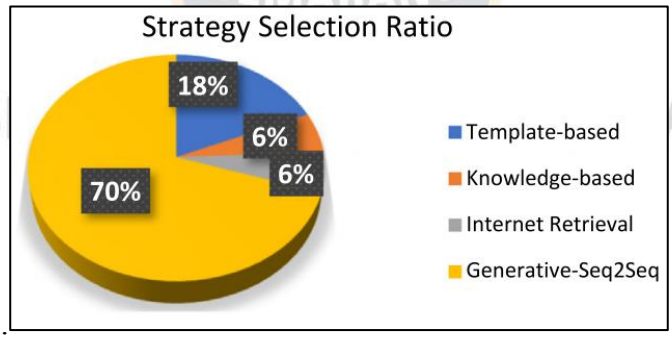
BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Chatbot memainkan peran sebagai penghubung interaksi manusia dan komputer yang menyimulasikan percakapan tertulis dengan tujuan memberi kesan sementara bahwa seseorang sedang berbicara dengan manusia tersebut (Sheikh dkk, 2019). Banyak penelitian yang telah mengembangkan *chatbot* untuk berbagai bidang keperluan. Nuruzzaman dan Hussain (2020) merancang *Dialog-based Chatbot* untuk industri asuransi yang mereka beri nama IntelliBot. Penelitian ini berfokus pada pemberian respon yang bermakna dan menarik bagi pengguna dalam domain yang spesifik, yaitu industri film dan asuransi. Dalam pengujiannya, IntelliBot dibandingkan dengan beberapa *existing chatbot,* yaitu RootyAI, ChatterBot dan DeepQA dengan menggunakan dataset *Cornell Movie Dialogue, Insurance Domain dan Insurance Terms & Keywords.* Metodologi yang digunakan dalam membangun chatbot ini adalah menggunakan empat pendekatan yaitu *template-based query, knowledge-based query, internet retrieval* dan *generative seq-to-seq.* Komposisi strategi seleksi yang digunakan IntelliBot dapat dilihat pada gambar 2.1.



Gambar 2.1 Strategi pemilihan respon pada IntelliBot

Metode *sequence to sequence* mendapat tugas dengan paling banyak pada model ini hingga mencapai 70%. Hasil yang didapat dalam implementasi ini adalah Intellibot secara kuantitas memiliki tingkat presisi yang tinggi atas respon yang diberikan dengan *F1 Score* 98,57% dibandingkan dengan *DeepQA* 82,64%,

RootyAI 47,31 % dan *ChatterBot* 43,96%. Hal ini membuktikan bahwa metode *sequence-to-sequence* dapat menghasilkan respon yang lebih akurat dan bersifat *generative* sehingga lebih mendekati percakapan manusia.

Teckchandani dkk (2019) membandingkan metode *query* berdasarkan *keywords* yaitu *Artificial Intelligence Markup Language* (AIML) dengan metode *sequence-to-sequence* berdasarkan tiga indikator pembanding, kepuasan pengguna, nilai *information retrieval* dan ketercapaian tujuan percakapan (*task completion*). Perbandingan ini menggunakan data *knowledge base* universitas Middlesex Mauritius mengenai penerimaan mahasiswa baru, aspek finansial, program pendidikan, akomodasi dan keterangan umum. Perbandingan dilakukan dengan *training* kedua model kemudian diuji dan dihitung validitas responnya dengan melibatkan 45 orang responden. Pengujian dengan indikator pertama yaitu kepuasan pengguna menghasilkan nilai 6,73 untuk AIML dan 5,52 untuk *sequence-to-sequence.* Indikator *information retrieval* menghasilkan skor 0,877 untuk AIML dan 0,649 untuk *sequence-to-sequence. Task completion* sebagai indikator pembanding terakhir menghasilkan 67,9% untuk AIML dan 63,8% untuk *sequence-to-sequence.* Berdasarkan hasil tersebut, Teckchandani dkk (2019) menyimpulkan bahwa AIML lebih baik terutama dalam kepuasan pengguna dan *information retrieval.* Hal ini dikarenakan respon *chatbot* dengan metode *sequence-to-sequence* yang bersifat *generative* membuat kalimat yang dihasilkan tidak terstruktur (*grammar problem*) dan cenderung menghasilkan perulangan kata yang sering digunakan (Teckcandhani dkk, 2019).

Aelani dan Gustaman (2021) mengembangkan *chatbot* untuk layanan informasi penerimaan peserta didik baru pada instansi kampus menggunakan *Multinomial Naïve Bayes (MNB)* dan *TF-IDF Weighting.* Proses dimulai dari *preprocessing,* pengecekan skor *TF-IDF Weighting,* klasifikasi menggunakan MNB kemudian menghasilkan respon berdasarkan hasil klasifikasi *intent. Preprocessing* yang digunakan adalah *casefolding*, *filtering*, *tokenization*, *stop word removal* dan *stemming.* Data yang digunakan adalah 1.330 pertanyaan berbahasa Indonesia yang terbagi dalam 17 *intents*. Berdasarkan hasil pengukuran pada *data testing* dengan metode *K-fold Cross Validation* didapat akurasi mencapai

99,3%. Kemudian dilakukan tes eksternal yang melibatkan 31 pengguna dengan 170 pertanyaan, didapat hasil akurasi sebesar 65%. Berdasarkan hasil tersebut, metode *query* pemilihan respon dengan menggabungkan *TF-IDF Weighting* dan MNB dapat menghasilkan akurasi yang cukup baik (diatas 90%).

Neural Responding Maching dikembangkan oleh peneliti dari *Noah's Ark Lab Huawei Technologies* (Shang dkk, 2015) untuk percakapan *short-text based* menggunakan konsep *encoder-decoder*. Data yang digunakan adalah data percakapan pada aplikasi Weibo, yaitu aplikasi *microblogging* seperti Twitter yang ada di China. Aplikasi Weibo memiliki batasan karakter dengan jumlah 140 *chinese* karakter baik pada *post* maupun *reply*, sehingga membuatnya ideal untuk digunakan sebagai data *short text conversation*. Metode yang digunakan dalam *encoder* dan *decoder* adalah *Recurrent Neural Network* karena memiliki keunggulan dalam memproduksi rangkaian kata dengan jumlah tertentu secara natural. Pengukuran secara otomatis seperti *BLEU Score* masih menjadi masalah pada pengaplikasiannya karena kemungkinan respon yang dihasilkan masih terlalu luas sehingga tidak dapat memberikan referensi pengukuran yang memadai. Metode evaluasi yang digunakan kemudian adalah penilai anotasi, yaitu lima orang dengan pengalaman minimal 3 tahun menggunakan Weibo diundang untuk diinstruksikan untuk membayangkan merespon pada kalimat yang dites. Tes yang dilakukan pada 110 *post* dengan panjang 6 sampai 22 karakter dan rata-rata 12,5 kata, menghasilkan nilai *kappa measure* 0,397. Lebih kecil dibandingkan dengan *competitor method* yaitu *retreival based* dengan 0,346 dan *statistical machine translation based* dengan 0,448.

Aziz (2015) meneliti implementasi *vector space* dalam pembangkitan *Frequently Asked Questions* (FAQ) secara otomatis sebagai solusi untuk menangani keluhan pelanggan. Metode yang digunakan adalah *Term Frequency – Inverse Document Frequency* (TF-IDF) dan *Vector Space Model* (VSM). Data keluhan pelanggan pada UPT Puskom UNS sejumlah 190 data digunakan sebagai objek penelitian. Pembagian *data training* dan *testing* dilakukan secara terurut dalam empat percobaan yaitu mulai 60%:10% sampai 90%:10%. Dalam eksperimen yang dilakukan, satu data keluhan digunakan sebagai *sample* untuk

kemudian dihitung skor *similarity*-nya dengan sejumlah data yang memiliki kemiripan. Pengujian dimulai dengan mencari nilai *threshold* yang tepat untuk digunakan dalam pencocokan, yaitu 0,5 dan 0,65 sebagai nilai awal untuk dilakukan percobaan. Hasil yang didapat adalah penggunaan nilai *threshold* 0,5 mendapatkan akurasi 62,09% dan pencocokan dengan nilai *threshold* 0,65 mendapatkan akurasi 83,18%. Berdasarkan hasil tersebut didapat bahwa nilai *threshold* 0,65 dapat memberikan hasil *similarity* yang lebih relevan dalam mengukur tingkat kesamaan dokumen khususnya pada data keluhan pelanggan.

Hasani dkk (2022) meneliti penerapan *deep learning* dengan pendekatan probabilitas *threshold* untuk mendeteksi *intent* "*out-of-scope*" (OOS) pada *task based chatbot*. Penelitian ini menggunakan model BiLSTM dengan eksperimen pada berbagai nilai *threshold*, yaitu 0,4, 0,5, 0,6, dan 0,7, untuk mengidentifikasi intent yang tidak sesuai dengan cakupan *chatbot*. Hasil eksperimen menunjukkan bahwa *threshold* 0.7 memberikan performa *recall* terbaik, mencapai tingkat optimal dalam mendeteksi *intent* OOS tanpa mengurangi relevansi. Studi ini menekankan pentingnya penentuan *threshold* yang sesuai untuk menjaga keseimbangan antara *recall* dan *precision*, sehingga *chatbot* dapat memberikan respons yang lebih akurat dalam skenario yang beragam.

Zhang dkk (2020) mengembangkan metode desain *chatbot* menggunakan pendekatan *hybrid retrieval* dan *generate* model berdasarkan data telemarketing. Penelitian ini memanfaatkan model *Vector Space Model* (VSM) yang dikombinasikan dengan pembobotan TF-IDF untuk meningkatkan akurasi dalam pencocokan *query* pengguna dengan respons yang tersedia. Nilai *threshold similarity* sebesar 0,8 digunakan untuk menentukan apakah hasil dari *retrieval* model cukup relevan atau perlu diproses lebih lanjut oleh *generate* model berbasis LSTM. Eksperimen menunjukkan bahwa pendekatan *hybrid* ini mampu meningkatkan akurasi respons *chatbot*, sekaligus mempertahankan efisiensi dalam pengelolaan sumber daya komputasi. Hal ini membuktikan pentingnya *threshold* sebagai parameter kritis dalam optimasi sistem *chatbot*.

Pengembangan *generative based chatbot* dilakukan Zalake N. dan Naik G. (2019) menggunakan *Deep Recurrent Neural Network* (RNN). *Dataset movie*

dialogue corpus dan *Ubuntu's technical support channels* digunakan sebagai data uji dalam penelitian dengan jumlah *sequence* mencapai 25.000 *sequences* dan 7002 *vocabularies*. Dua model arsitektur *Deep* RNN yaitu *Long Short Term Memory* (LSTM) dan *Bidirectional LSTM cell* digunakan untuk membangun model *chatbot* dengan menambahkan *attention mechanism,* sehingga chatbot dapat mengingat vektor masukan sebagai konteks hingga 30 kata dalam satu kalimat. *Training* model dengan dataset yang digunakan mencapai waktu 48-72 jam dengan memanfaatkan 2GPU dan ukuran memori 50GB. Pengujian dilakukan dengan menghitung nilai akurasi dan *BLEU Score.* Akurasi yang didapat dari implementasi data *ubuntu chat corpus* dan *movie dialogue corpus* secara berurutan adalah 85,16% dan 93,64%. *Word embeddings, Beam Search* dan *Named Entity Recognition* menjadi catatan dalam pengembangan model *chatbot Deep RNN* sehingga dapat digunakan secara masal (Zalake dan Naik, 2019).

Jurgita K.D (2020) mengembangkan *Generative Chatbot* yang di-*train* menggunakan data terbatas, seperti yang tertera dalam judulnya yaitu *A Domain Specific Generative Chatbot Trained from Little Data*. Data yang digunakan merupakan data yang dibuat secara manual dengan 567 pasang pertanyaaan dan jawaban mengenai informasi perusahaan Tilde seperti informasi produk, harga, layanan dan teknologi yang digunakan. *Data Preprocessing* diimplementasikan pada *dataset* sebelum menjadi *data training* untuk model, yaitu *punctuation removal* dan *punctuation separation*. Kedua *data preprocessing* tersebut diimplementasikan agar *chatbot* dapat menghasilkan respon dengan tanda baca sekaligus, karena dataset tersedia dalam dua bahasa yaitu Bahasa Inggris dan Bahasa Lithuania. Model LSTM, *stacked LSTM* dan *BiLSTM* (*Bidirectional* LSTM) diimplementasikan dalam penelitian dengan *punctuation treatment* diatas. Hasilnya adalah pengujian dengan data Bahasa Inggris dan Bahasa Lithuania mendapat *BLEU Score* 0,513 dan 0,505 untuk perlakuan *punctuation removal* sedangkan untuk *punctuation separation* mendapat skor 0,488 dan 0,439.

Yogish dkk (2019) membandingkan lima metode *Information Retrieval* untuk mengetahui kemampuan metode dalam *query* sebuah informasi spesifik dalam dokumen. Pertama *Term Frequency* (TF) yaitu metode *query* hanya

berdasarkan parameter lokal, frekuensi kemunculan kata t dalam dokumen d. Ke-dua *Classical TF-IDF,* yaitu query dengan parameter lokal (*term frequency*) dan global (*inverse document frequency*). Ke-tiga *Normalized TF-IDF,* yaitu TF-IDF yang sudah dinormalisasi dengan tujuan mengurangi ketidakseimbangan untuk dokumen yang lebih panjang. Metode ke-empat yaitu *Sub-linear Normalized TF-IDF,* menambahkan log untuk mengurangi tingkat kepentingan sebuah kata dengan frekuensi tinggi dalam dokumen. Metode ke-lima adalah *Classical TF-IDF Model 2,* yaitu metode klasik TF-IDF yang dikombinasikan dengan *sub-linear normalized TF-IDF* dengan prinsip bahwa dokumen mempunyai ikatan yang kuar dengan kata yang memiliki skor TF-IDF lebih tinggi. Formula untuk metode tersebut dapat dilihat pada persamaan 2.1.

$$TF - IDF_{t,d} = freq_{t,d} \cdot log \frac{N+1}{df_t} \quad (2.1)$$

dengan

$freq_{t,d}$ = Frekuensi kata *t* dalam dokumen *d*

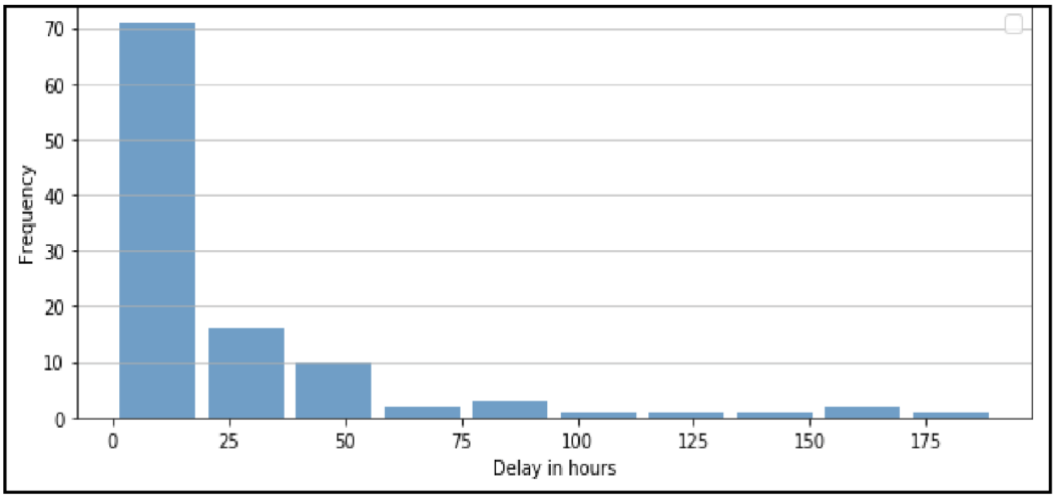
N = Jumlah dokumen dalam korpus

df_t = Jumlah dokumen dengan kata *t*

Berdasarkan hasil pengujian dengan membandingkan lima metode tersebut didapat bahwa metode *Normalized TF-IDF* dan *Sub-linear Normalized TF-IDF* menghasilkan nilai *similarity* yang lebih tinggi untuk dokumen yang bersesuaian dibandingkan dengan metode TF dan *Classical TF-IDF*.

Aleedy dkk (2019) mengembangkan *chatbot* berbasis *generative* dan menganalisis respon yang dihasilkan. *Dataset* yang digunakan adalah *Customer Support on Twitter* dari Kaggle yang berisi 2.811.774 *posts and replies* layanan tanya jawab dengan 108 *brands*. Saat melakukan eksplorasi dan analisis pada dataset, didapat bahwa Amazon menangani hingga 84.600 pertanyaan dalam tujuh bulan. Jumlah yang sangat banyak untuk dilayani *customer service* dengan memperhatikan jam dan hari kerja. Hal ini menyebabkan terjadinya *delay* dalam merespon pertanyaan pengguna hingga lebih dari 50 jam (2 hari) untuk hampir 10

brand. Delay yang terjadi dalam jam berbanding frekuensi pertanyaan ditunjukkan pada gambar 2.2.



Gambar 2.2 *Delay* layanan *customer service* (Aleedy dkk, 2019)

Gambar 2.2 menampilkan tingkat *delay* yang masih cukup banyak pada rentang waktu 25 jam atau lebih yang dapat memengaruhi kepuasan pengguna layanan. Data tersebut kemudian diolah menggunakan metode *preprocessing filtering,* yaitu membersihkan dataset dari tautan, gambar, angka, tanda baca, *mentions,* emoji, kata non-bahasa inggris, dan substitusi singkatan dengan bentuk kepanjangannya. Kemudian data diubah menjadi vektor menggunakan *Bag of Words* (BoW) agar *compatible* untuk proses *training. Chatbot* dibangun dengan tiga model yaitu LSTM, GRU dan CNN dengan 700.000 *data training* dan diuji dengan 30.000 *data testing.* Hasil pengujian menunjukkan keunggulan model LSTM dengan *BLEU Score* 0,36 dan nilai *cosine similarity* 0,8. Berdasarkan hasil tersebut dapat disimpulkan bahwa *generative based chatbot* dengan model LSTM dapat membantu merespon pertanyaan pengguna secara otomatis dengan operasional 24 jam.

Chatbot berbasis *retrieval* juga dikembangkan oleh Erline M. dan Christian Y. (2022) dengan metode *Knuth-Morris-Pratt* (KMP) untuk melayani pertanyaan pengunjung website mengenai informasi umum universitas. Sebanyak 193 pertanyaan contoh dalam FAQ digunakan sebagai dataset yang terbagi dalam 18 *intents* disertai jawaban yang didapat dari teknik *web scrapping* pada *official*

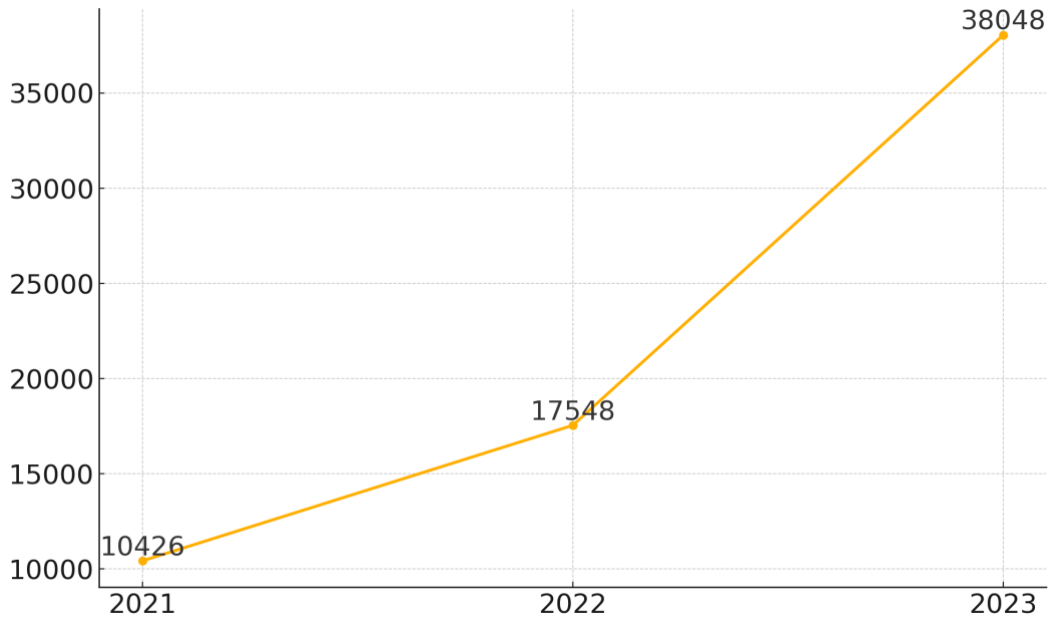
website. Metode *preprocessing* yang diaplikasikan pada pengembangan *chatbot* adalah *tokenization* dan *lemmatization* dengan tujuan mendapatkan setiap kata dasar dalam dataset sehingga memudahkan dalam membagi menjadi kelompok-kelompok *intent.* Pengembangan dilakukan dengan bahasa pemrograman *python* dengan *library Natural Language Toolkit* (NLTK). *Chatbot* kemudian diuji dengan teknik *black box testing* menggunakan 100 pertanyaan sebagai data *testing.* Pengujian menghasilkan nilai akurasi 86%, dengan kesimpulan *chatbot* dapat membantu memberikan informasi yang dibutuhkan pengguna secara *real time,* kapanpun dan dimanapun tanpa harus mengunjungi bagian informasi universitas.

Dalam hal mereplikasi cara manusia dalam memproduksi rangkaian kalimat untuk menjawab suatu pertanyaan merupakan tugas yang rumit bagi sebuah *chatbot.* Marche (2021) dalam gagasannya yang berjudul "*The Chatbot Problem"* mengungkapkan bahwa kekuatan *Artificial Intelligence* (AI) dapat menjadi berbahaya, ada kemampuan unik dalam AI untuk memproses bahasa natural menjadi menyeramkan. Pada tahun 2016 *chatbot* buatan *Microsoft* yang bernama *Tay* menjadi *chatbot* rasis dan disebut anti feminisme dalam 16 jam setelah *launching* yang memaksa perusahaan untuk melakukan *take down* pada *chatbot* tersebut. *Natural Language Processing* telah menampakkan percakapan yang kurang menyenangkan ke permukaan, menimbulkan pertanyaan bagi perkembangan teknologi "*ethical framework* apa yang dapat digunakan sebagai acuan dalam distribusi dan pemahaman berbahasa?" pertanyaan yang kemudian menjadi tantangan bagi keberlanjutan pemanfaatan teknologi. *Chatbot* sebenarnya hanya sebuah alat yang netral tidak memiliki tendensi positif dan negatif dalam inti kerangkanya, yang kemudian menjadi masalah adalah ketika dilatih dalam kapabilitas berbahasa berdasarkan data masukan. Pemilihan *dataset* metode pengolahannya menjadi penting dalam pengembangan *chatbot.* Dengan data yang tepat, *chatbot* dapat menjadi alat yang bermanfaat dan membantu pekerjaan manusia.

2.2. Kondisi Layanan *Helpdesk* UNNES

Sebelum pandemi COVID-19, UNNES menyediakan layanan helpdesk secara tatap muka (offline). Namun, setelah pandemi berlangsung, layanan bantuan

mulai beralih ke format daring (online) dengan memanfaatkan email dan *live chat*. Layanan obrolan langsung (*live chat*) memungkinkan sesi tanya jawab secara waktu nyata dengan Customer Service (CS) dan menjadi salah satu metode interaksi pilihan karena responnya yang cepat (Adamopoulou & Moussiades, 2020). Meski demikian, penanganan yang bersifat individual terhadap setiap pengguna membuat CS perlu berhadapan dengan volume permintaan yang kian bertambah. Pada Gambar 2.3, diperlihatkan peningkatan jumlah permohonan layanan *live chat* di helpdesk UNNES dari tahun 2021 hingga 2023.

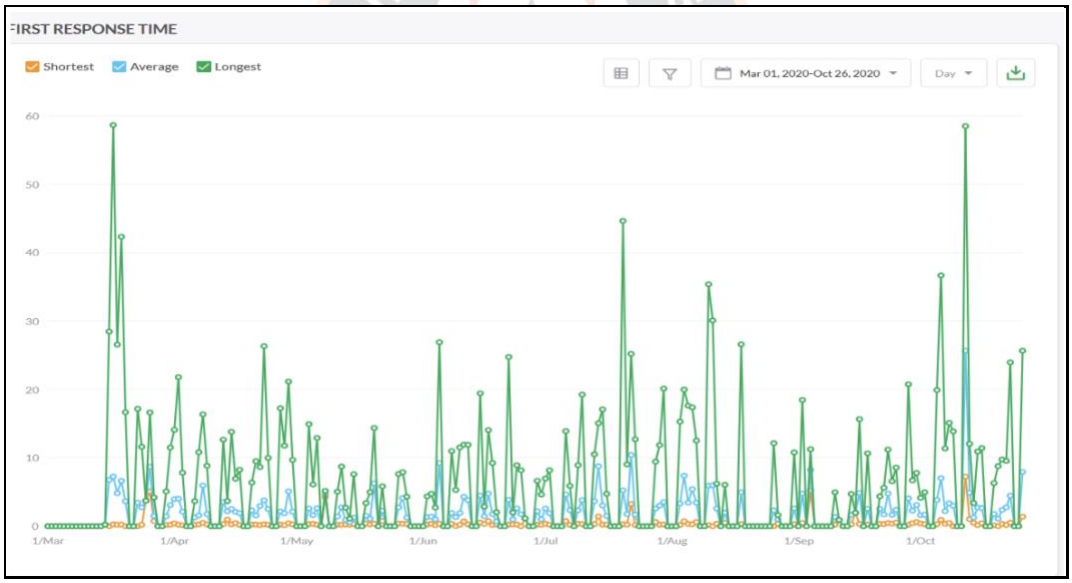


Gambar 2.3 Jumlah layanan *live chat* UNNES tahun 2021 hingga 2023

Grafik ini menggunakan sumbu horizontal (x) untuk tahun dan sumbu vertikal (y) untuk jumlah permohonan harian yang dirata-ratakan sepanjang periode tersebut. Garis kuning dalam grafik mencerminkan tren peningkatan yang cukup signifikan. Dalam konteks penelitian ini, penting untuk menunjukkan bahwa rerata harian *live chat* tahun 2021, 2022, dan 2023 masing-masing adalah 29, 49, dan 105 aduan per hari. Meningkatnya frekuensi aduan ini tidak hanya menunjukkan bertambahnya kebutuhan layanan oleh pengguna, tetapi juga menjadi sinyal bahwa tenaga CS konvensional bisa kewalahan jika tidak diimbangi oleh peningkatan

kualitas atau efisiensi. Islam (2024) turut menggarisbawahi pentingnya menjaga reputasi dan kualitas layanan institusi melalui pengelolaan keluhan yang efisien.

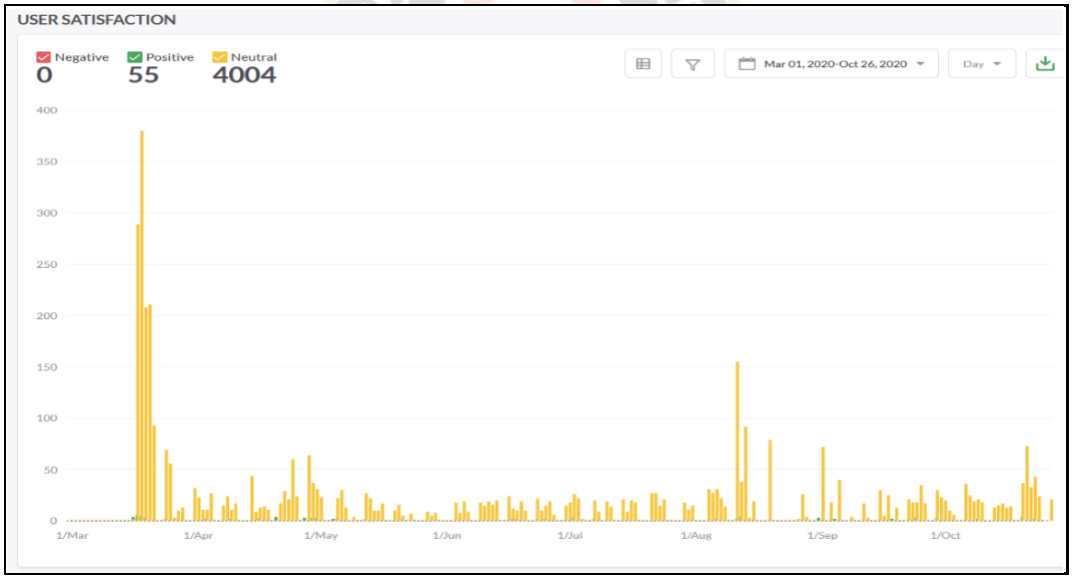
Faktor lain yang berpengaruh terhadap efektivitas layanan *helpdesk* di UNNES adalah lamanya sesi *chat* yang harus dilakukan oleh CS. Rata-rata durasi sesi *chat* mencapai 16 menit 20 detik dan durasi terlama mencapai 115 menit. Variasi ini dapat disebabkan oleh beragam faktor, seperti tingkat kesulitan pertanyaan pengguna, ketersediaan personel CS, serta kepadatan antrean. Dengan demikian, durasi yang terlalu lama berpotensi menyebabkan antrean menumpuk, sedangkan durasi terpendek mengindikasikan adanya pertanyaan sederhana atau proses penanganan yang lancar. Kelancaran pada penanganan tiket aduan salah satunya ditentukan pada waktu tunggu respon pertama saat pengguna mengirimkan tiket aduan, seperti ditunjukkan pada Gambar 2.4.



Gambar 2.4 Grafik waktu tunggu respon pertama di *helpdesk* UNNES

Gambar 2.4 berfokus pada waktu tunggu respon pertama. Grafik ini menampilkan sumbu horizontal (x) berupa satuan hari, sedangkan sumbu vertikal (y) merepresentasikan waktu tunggu (dalam menit) mulai dari tanggapan paling cepat hingga paling lama. Pada grafik ini, warna oranye biasanya menunjukkan nilai tanggapan tercepat, garis biru untuk rata-rata, dan garis hijau untuk tanggapan terlama. Data menunjukkan bahwa rata-rata waktu tunggu tanggapan awal adalah 5

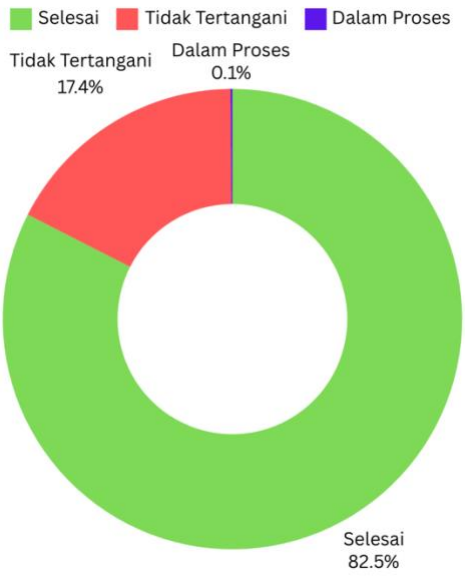
menit, dengan waktu tercepat mendekati 0 menit dan waktu terlama mendekati 1 jam. Implikasinya, keterlambatan respons pertama berisiko menurunkan kepuasan pengguna, sebab pelanggan yang mengajukan keluhan umumnya membutuhkan jawaban instan (Broadbent & Lodge, 2021). Kinerja *helpdesk* akan dipersepsikan lebih baik jika respon pertama ini diberikan secara cepat, sehingga pengguna setidaknya mengetahui bahwa pertanyaannya sedang dalam proses penanganan. Kecepatan penanganan awal ini juga mempengaruhi keberlanjutan sesi *chat* dengan pengguna, dengan lamanya respon pertama, pengguna dapat menganggap bahwa layanan sedang tidak berfungsi dengan baik atau ada antrean yang menyebabkan layanan terganggu, tentu hal ini dapat mempengaruhi kepuasan pengguna. Kaitannya dengan persepsi pengguna, Gambar 2.5 menyediakan gambaran mengenai tingkat kepuasan mereka terhadap layanan CS.



Gambar 2.5 Skor kepuasan pengguna pada *helpdesk* UNNES

Grafik pada Gambar 2.5 memuat sumbu x sebagai dimensi waktu dan sumbu y untuk jumlah penilaian yang diterima. Bar kuning menggambarkan tanggapan netral, sementara penilaian positif atau negatif (jika ada) ditampilkan dengan bar warna merah dan hijau, meskipun jumlahnya tampak sangat sedikit. Dari total 4004 penilaian, sekitar 99% di antaranya bersifat netral dan hanya 1% yang positif. Persentase yang cukup besar untuk penilaian netral ini menekankan bahwa mayoritas pengguna tidak mengungkapkan ketidakpuasan, tetapi juga belum

menyatakan kepuasan yang memuaskan. Dengan kata lain, terdapat potensi peningkatan penilaian positif apabila waktu respon dapat diperbaiki serta informasi yang diberikan lebih relevan dan tepat sasaran, sehingga dapat mengurangi aduan yang ditinggalkan tanpa mengisi kuesioner kepuasan seperti ditunjukkan pada gambar 2.6.



Gambar 2.6 Persentase Status Penyelesaian Aduan oleh *helpdesk* UNNES

Gambar 2.6 menunjukkan tingkat penyelesaian aduan tiket pada *helpdesk* UNNES dengan warna hijau (Selesai), merah (Tidak Tertangani), dan ungu (Dalam Proses). Pada diagram tersebut, terlihat bahwa 82,5% aduan berhasil terselesaikan oleh CS, 17,4% atau sekitar 6.606 aduan tidak tertangani, dan 0,1% masih dalam proses. Persentase aduan yang tidak tertangani ini sangat relevan dengan pendapat MacDoland (2020) yang menyatakan bahwa sekitar 21% antrian *live chat* dapat terabaikan pada layanan helpdesk pada umumnya. Melalui data ini, dapat disimpulkan bahwa meski sebagian besar aduan berhasil diselesaikan, masih terdapat kendala dalam penanganan keluhan tertentu yang tidak terantisipasi oleh CS. Keterbatasan jumlah CS, kurangnya optimasi pelayanan, serta rendahnya intensitas pemanfaatan FAQ oleh pengguna menjadi alasan mengapa persentase aduan yang tidak terlayani cukup tinggi.

2.3. *Chat Service*

Chat service merupakan salah satu jenis komunikasi internet yang menawarkan transmisi instan antara pengirim dan penerima dalam bentuk teks/*chat* (Elmorshidy, 2013). Layanan *chat service* atau yang biasa disebut *live chat support* merupakan salah satu solusi pelayanan *real time* untuk melayani pertanyaan/aduan yang dihadapi customer dalam menggunakan produk atau jasa. Singkatnya waktu tunggu respon balasan pada *live chat,* layanan ini mendapat nilai *satisfaction rate* tertinggi dibandingkan kanal-kanal pelayanan informasi lainnya, yaitu mencapai 73%, diikuti oleh email dengan *rate* 61% (Reijrink, 2020). *Customer* dapat secara langsung berkomunikasi dengan *customer service* untuk tanya jawab mengenai masalah yang dialami *customer*.

Pelayanan yang dilakukan secara langsung oleh *customer service* menyebabkan kebutuhan tenaga *customer service* meningkat. *Scalability* perusahaan dan bertambahnya *customer* membuat antrian *chat* yang perlu dilayani semakin banyak, sehingga mengakibatkan sebagian *chat* terabaikan. Sebanyak 21% dari antrian *live chat* tidak terlayani/terabaikan (MacDonald, 2020).

2.4. *Natural Language Processing*

Natural Language Processing (NLP) merupakan cabang dari kecerdasan buatan (*Artificial Intelligence*) yang fokus pada interaksi antara komputer dan bahasa manusia. Tujuan utama NLP adalah memungkinkan komputer untuk memahami, menganalisis, menghasilkan, dan merespons bahasa alami dengan cara yang mirip manusia (Jurafsky & Martin, 2020). NLP menggabungkan berbagai disiplin ilmu seperti linguistik, ilmu komputer, dan matematika untuk mengembangkan model dan algoritma yang dapat memproses data tekstual dan lisan.

Komponen utama dalam NLP meliputi tokenisasi, yaitu proses memecah teks menjadi unit-unit yang lebih kecil seperti kata atau frasa, *part-of-speech* (POS) *tagging*, yang menentukan kategori gramatikal setiap kata dalam kalimat, *parsing* sintaksis, yang menganalisis struktur kalimat untuk memahami hubungan antar kata

dan frasa, *named entity recognition* (NER), yang mengidentifikasi entitas penting dalam teks seperti nama orang, tempat, dan organisasi, *sentiment analysis*, yang menentukan emosi atau opini dalam teks, serta *machine translation*, yang menerjemahkan teks dari satu bahasa ke bahasa lain secara otomatis (Jurafsky & Martin, 2020). Teknik dan metode dalam NLP telah berkembang dari pendekatan berbasis aturan (*rule-based*) menuju metode berbasis statistik dan *deep learning*. *Pattern matching*, misalnya, melibatkan pencocokan pola tertentu dalam teks untuk mengekstrak informasi atau memahami struktur kalimat, yang sering digunakan dalam aplikasi seperti *chatbot* sederhana (Hearst, 1992). M*achine learning* menggunakan algoritma statistik untuk mempelajari pola dari data, sementara *deep learning* dengan arsitektur *neural network* seperti *Recurrent Neural Networks* (RNN), *Long Short-Term Memory* (LSTM), dan *Transformer* telah merevolusi kemampuan NLP dalam memahami konteks dan makna teks (Goldberg, 2016). Model *sequence-to-sequence* (Seq2Seq) dirancang untuk mentransformasikan urutan input menjadi urutan output yang berbeda, sangat efektif dalam tugas-tugas seperti penerjemahan mesin, peringkasan teks, dan pembuatan *chatbot* yang menghasilkan respons relevan dan koheren (Sutskever dkk, 2014). Aplikasi NLP dalam pengembangan *chatbot* sangat luas dan berkembang pesat. *Chatbot* digunakan untuk menyediakan layanan otomatis dalam berbagai bidang seperti layanan pelanggan, pendidikan, dan kesehatan. Pengembangan *chatbot* melibatkan komponen NLP seperti pemahaman bahasa alami (*Natural Language Understanding*/NLU), *Natural Language Generation* (NLG), serta manajemen dialog. Metode *pattern matching* digunakan untuk mengenali pola tertentu dalam input pengguna dan memberikan respons yang sesuai, sementara model Seq2Seq memungkinkan *chatbot* untuk memahami konteks dan menghasilkan respons yang lebih alami dan variatif (El-Sallab dkk, 2015).

Evaluasi kinerja model NLP sangat penting untuk memastikan bahwa model tersebut berfungsi dengan baik dan memenuhi tujuan yang diinginkan. Berbagai metrik digunakan untuk mengevaluasi kinerja, tergantung pada tugas spesifik yang dilakukan oleh model. Salah satu metrik evaluasi yang umum digunakan dalam penerjemahan mesin dan pengembangan *chatbot* adalah BLEU (*Bilingual*

Evaluation Understudy) *Score*, yang mengukur kualitas terjemahan dengan membandingkan *output* mesin dengan referensi terjemahan yang telah ada (Papineni dkk, 2002). Selain BLEU, metrik lain seperti *Precision, Recall, F1-Score,* dan *METEOR* juga sering digunakan untuk mengevaluasi tugas-tugas NLP tertentu (Sokolovska dkk, 2019). Dengan memahami teori dan teknik dasar NLP, pengembangan sistem *chatbot* tanya jawab otomatis dapat dilakukan secara lebih efektif dan efisien. Pemanfaatan berbagai metode seperti *pattern matching* dan *sequence-to-sequence models* memungkinkan *chatbot* untuk memahami dan menghasilkan bahasa manusia dengan tingkat kompleksitas yang tinggi, sementara evaluasi yang tepat melalui metrik seperti BLEU *Score* memastikan bahwa model yang dikembangkan memenuhi standar kualitas yang diharapkan.

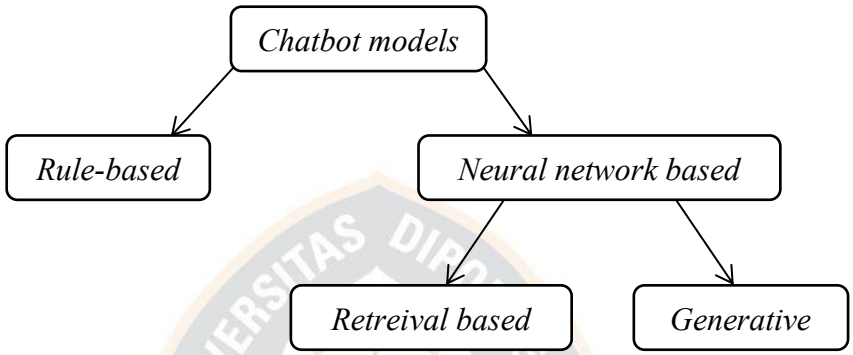
2.5. *Chatbot*

Chatbot terdiri dari dua kata penyusun yaitu "*chat*" dan "*robot*". *Chatbot* adalah program komputer yang meniru bahasa manusia dengan bantuan sistem dialog berbasis teks (Zumstein dan Hundertmark, 2017). Eliza merupakan *chatbot* pertama yang berhasil dibangun pada tahun 1966. Eliza menyimulasikan seorang psikoterapi berkomunikasi dengan mengembalikan kalimat pengguna dalam bentuk pertanyaan interogatif (Adamopoulou dan Moussiades, 2020). Kemampuan komunikasinya masih terbatas, namun menjadi inspirasi untuk pengembangan chatbot lainnya (Klopfenstein dkk, 2017).

Perkembangan chatbot yang sangat pesat membuat pilihan algoritma dalam membangun chatbot semakin beragam. Secara umum, *chatbot* bekerja dengan menggunakan basis pengetahuan sebagai landasan pemilihan jawaban atas pertanyaan *end user.* Basis pengetahuan ini kemudian digunakan sebagai masukan sebuah model pada proses *training,* sehingga model tersebut dapat memiliki kecerdasan saat menerima masukan data uji.

Berdasarkan model pemrosesannya, *chatbot* dibagi menjadi dua kategori besar yaitu *rule based method* dan *AI based method. Rule based* merupakan chatbot yang melakukan *pemindaian* langsung pada *user input* dengan frasa dan *template* yang sudah tersedia untuk memproduksi respon. Yang kedua, *AI based method* atau

neural-network-based approach merupakan metode yang memanfaatkan *deep learning* dalam memproduksi respons, chatbot model di-*training* menggunakan dataset besar sehingga dapat menghasilkan respon yang relevan dan sesuai grammar (Agarwal, 2020). Klasifikasi chatbot dalam memproduksi respon ditunjukkan pada gambar 2.7.



Gambar 2.7 Klasifikasi *Chatbot*

Berdasarkan fungsionalitasnya, chatbot dibagi menjadi dua tipe, *task-oriented chatbot* dan *open-domain chatbot. Task-oriented* merupakan tipe *chatbot* yang sangat mahir dalam mengeksekusi masalah spesifik dan menangani topik domain tertentu, seperti reservasi restoran, reservasi tiket penerbangan, promosi film dan sebagainya. Kedua, *open-domain chatbot* merupakan tipe *conversational* yang mencoba meniru percakapan manusia, tujuannya adalah untuk menghasilkan respon yang sangat mendekati perilaku manusia dalam melakukan percakapan sehingga pengguna merasa dilayani manusia sungguhan (Agarwal 2020).

Pendekatan algoritma pembangun *core chatbot* semakin beragam. Beberapa pendekatan yang cukup popular diantaranya adalah *parsing*. Teknik *parsing* menggunakan proses analisis pada teks masukan dan memanipulasinya dengan menggunakan beberapa *layer* fungsi *Natural Language Processing* (NLP). *Pattern matching* atau pencocokan pola adalah Teknik yang digunakan banyak chatbot dan cukup populer dalam sistem tanya-jawab yang bergantung pada tipe teks yang ditanyakan, kalimat ungkapan singkat atau makna semantik dari kalimat masukan. *Artificial Intelligence Markup Language* (AIML) adalah sebuah Bahasa markup yang sangat umum digunakan pada basis pengetahuan dalam membangun *chatbot.*

Chat script adalah teknik yang melengkapi AIML, ketika tidak ada pola yang cocok pada data AIML. *Chat script* sudah memiliki konsep sintaks dan logika sederhana dimana didalamnya juga mengenal konsep variabel. *SQL and Relational Database* merupakan teknik *query* respon berdasarkan pada basis data. *Chatbot* yang didesain menggunakan metode ini akan mengenali konteks, yaitu mengingat percakapan-percakapan sebelumnya. Teknik markov *chain* digunakan untuk membangun *chatbot* ang memungkinkan menghasilkan respon yang lebih konsisten. *Language Tricks* adalah teknik menambahkan variasi basis pengetahuan berupa frasa, kalimat atau bahkan paragraf yang ditambahkan pada data basis pengetahuan dengan tujuan untuk memperkaya jangkauan pertanyaan yang dapat dijawab dengan benar (Abdul-Kader dan Woods, 2015).

2.6. *Pattern Matching* pada pengembangan *Chatbot*

Pattern matching merupakan salah satu teknik yang cukup sering digunakan dalam membangun *chatbot. Pattern Matching* adalah proses pengecekan token masukan atau kalimat berdasarkan ada atau tidaknya sebuah unsur dalam pola. (Nivedhitha, 2021). *Chatbot* dibangun untuk membantu melayani *customer* dalam memecahkan masalah. Basis pengetahuan yang digunakan dapat berupa data tanya-jawab atau yang biasa disebut dengan *Frequently Asked Questions* (FAQ). FAQ adalah suatu data pasangan pertanyaan dan jawaban mengenai informasi umum yang paling sering ditanyakan *customer* dengan tujuan memberi fasilitas *self solving* untuk para pengguna. Teknik *pattern matching* menggunakan basis pengetahuan berbentuk FAQ. Data FAQ ini kemudian diolah pada proses *training* sehingga model dapat mengenali pasangan pertanyaan dan jawaban yang diberikan. Eliza dan Alice adalah contoh *chatbot* pendahulu yang menggunakan metode *pattern matching* (Adamopoulou, 2020). Cara kerja *pattern matching* adalah dengan cara mencari pertanyaan yang paling sesuai menggunakan parameter-parameter dan prinsip *machine learning.* Setelah mendapatkan pertanyaan yang sesuai, ambil pasangan jawaban pertanyaan tersebut kemudian kembalikan pada *user* sebagai respon. Jadi, semua respon yang dihasilkan metode *pattern matching*

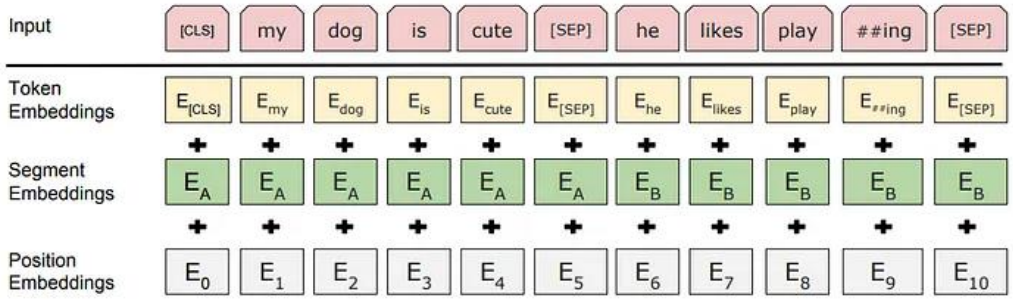
adalah data yang sudah diterima model saat *training,* tidak ada bentuk respon di luar data tersebut.

Salah satu kelemahan menggunakan metode *pattern matching* adalah pilihan respon yang sudah ditetapkan saat *training* dan tidak dapat dikostumisasi saat *chatbot* berjalan. Metode *pattern matching* juga tidak menggunakan proses *compose response,* dimana memungkinkan adanya jawaban baru diluar dataset yang disusun *real time* saat *query* pertanyaan masuk (Shang dkk., 2015).

2.6.1. BERT Similarity

Sejak diperkenalkannya BERT (*Bidirectional Encoder Representations from Transformers*) oleh Devlin dkk. pada tahun 2019, teknologi ini telah menjadi dasar penting untuk berbagai tugas pemrosesan NLP. Dalam perkembangan terkini, BERT sangat berpengaruh dalam pengembangan metode untuk menilai kesamaan teks, sebuah aplikasi yang sangat penting dalam bidang pencarian informasi, pengelompokan dokumen, dan sistem jawaban pertanyaan. Sifat bidirectional BERT memungkinkannya untuk memahami konteks kata dalam sebuah kalimat secara lebih mendalam dari pada model *unidirectional*, memungkinkan BERT untuk menangkap nuansa makna yang esensial untuk menentukan kesamaan teks.

Langkah-langkah algoritma BERT dimulai dengan proses tokenisasi, vektorisasi, mengubah vektor kalimat menjadi matriks *embeddings,* kemudian menghasilkan *similarity score* dengan membandingkan dua kalimat seperti ditunjukkan pada gambar 2.8.



Gambar 2.8 Algoritma BERT (Devlin dkk, 2019)

Langkah-langkah algoritma BERT:

a. Tokenisasi

Tokenisasi adalah proses mengubah kalimat atau teks T menjadi sekumpulan token $\{t_1, t_2, t_3, \dots, t_m\}$ di mana setiap t_j a adalah pecahan kata/frasa yang lebih kecil.

b. Vektorisasi

Pada proses vektorisasi, token-token t_j diubah menjadi vektor numerik v_j dalam ruang vektor yang dimensinya ditentukan oleh model yang BERT digunakan.

c. Pembentukan *Embeddings*

Proses ini mengekstrak embedding kalimat dari matriks vektor token, yang mewakili konteks semantik keseluruhan kalimat.

Penelitian terkini telah fokus pada pemanfaatan model BERT yang telah dilatih sebelumnya untuk menghitung kesamaan semantik antar teks. Dengan melakukan *fine-tuning* pada BERT berbasis dataset spesifik, para peneliti dapat meningkatkan kinerja model dalam mengidentifikasi teks yang serupa, bahkan ketika teks tersebut menggunakan kata-kata atau struktur yang berbeda. Sebagai contoh, sebuah studi pada tahun 2022 menunjukkan bahwa model BERT yang telah di-*finetune* menghasilkan hasil generasi lebih baik dan lebih informatif dengan menangkap informasi dalam teks untuk dataset domain tertentu (Qiu Y. dan Jin Y., 2022). Model-model ini tidak hanya lebih akurat tetapi juga jauh lebih cepat dalam memproses volume teks yang besar, yang sangat krusial untuk aplikasi *real-time*.

Selain itu, penggunaan BERT terhadap berbagai bahasa dan domain memperluas utilitasnya dalam aplikasi global. Seiring dengan kemajuan teknologi NLP, peningkatan berkelanjutan dalam arsitektur berbasis BERT diharapkan akan mendorong kemampuan analisis semantik, menyediakan hasil yang lebih tepat dan relevan secara kontekstual. Metodologi yang terus berkembang ini menekankan potensi model berbasis *transformer* untuk merevolusi cara mesin memahami bahasa manusia, menandai pergeseran signifikan dalam komputasi linguistik.

2.6.2 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) adalah salah satu teknik dalam pemrosesan teks yang digunakan untuk mengevaluasi pentingnya sebuah kata dalam sebuah dokumen relatif terhadap sebuah kumpulan dokumen (Lan, 2022).

a. *Term Frequency* (TF) mengukur seberapa sering sebuah kata muncul dalam sebuah dokumen. Nilai TF dihitung menggunakan Persamaan 2.1 dengan membagi jumlah kemunculan kata tertentu dalam dokumen dengan jumlah total kata dalam dokumen tersebut.

$$\mathrm{TF(t, d)} = \frac{Jumlah\ kemunculan\ t\ dalam\ d}{Jumlah\ kata\ dalam\ d} \tag{2.1}$$

dengan

t = Kata tertentu (*term*) yang sedang diukur frekuensinya.

d = Dokumen tempat kata tertentu tersebut muncul.

b. *Inverse Document Frequency* (IDF) mengukur seberapa penting sebuah kata dalam keseluruhan korpus. Kata yang muncul dalam banyak dokumen akan memiliki nilai IDF yang rendah, sedangkan kata yang jarang muncul akan memiliki nilai IDF yang tinggi. Dihitung dengan Persamaan 2.2.

$$\mathrm{IDF(t, d)} = \log\left(\frac{N}{DF(t)}\right) \tag{2.2}$$

dengan

t = Kata tertentu (term) yang sedang diukur pentingnya dalam korpus.

D = Korpus (kumpulan dokumen) tempat kata tersebut dianalisis.

N = Total jumlah dokumen dalam korpus D.

DF(t) = Jumlah dokumen dalam korpus D yang mengandung kata t.

TF-IDF adalah hasil perkalian antara TF dan IDF seperti pada Persamaan 2.3. Menggabungkan kedua metrik ini untuk memberikan bobot pada kata-kata

berdasarkan frekuensi kemunculan mereka dalam dokumen tertentu serta kepentingannya dalam korpus.

$$\mathrm{TF-IDF(t, d, D)} = \mathrm{TF(t, d)} \times \mathrm{IDF(t, D)} \tag{2.3}$$

dengan

- **TF(t, d)**: Nilai *Term Frequency* dari kata t dalam dokumen d.
- **IDF(t, D)**: Nilai *Inverse Document Frequency* dari kata t dalam korpus D.

2.6.3 USE

Universal Sentence Encoder (USE) adalah model pembelajaran mesin yang dirancang untuk menghasilkan representasi vektor dari kalimat atau paragraf yang bersifat universal dan dapat digunakan dalam berbagai tugas *Natural Language Processing* (NLP). Representasi ini dapat digunakan untuk berbagai aplikasi seperti pengelompokan teks, pencarian semantik, klasifikasi teks, dan analisis sentimen (Deepthi & Sowjanya, 2021).

USE dapat digunakan dalam *Chatbot Query Retrieval* untuk meningkatkan kemampuan pemahaman dan pencarian jawaban terhadap pertanyaan pengguna. Proses tersebut melibatkan tiga langkah utama, diantaranya:

a. Pengolahan teks input dari pengguna yang dikonversi menjadi *embeddings* menggunakan model.
b. Pencocokan semantik dengan membandingkan pertanyaan pengguna dengan respons dari basis data yang telah disiapkan untuk menemukan keoccokan semantic terbaik. Salah satu metode yang paling sering digunakan untuk mengukur kesamaan antara *embeddings* adalah *cosine similarity* seperti pada Persamaan 2.4.

$$cosine\_similarity(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{2.4}$$

Dimana *A* dan *B* adalah *embeddings* dari kalimat yang dibandingkan. *Cosine similarity* memberikan nilai antara -1 dan 1, di mana nilai 1 menunjukkan kesamaan sempurna.

c. Pemilihan respon yang paling relevan berdasarkan tingkat kesamaan semantic tertinggi.

2.6.4 SBERT

Sentence-BERT (SBERT) adalah modifikasi dari BERT yang dirancang untuk meningkatkan kinerja dalam tugas pencocokan semantik dan pengukuran kesamaan teks pada tingkat kalimat. SBERT menggabungkan arsitektur BERT dengan jaringan saraf *siamese* atau *triplet* untuk menghasilkan *embeddings* yang dapat dibandingkan secara langsung menggunakan metrik jarak seperti *cosine similarity* (Reimers & Gurevych, 2019). SBERT secara khusus dioptimalkan untuk tugas-tugas seperti pencarian semantik, klasifikasi teks, dan clustering teks (Wang & Kuo, 2020). SBERT mengadopsi artsitektur dasar BERT dengan beberapa modifikasi penting.

a. *Input Layer*

Kalimat input diproses dengan cara yang sama seperti pada BERT, di mana setiap token dalam kalimat dikonversi menjadi representasi vektor menggunakan embedding yang telah dilatih sebelumnya.

b. *Transformer Layer*

Seperti BERT, SBERT menggunakan beberapa lapisan transformer yang memungkinkan model untuk memahami konteks kata dalam kalimat secara bidirectional.

c. *Pooling Layer*

Setelah melalui lapisan *transformer*, representasi vektor dari token-token dalam kalimat digabungkan menggunakan teknik *pooling* (seperti *mean pooling* atau *max pooling*) untuk menghasilkan satu vektor tetap yang mewakili keseluruhan kalimat.

SBERT memungkinkan pembelajaran representasi kalimat yang lebih efisien dan akurat, memungkinkan pemodelan hubungan semantik antar kalimat dengan lebih baik. Ini membuat SBERT sangat berguna dalam aplikasi seperti pencarian dokumen, penarikan informasi, dan *chatbot* yang membutuhkan pemahaman konteks yang mendalam (Tao et al., 2019).

2.6.5 GloVe

Global Vectors for Word Representation (GloVe) adalah metode pembelajaran terdistribusi untuk menghasilkan representasi kata (*word embeddings*) yang dikembangkan oleh Stanford University. GloVe menggabungkan pendekatan berbasis matriks *co-occurrence* dengan pembelajaran vektor kata yang dioptimalkan untuk menangkap makna semantik dari kata dalam teks. Model GloVe dibangun dari statistik global dari korpus besar, di mana setiap kata direpresentasikan sebagai vektor dalam ruang multidimensi (Pennington et al., 2014).

GloVe dirancang untuk menangkap informasi semantik dan sintaktik dari kata-kata, memungkinkan vektor kata yang dihasilkan untuk digunakan dalam berbagai aplikasi NLP seperti klasifikasi teks, analisis sentimen, dan penarikan informasi. Kelebihan GloVe termasuk efisiensi komputasi dan kemampuan untuk menghasilkan representasi kata yang sangat akurat dari korpus besar (Mikolov et al., 2013).

Dimulai dengan membangun matriks *co-occurrence* yang mencatat seberapa sering pasangan kata muncul bersama dalam konteks tertentu dalam korpus besar. Model GloVe dilatih untuk meminimalkan fungsi *loss* yang menghubungkan representasi vektor kata-kata dengan probabilitas kemunculan bersama mereka (Sakketou & Ampazis, 2020). Fungsi *loss* ini dirancang untuk memastikan bahwa rasio probabilitas kemunculan bersama kata-kata berhubungan dengan jarak *Euclidean* antara vektor mereka dalam ruang *embedding* seperti pada Persamaan 2.5.

$$J = \sum_{i,j=1}^{V} f(X_{i,j})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log(X_{i,j}))^2 \quad (2.5)$$

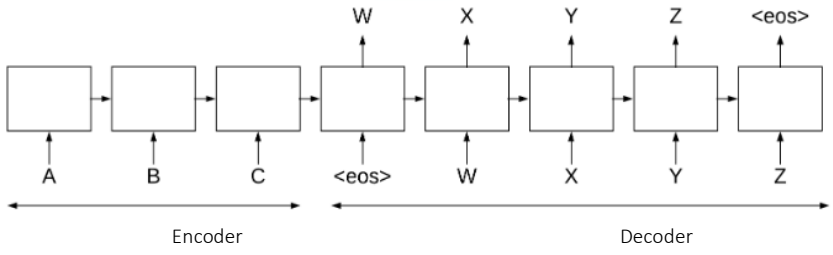
Pada Persamaan 2.5 $X_{i,j}$ adalah frekuensi kemunculan bersama kata i dan j, w_i dan $\tilde{w}_j$ adalah vektor representasi kata, dan b_i dan $\tilde{b}_j$ adalah bias.

2.7 Sequence-to-sequence pada pengembangan *Chatbot*

Pendekatan yang berbeda dilakukan oleh (Shang dkk., 2015) pada penelitiannya. Pengembangan *chatbot* menggunakan konsep *Statistical Machine*

Translation (SMT) di mana respon yang dihasilkan dapat lebih fleksibel berdasarkan masukan kalimat pertanyaan. Metode ini termasuk dalam kategori *generative responder* karena respon yang dihasilkan merupakan hasil komposisi dari sejumlah basis pengetahuan yang digunakan saat *training.*

Sequence-to-sequence (seq2seq) merupakan salah satu algoritma *deep learning* yang biasa digunakan pada *machine translation* yang dapat diadopsi menjadi algoritma *query* tanya-jawab (Chandra dan Suyanto, 2019). *Seq2seq* menggunakan konsep *Recurrent Neural Network* (RNN), yaitu pengembangan dari *traditional neural network* yang memperhatikan informasi yang dihasilkan dari proses pada jaringan sebelumnya. *Recurrent Neural Network* digambarkan seperti perulangan di mana setiap jaringan menerima input dari jaringan sebelumnya, melakukan sejumlah proses kemudian meneruskan informasi ke jaringan berikutnya. Kelemahan pada jaringan RNN adalah adanya *gap* antara informasi dari jaringan sebelumnya yang tidak memenuhi kebutuhan pada jaringan yang sedang melakukan pemrosesan. *Long Short-Term Memory (LSTM)* merupakan bentuk lain dari RNN yang dapat mengatasi masalah tersebut. LSTM mempunyai ingatan memori yang dapat menyimpan informasi lebih lama, sehingga dapat meningkatkan akurasi dari pemrosesan yang dilakukan pada *current network*. *Seq2seq* menggunakan dua layer LSTM sebagai *encoder* dan *decoder* sebagaimana dapat dilihat pada gambar 2.9.



Gambar 2.9 Arsitektur *seq2seq model*

Bagian sebelah kiri/*encoder* adalah teks masukan berupa kalimat yang diubah menjadi vector konteks (satu kata satu komponen vektor) yang kemudian diproses menggunakan *Recurrent Neural Network* sehingga menghasilkan vector target pada sebelah kanan. Vektor hasil tersebut kemudian di-*decode* untuk

mendapatkan kalimat respon secara utuh. Proses ini berjalan *sequence to sequence* atau bagian demi bagian di mana satu vektor hasil merupakan hasil generasi dari satu vektor masukan. Hal ini yang menyebabkan respon pada metode *sequence to sequence* dapat berbeda dari dataset yang digunakan.

Proses pada algoritma *seq2seq* secara umum adalah *encoder-decoder*, namun dalam tahap rincinya dimulai dari proses tokenisasi Tahap persiapan data, data yang akan diolah dipersiapkan agar dapat diproses oleh model. Proses persiapan mencakup normalisasi teks, tokenisasi, pembuatan kosa kata (*vocabulary*), dan penambahan token khusus. Langkah pertama adalah membersihkan teks dari karakter-karakter yang tidak relevan, seperti tanda baca, lalu mengubah teks ke bentuk huruf kecil (*lowercase*).

- *input* asli : "Bagaimana cara reset password akun sikadu?"
- Setelah normalisasi (menghilangkan tanda "?"):

"bagaimana cara reset password akun sikadu"

Setelah teks dinormalisasi, kalimat dipecah (di-tokenisasi) menjadi deretan kata (token). Misalnya, dari kalimat “bagaimana cara reset password akun sikadu”, akan diubah menjadi:

["bagaimana", "cara", "reset", "password", "akun", "sikadu"]

Tujuannya adalah agar setiap kata dapat diperlakukan sebagai satuan unit yang akan diolah lebih lanjut oleh model.

Langkah selanjutnya adalah membangun sebuah *dictionary* yang berisi seluruh token unik (kosa kata) yang muncul di korpus pelatihan. Setiap token unik akan dipetakan ke indeks *integer* tertentu.

{<sos>, <eos>, <unk>, bagaimana, cara, reset, password, akun, sikadu,…}

Maka dapat ditetapkan indeks (*ilustrasi*):

- <sos> → 1
- <eos> → 2
- <unk> → 3
- "bagaimana" → 4
- "cara" → 5
- "reset" → 6

- "password" → 7
- "akun" → 8
- "sikadu" → 9

Saat *inference* (penggunaan *chatbot*), jika ditemui kata yang tidak dikenal, maka token tersebut akan dilabeli <unk>.

Token <sos> (*start of sequence*) dan <eos> (*end of sequence*) ditambahkan untuk menandai awal dan akhir suatu urutan. Pada tahap training atau inference, token <sos> akan diletakkan di awal urutan *input decoder*, sedangkan <eos> di akhir urutan target keluaran. Sehingga, apabila input yang akan di-*encode* adalah "bagaimana cara reset password akun sikadu", indeks yang dihasilkan menjadi:

[4, 5, 6, 7, 8, 9]

Sementara data target untuk *decoder* mungkin diawali <sos> dan diakhiri <eos>:

[1, ..., ..., 2]

Setelah setiap token dipetakan ke indeks *integer*, dibutuhkan representasi vektor untuk menangkap hubungan semantik antar-kata. Representasi ini umumnya diperoleh dari sebuah *embedding matrix* E berukuran ($V \times d$), di mana V adalah ukuran *vocabulary* dan d adalah dimensi *embedding* (misal 128 atau 256). Setiap token dengan indeks i akan direpresentasikan sebagai vektor $E[i]$ berdimensi d. ujuannya adalah agar kata-kata yang memiliki kemiripan semantik memiliki representasi vektor yang mirip.

Setelah *embedding* vektor diperoleh, pendekatan *seq2seq* menggunakan dua komponen utama, yaitu *encoder* dan *decoder*. *Encoder* merupakan RNN (atau LSTM/GRU) yang menerima serangkaian *embedding* dari token *input* $\{x1, x2, ..., x\mathrm{T}\}\{x1, x2, ..., x\mathrm{T}\}$. Pada setiap waktu t_t, *encoder* menghasilkan *hidden state* h_t. Jika digunakan LSTM, maka terdapat *cell state* c_t tambahan.

Pada LSTM, *hidden state* h_t dan *cell state* c_t dihitung dengan Persamaan 2.9 dan Persamaan 2.11.

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \tag{2.6}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.7}$$

$$\tilde{c}_t = \tan(W_c[h_{t-1}, x_t] + b_c) \quad (2.8)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (2.9)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (2.10)$$

$$h_t = o_t \odot \tan(c_t) \quad (2.11)$$

dengan

f_t = *forget gate*, i_t = *input gate*, o_t = *output gate*.

σ = *sigmoid function*, tanh = *hyperbolic tangent*, $\odot$ = operasi perkalian *element-wise*.

x_t = *input embedding* pada waktu t_t.

W dan b = parameter matriks dan *bias*.

Setelah proses ini, *encoder* akan menghasilkan *hidden state* akhir h_T (dan c_T pada LSTM) yang merangkum informasi keseluruhan urutan *input*.

Decoder merupakan RNN (atau LSTM/GRU) yang memprediksi urutan keluaran $\{y_1, y_2, \ldots, y_M\}$ dengan menggunakan hidden state terakhir dari encoder sebagai initial state $\left(tan_0^{dec} = tan_T^{enc}\right)$. Pada langkah *t*, decoder menerima masukan berupa:

1. Vektor *embedding* token keluaran di waktu sebelumnya $E[y_{t-1}]$, atau jika pelatihan di awal, digunakan E[<sos>].
2. *Hidden state* tan_0^{dec} .

Kemudian, *decoder* memprediksi token berikutnya y_t. Pada LSTM, formulanya mirip dengan persamaan di atas, hanya berbeda pada parameter yang terpisah untuk *decoder*. Setiap waktu *t*, prediksi token keluar $\hat{y}_t$ dapat menggunakan *softmax* pada Persamaan 2.12 dengan W_s dan b_s adalah parameter keluaran.

$$\hat{y}_t = arg\ max\left(Softmax\left(W_s h_t^{dec} + b_s\right)\right) \quad (2.12)$$

Untuk memperjelas konsep encoding dan decoding yang telah dijelaskan, berikut adalah contoh *input* "Bagaimana cara reset password akun sikadu?" yang telah melalui tahap normalisasi, tokenisasi, dan pemetaan indeks, menghasilkan:

Input tokens =["*bagaimana*", "*cara*", "*reset*", "*password*", "*akun*", "*sikadu*"]
Indexed : [4, 5, 6, 7, 8, 9]

2.8 Transformer

Transformer pertama kali diperkenalkan oleh Vaswani dkk. dalam publikasi berjudul "*Attention is All You Need*" pada tahun 2017, telah mengubah paradigma pemrosesan bahasa alami (NLP) dengan menggantikan model-model sekuensial tradisional seperti RNN dan CNN dengan mekanisme yang sepenuhnya berbasis pada *self-attention* (Vaswani dkk., 2017). Keunggulan utama dari arsitektur ini adalah kemampuannya untuk mengolah seluruh urutan *input* sekaligus tanpa mengandalkan rekursi, sehingga mempercepat proses pelatihan dan meningkatkan kemampuan model untuk skalabilitas. *Self-attention* memberikan fleksibilitas kepada model untuk menilai pentingnya setiap kata dalam konteks keseluruhan kalimat, yang sangat penting dalam aplikasi seperti *machine translation* dan *text generation*.

Pengembangan *transformer* telah mendorong inovasi signifikan dalam NLP, memungkinkan penciptaan model-model canggih seperti BERT, GPT, dan T5. Model-model ini memanfaatkan arsitektur transformer untuk menghasilkan representasi bahasa yang lebih dalam dan kontekstual, yang telah terbukti sangat efektif dalam berbagai tugas NLP mulai dari analisis sentimen hingga ringkasan otomatis teks (Raffel dkk., 2020). Sukses dari model-model berbasis *transformer* ini tidak hanya menandai peningkatan teknis dalam desain algoritma tetapi juga membuka jalan baru dalam pemahaman komputasi terhadap bahasa manusia.

2.9 Large Language Model

Large Language Models (LLMs) seperti GPT (*Generative Pre-trained Transformer*) dan model-model lanjutannya telah merevolusi bidang pemrosesan bahasa alami (NLP) dengan mengadopsi pendekatan berbasis pada arsitektur *transformer* yang dilatih dengan dataset sangat besar. Pendekatan ini memungkinkan model untuk memahami kompleksitas bahasa manusia dengan tingkat kedalaman tinggi dan presisi. Uniknya, LLM memiliki kemampuan untuk

melakukan *transfer learning*, di mana mereka dapat mengaplikasikan pengetahuan yang diperoleh dari satu tugas ke tugas lain tanpa perlu pemrograman ulang (Kaplan dkk., 2020).

Dalam praktiknya, LLM telah menunjukkan keunggulan dalam berbagai tugas NLP, mulai dari menjawab pertanyaan hingga menghasilkan teks yang koheren dan merangkum informasi panjang dengan akurat. Contohnya adalah GPT-3, yang memiliki kapasitas 175 miliar parameter, telah menetapkan standar baru dalam fleksibilitas dan kemampuan adaptasi dalam pemrosesan bahasa. Namun, meskipun LLM menawarkan potensi besar, LLM juga menimbulkan tantangan seperti masalah bias dan kebutuhan akan sumber daya komputasi yang sangat besar, yang memerlukan strategi yang cermat untuk memanfaatkan keunggulan dan meminimalkan risiko yang terkait (Brown dkk., 2020).

2.10 BLEU Score

BLEU (Bilingual Evaluation Understudy) Score (Papineni dkk, 2002) adalah sebuah matriks yang digunakan untuk mengetahui kualitas *output* dari sebuah mesih penerjemah dibandingkan dengan *output* aktual. *BLEU* matriks menghasilkan skor translasi antara 0 dan 1, atau dapat juga berbentuk presentase. Sederhananya, *BLEU* matriks menghitung jumlah kata yang bersesuaian dengan referensi kalimat dalam dataset translasi secara terurut (Caldarini dkk, 2022).

Perhitungan *BLEU Score* dimulai dengan menghitung presisi *N-grams* yaitu perbandingan jumlah kata yang sama antara kalimat target dan kalimat yang diprediksi dengan N adalah jumlah kata terurut yang bersesuaian. Kemudian nilai presisi dari *N-grams* di gabungkan dengan persamaan 2.2.

$$\textit{Geometric Average Precision (N)} = exp(\sum_{n-1}^{N} w_n \log p_n) \qquad (2.2)$$

dengan

N = Jumlah n-gram (1-4)

w_n = Bobot n-gram ke n (1/N)

p_n = presisi n-gram (rasio n-gram hipotesis dibanding n-gram referensi)

Selanjutnya adalah menghitung *Bravety Penalty*, yaitu faktor perbandingan jumlah kata dalam kalimat target dan hasil kalimat yang diprediksi, dihitung dengan persamaan 2.3.

$$Brevity\ Penalty = \begin{cases} 1, & jika\ c > r \\ e^{(1-\frac{r}{c})}, & jika\ c \leq r \end{cases} \quad (2.3)$$

dengan

r = jumlah kata dalam kalimat target

c = jumlah kata dalam kalimat hasil prediksi

dalam hal ini nilai *brevety penalty* tidak dapat lebih dari 1 walaupun jumlah kata dalam kalimat hasil prediksi jauh lebih banyak dibandingkan dengan jumlah kata kalimat target. Selanjutnya perhitungan *BLEU Score* didapat dengan mengalikan *Brevity Penalty* dengan *Geometric Average Precision* sesuai yang ditunjukkan pada persamaan 2.4.

$$BLEU\ Score = Brevity\ Penalty \cdot Geometric\ Average\ Precision \quad (2.4)$$

Perhitungan *BLEU Score* dilakukan pada *corpus*, bukan pada setiap kalimat dan kemudian diambil rata-rata skor untuk setiap kalimat (Doshi, 2021). Kelebihan perhitungan *BLEU Score* diantaranya adalah cepat dan mudah dimengerti, menyerupai cara manusia dalam mencocokkan atau membandingkan teks, *language independent* yaitu tidak terpaku pada satu bahasa sehingga dapat digunakan dalam berbagai macam kebutuhan NLP, dapat digunakan ketika *output* merupakan beberapa kalimat, dan dapat digunakan secara bebas sehingga memudahkan dalam membandingkan metode-metode NLP.

Selain itu, BLEU *Score* merupakan alat evaluasi yang efisien dan sering dipakai dalam penelitian NLP karena kesederhanaannya. Meskipun BLEU *Score* memiliki beberapa keterbatasan, seperti tidak mempertimbangkan arti setiap kata atau tidak dapat membedakan tingkat kepentingan kata dalam kalimat, kelebihannya dalam memberikan evaluasi yang cepat dan sederhana menjadikannya sangat berguna dalam berbagai konteks pengembangan NLP. Alat

ini tetap relevan dalam memberikan evaluasi komparatif secara efektif untuk berbagai metode dan model NL