui ke *cell state.* Proses ini akan melewati fungsi aktivasi *sigmoid* dan *tanh*. Rumus *sigmoid* akan digunakan lagi untuk perhitungan. Perhitungan lainnya adalah rumus *tanh*, dimana akan membentuk vektor baru yang akan ditambahkan ke *cell state*. Berikut adalah rumus yang digunakan di *input gate*.

Berikut merupakan rumus *sigmoid:*

$$i_t = \sigma(W_f * [h_{t-1}, x_t] + b_i) \quad (2.6)$$

Keterangan:

- x_t = *input* data di *timestep t*
- h_{t-1} = *hidden state* di *timestep t-1*
- b_i = *weight input*
- W_f = *weight matrix hidden state*

Berikut merupakan rumus *tanh:*

$$\tilde{c}_t = \tanh(W_c * [h_{t-1}, x_t] + b_c) \quad (2.7)$$

Keterangan:

- x_t = *input* data di *timestep t*
- h_{t-1} = *hidden state* di *timestep t-1*
- b_c = *weight input*
- W_c = *weight matrix hidden state*

Tahap ketiga adalah meng-*update cell state* sebelumnya dengan hasil *forget gate* dan *input gate* yang telah diperoleh. Berikut merupakan rumus untuk meng-*update cell state*:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \quad (2.8)$$

Keterangan:

- f_t = Nilai dari *forget gate*
- c_{t-1} = Nilai *cell state* sebelumnya
- i_t = Nilai dari *input gate sigmoid*
- $\tilde{c}_t$ = Nilai dari *input gate tanh*

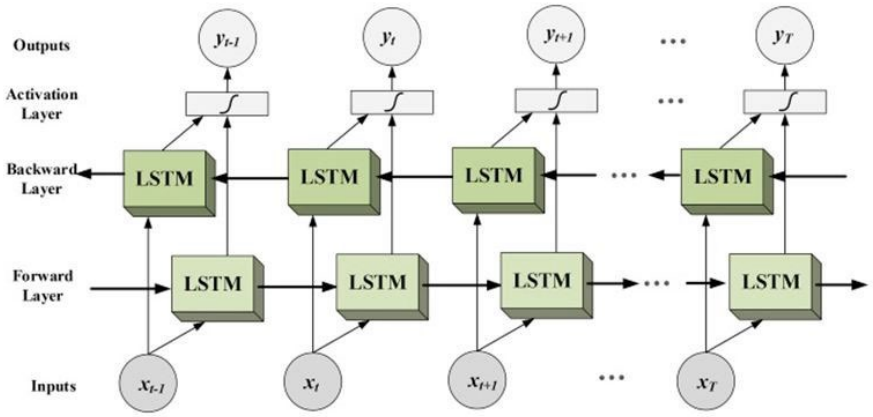
Tahap terakhir adalah *output gate*. *Hidden state* akan dilakukan fungsi *sigmoid*, dan *cell state* yang telah ter-*update* tadi akan dilakukan fungsi *tanh*. Setelah 2 hasil *output* diperoleh, maka akan dilakukan perkalian sebelum lanjut ke *layer* berikutnya. Berikut merupakan rumus perkalian di *output gate*:

$$h_t = o_c * \tanh(c_t) \quad (2.9)$$

Keterangan:

- o_c = Hasil output dari *hidden layer* yang telah dilakukan aktivasi *sigmoid*
- c_t = Nilai *cell state* yang telah di-*update*
- h_t = Nilai *hidden state* yang akan dilanjutkan ke *layer* berikutnya

Namun, *input* pertanyaan yang diberikan oleh *user* kadang tidak urut atau kacau dalam susunan kata. Hal ini dapat membuat bingung algoritma karena pembacaan kalimat dimulai dari kiri dan bergeser ke kanan. Dengan penggunaan algoritma *Bidirectional Long Short Term Memory*, susunan kata yang kacau tadi tetap dapat diprediksi artinya secara akurat. Beda *Bi-LSTM* dengan *LSTM* adalah *Bi-LSTM* menggunakan 2 model *LSTM* dengan arah yang berbeda. *LSTM* pertama mengecek kalimat dari kiri ke kanan dan *LSTM* kedua mengecek kalimat dari kanan ke kiri.



Gambar 2.11 *Bidirectional Long Short Term Memory*

Sumber: Verma, Y. (2021, Jul 17). *Complete Guide To Bidirectional LSTM.*

Gambar 2.11 menunjukkan ilustrasi cara kerja *Bi-LSTM*. Terdapat 2 arah *LSTM,* yaitu *Forward Layer* dan *Backward Layer.* Hasil yang diperoleh dari kedua *LSTM* tadi akan dirata-rata dan dimasukkan ke dalam *activation layer.* Setelah itu, baru akan diperoleh hasil akhir.

2.2 Tinjauan Studi

2.2.1 *Chatbot* untuk *Website* UK Petra dengan *Hidden Markov Model* dan *k-Nearest Neighbor* (Kevin, 2021)

- Masalah yang diangkat di penelitian ini adalah belum adanya *chatbot* yang disediakan untuk *website* UK Petra. Tujuan dari penelitian ini adalah untuk menghasilkan suatu *chatbot* yang dapat membantu calon mahasiswa / mahasiswa / orang tua mahasiswa / masyarakat umum dalam memperoleh informasi mengenai UK Petra.
- Metode yang diusulkan dari penelitian ini adalah menggunakan *k-Nearest Neighbor* untuk mendeteksi *intent* / label dari suatu pertanyaan. *Hidden Markov Model* akan digunakan untuk merangkai jawaban baru berdasarkan *dataset* yang ada.
- Hasil dari penelitian ini menunjukan bahwa *chatbot* dengan metode *k-Nearest Neigbor* berhasil memberikan jawaban yang cukup akurat untuk pertanyaan pendek (2-3 kata) dan sedang (4-7 kata). Namun akurasi *chatbot* turun drastis saat diberikan pertanyaan yang panjang (lebih dari 8 kata). Metode *Hidden Markov Model* memberikan jawaban yang kacau dan tidak dapat dimengerti.
- Perbedaan penelitian yang dilakukan dengan skripsi ini adalah metode yang digunakan di penelitian ini adalah *k-Nearest Neighbor* untuk mendeteksi label pertanyaan. Di skripsi ini akan menggunakan metode *k-Means* untuk melakukan *clustering* pertanyaan-

pertanyaan yang memiliki arti mirip. *Bidirectional Long Short Term Memory* akan digunakan untuk pemberian jawaban berdasarkan pertanyaan yang sudah di *cluster* tadi. Penggunaan *k-Means* akan mengurangi probabilitas jawaban yang tidak sesuai, sedangkan *Bi-LSTM* akan mampu memprediksi jawaban dari pertanyaan yang kompleks.

2.2.2 *A New Chatbot for Customer Service on Social Media* (Xu et al, 2018)

- Masalah yang diangkat dari penelitian ini adalah kurang tersedia nya *chatbot* dari *social media* yang didedikasikan untuk menjawab pertanyaan emosional dan informasional dari *user.* Hal ini karena kecenderungan *user* untuk meminta bantuan / jawaban melalui *social media* daripada mencari di internet / menelpon ahli. Tujuan dari penelitian ini adalah untuk menghasilkan suatu *chatbot social media Twitter* yang dapat memberikan jawaban akurat dari pertanyaan *user* yang bersifat emosional maupun informasional.
- Metode yang diusulkan dari penelitian ini adalah metode *Long Short Term Memory* untuk memberikan respons kepada *user*. 2 *Long Short Term Memory* digunakan sebagai *encoder* dan *decoder* yang nantinya berujung kepada pemberian jawaban kepada *user.*
- Hasil dari penelitian ini adalah *chatbot* berhasil memberikan jawaban yang akurat ke pertanyaan yang bersifat emosional. Namun akurasi *chatbot* turun saat diberikan pertanyaan yang bersifat informasional. Hal ini disebabkan oleh perlunya konteks pertanyaan yang lebih banyak dan jelas agar *chatbot* dapat memberikan jawaban yang akurat.
- Perbedaan dari penelitian yang dilakukan dengan skripsi ini adalah metode yang digunakan di penelitian ini adalah *Long Short Term Memory* untuk memberikan respons jawaban. Di skripsi ini akan menggunakan metode *k-Means* untuk melakukan *clustering* pertanyaan-pertanyaan yang memiliki arti mirip. *Bidirectional Long Short Term Memory* akan digunakan untuk pemberian jawaban berdasarkan pertanyaan yang sudah di *cluster* tadi. Penggunaan *Bi-LSTM* ini akan mampu menjawab pertanyaan-pertanyaan kompleks, yang mungkin gagal dilakukan oleh algoritma *Long Short Term Memory* saat diberikan pertanyaan informasional.

2.2.3 *Automated Thai-FAQ Chatbot using RNN-LSTM* (Muangkammuen et al, 2018)

- Masalah yang diangkat dari penelitian ini adalah semakin diperlukannya suatu *service* yang baik dalam *handling customer* ditengah perkembangan *e-commerce* yang pesat.

Seringkali, *customer* memerlukan bantuan saat mereka ingin membeli sesuatu atau hanya ingin memperoleh informasi. Oleh karena itu, akan digunakan *chatbot* untuk meng *handle* permasalahan tersebut. *Dataset* yang akan digunakan oleh *chatbot* adalah *dataset* pertanyaan-pertanyaan *customer support* berbahasa Thailand sebanyak 2636 pertanyaan dan jawaban.

- Metode yang akan digunakan di penelitian ini adalah algoritma *Long Short Term Memory*. *Dataset* telah dilakukan kategorisasi sebanyak 80 kelas secara manual berdasarkan tipe / kemiripan pertanyaan. Algoritma ini akan digunakan untuk mengklasifikasi jawaban yang benar dari suatu pertanyaan.
- Hasil yang diperoleh dari penelitian ini adalah *chatbot* mampu memproses sebanyak 86.36% pertanyaan dan memperoleh akurasi sebanyak 93.2%.
- Perbedaan dari penelitian yang dilakukan dengan skripsi ini adalah penelitian menggunakan *LSTM* untuk mengklasifikasikan jawaban. Selain itu, penelitian ini melakukan *clustering* 80 kelas secara manual. Skripsi ini menggunakan *Bidirectional Long Short Term Memory* dengan tujuan untuk menjawab pertanyaan yang strukturnya tidak baik dan panjang. Di penelitian ini, tidak ditunjukan apakah terdapat pertanyaan yang kompleks atau tidak. Skripsi ini juga akan menggunakan *k-Means* untuk *clustering* pertanyaan yang mirip secara otomatis, tidak secara manual seperti yang dilakukan penelitian ini.

2.2.4 *LSTM and Simple RNN Comparison in the Problem of Sequence to Sequence on Conversation Data Using Bahasa Indonesia* (Prabowo et al, 2018)

- Masalah yang diangkat dari penelitian ini adalah perbandingan kemampuan *chatbot* saat menggunakan algoritma *Long Short Term Memory* dan algoritma *Recurrent Neural Network* sederhana. Permasalahan ini akan dites dengan menggunakan *dataset* berbahasa Indonesia. *Dataset* ini merupakan kumpulan pembicaraan dari *customer* dengan *customer service* yang berfokus kepada bidang bisnis transportasi, seperti membeli tiket untuk kereta api dan pesawat. Total dari *training* data sebanyak 50 data pembicaraan.
- Metode yang digunakan di penelitian ini ada 2, yaitu *LSTM* dan *RNN* sederhana. 2 metode ini nantinya akan digunakan sebagai perbandingan kemampuan dan akurasi *chatbot*.

- Hasil dari penelitian ini adalah *LSTM* lebih mampu memberikan jawaban yang lebih akurat dibanding *RNN* sederhana. Masalah lainnya yang bisa menjadi penyebab hasil akhir ini adalah data yang terlalu sedikit.
- Perbedaan dari penelitian yang dilakukan dengan skripsi ini adalah penelitian ini menggunakan *LSTM* dan *RNN* sederhana sebagai *testing* akurasi *chatbot*. Di skripsi ini akan menggunakan metode *k-Means* untuk melakukan *clustering* pertanyaan-pertanyaan yang memiliki arti mirip. *Bidirectional Long Short Term Memory* akan digunakan untuk pemberian jawaban berdasarkan pertanyaan yang sudah di *cluster* tadi. Permasalahan kekurangan data *training* yang dialami di penelitian yang dilakukan juga teratasi di skripsi ini. Skripsi ini akan menggunakan *dataset* sebanyak 1088 pertanyaan dan jawaban.

2.2.5 Implementasi *Natural Language Processing* pada Sistem *Chatbot* Informasi Saham dengan Algoritma *Long Short-Term Memory (LSTM)* dan *Fuzzy String Matching* (Azni, 2022)

- Masalah yang diangkat dari penelitian ini adalah *investor* saham kesusahan dalam memperoleh informasi-informasi mengenai saham. *Investor* saham kadang memakan banyak waktu untuk harus masuk ke berbagai situs dan *searching* di internet untuk memperoleh informasi saham yang diperlukan. Tujuan dari penelitian ini adalah untuk membuat suatu *chatbot* yang dapat memberikan informasi mengenai pertanyaan saham-saham pemula serta memberikan harga saham secara *real time.*
- Metode yang digunakan di penelitian ini adalah *Long Short-Term Memory* untuk memprediksi label pertanyaan secara akurat dan tepat. *Fuzzy String Matching* akan digunakan untuk menampilkan *response* jawaban yang tepat.
- Hasil dari penelitian ini adalah *chatbot* berhasil dalam memberikan jawaban yang akurat mengenai pertanyaan saham pemula yang *simple* dan sederhana.
- Perbedaan dari penelitian yang dilakukan dengan skripsi ini adalah metode yang digunakan di penelitian ini adalah *Long Short Term Memory* untuk memprediksi *label* pertanyaan dan *Fuzzy String Matching* untuk memberikan jawaban. Di skripsi ini akan menggunakan metode *k-Means* untuk melakukan *clustering* pertanyaan-pertanyaan yang memiliki arti mirip. Pengelompokan ini diharapkan dapat membuat *chatbot* lebih akurat saat ingin menjawab pertanyaan di kelompok tertentu. *Bidirectional Long Short Term Memory* akan digunakan untuk pemberian jawaban berdasarkan pertanyaan yang

sudah di *cluster* tadi. Penggunaan *Bi-LSTM* ini akan lebih mampu menjawab pertanyaan yang lebih kompleks, dimana di penelitian ini tidak ditunjukan.