

**IMPLEMENTASI MODEL BAHASA *OPENAI GPT-3* UNTUK
APLIKASI *TEXT CONTENT GENERATOR* BERBASIS WEB DAN
APLIKASI *MOBILE***

Skripsi

Oleh

KAIRA MILANI FITRIA



JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

2022

ABSTRAK

IMPLEMENTASI MODEL BAHASA *OPENAI GPT-3* UNTUK APLIKASI *TEXT CONTENT GENERATOR* BERBASIS WEB DAN APLIKASI MOBILE

Oleh

KAIRA MILANI FITRIA

Kemajuan teknologi telah mendorong pertumbuhan bisnis digital untuk meningkatkan teknik pemasaran produknya. Salah satu hal yang dibutuhkan saat ini adalah penggunaan teknik *copywriting* dalam konten pemasaran digital. Terdapat beberapa usaha yang mengalami kesulitan dalam menerapkan teknik *copywriting* karena belum mampu menyediakan anggaran untuk jasa *copywriter*, hal ini terjadi pada jenis usaha kecil dan menengah. Permasalahan tersebut mendorong untuk menciptakan suatu sistem yang dapat berperan sebagai *copywriter* agar menghemat anggaran biaya pada usaha skala kecil. Berdasarkan permasalahan tersebut, dilakukan penelitian penerapan *Artificial Intelligence*, dengan menggunakan model bahasa *OpenAI GPT-3* untuk membuat aplikasi bernama "*AI Caption Generator*" yang berperan sebagai mesin *copywriter* dan dapat membantu pembuatan konten tulisan. Penelitian dilakukan dengan metode *few-shot learning* pada model bahasa *GPT-3*. Hasil aplikasi dapat diakses pada website dan aplikasi berbasis android yang dapat diunduh pada Google Play Store. Kemampuan aplikasi diuji pada mesin pembuat teks berbayar, seperti *copy.ai*, *ryte.me*, dan *quillbot* dengan hasil skor *BERTscore* 86% (*precision*), 88% (*recall*), 87% (*F1-Score*), dan skor *ROUGE* 55% (*precision*), 60% (*recall*), 57% (*F1-Score*), dengan total *rating* 5,0 pada *Google Play Store* sehingga aplikasi *AI Caption Generator* memiliki kelayakan penggunaan pada berbagai kalangan.

Kata kunci : *GPT-3*, model bahasa, *copywriting*, konten digital, *text generator*

ABSTRACT

LANGUAGE MODEL IMPLEMENTATION OPENAI GPT-3 FOR APP TEXT CONTENT GENERATOR WEB BASED AND MOBILE APPLICATIONS

By

KAIRA MILANI FITRIA

Technological advances have driven the growth of digital businesses to improve their product marketing techniques. One of the things needed today is copywriting techniques in digital marketing content. Several businesses experience difficulties in implementing copywriting techniques because they have yet to be able to provide a budget for copywriting services. This occurs in small and medium businesses. These problems encourage the creation of a system that can act as a copywriter to save on budget costs for small-scale businesses. Based on these problems, research on the application of Artificial Intelligence was carried out using the OpenAI GPT-3 language model to create an application called "AI Caption Generator," which acts as a copywriter and can help create written content. The research was conducted using the few-shot learning method in the GPT-3 language model. Application results can be accessed on websites and Android-based applications downloaded from the Google Play Store. The ability of the application was tested on paid text generator engines, such as copy.ai, ryte.me, and quillbot, with a BERTscore of 86% (precision), 88% (recall), 87% (F1-Score), and a ROUGE score of 55% (precision), 60% (recall), 57% (F1-Score), with a total rating of 5.0 on the Google Play store so that the AI Caption Generator application is appropriate for use in various circles.

Keywords : GPT-3, language model, copywriting, digital content, text generator

**IMPLEMENTASI MODEL BAHASA *OPENAI GPT-3* UNTUK APLIKASI
TEXT CONTENT GENERATOR BERBASIS WEB DAN APLIKASI *MOBILE***

Oleh:

KAIRA MILANI FITRIA

Skripsi

Sebagai salah satu syarat untuk mendapat gelar

SARJANA TEKNIK

pada

Jurusan Teknik Elektro

Fakultas Teknik

Universitas Lampung



JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

2022

Judul Skripsi : IMPLEMENTASI MODEL BAHASA OPENAI GPT-3 UNTUK APLIKASI TEXT CONTENT GENERATOR BERBASIS WEB DAN APLIKASI MOBILE

Nama Mahasiswa : Kaira Milani Fitria
Pokok Mahasiswa : 1815031032
Jurusan : Teknik Elektro
Fakultas : Teknik




Herlinawati, S.T., M.T
NIP. 197103141999032001


Dr. Eng. Nining Purwasih, S.T., M.T
NIP. 19740422200122001

MENGESAHKAN

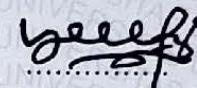
1. Tim Pengaji

Ketua : **Dr. Misfa Susanto, S.T., M.Sc.**

Sekretaris : **Dr. Ing. Melvi, S.T., M.T.**

Pengaji : **Yetti Yuniati, S.T., M.T.**


.....

2. Dekan Fakultas Teknik




Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc

NIP. 197509282001121002



Tanggal Lulus Ujian Skripsi : 16 Desember 2022

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa dalam skripsi ini tidak terdapat karya yang pernah dilakukan orang lain dan sepanjang pengetahuan saya tidak terdapat atau diterbitkan oleh orang lain, kecuali secara tertulis diacu dalam naskah ini sebagaimana yang disebutkan dalam daftar pustaka. Selain itu, saya menyatakan bahwa skripsi ini dibuat oleh saya sendiri.

Apabila pernyataan saya tidak benar, maka saya bersedia dikenai sanksi sesuai dengan hukum yang berlaku

Bandarlampung, 17 Desember 2022



Kaira Milani Fitria

NPM. 1815031032

RIWAYAT HIDUP



Penulis dilahirkan di Magetan, Jawa Timur, pada tanggal 4 Januari 2000. Penulis merupakan anak tunggal dari pasangan Bapak Suwondo dan Ibu Siti Aisah. Penulis memulai pendidikan di SDN 2 Negeri Sakti pada tahun 2006 hingga 2012, SMPN 13 Bandarlampung pada tahun 2012 hingga 2015, dan SMK SMTI Bandarlampung jurusan Kimia Industri pada tahun 2015 hingga 2018. Penulis menjadi mahasiswa Jurusan Teknik Elektro, Universitas Lampung pada tahun 2018 melalui jalur SBMPTN. Selama menjadi mahasiswa, penulis berkesempatan menjadi asisten praktikum mata kuliah Pengukuran Besaran Listrik, Fisika Dasar TI, Rangkaian Listrik, dan Pengukuran Besaran Listrik mulai tahun 2020 hingga 2022 serta tergabung dalam keanggotaan asisten Laboratorium Pengukuran Besaran Listrik. Selain itu, penulis tergabung dalam lembaga kemahasiswaan yang ada dalam Jurusan Teknik Elektro (HIMATRO) sebagai anggota Divisi Komifo pada tahun 2019 hingga 2021. Penulis melaksanakan kerja praktik (KP) pada bulan September – Desember 2021 di PT. Microsoft Indonesia dengan project Microsoft Productivity: The Modern Workplace dengan mengangkat judul laporan “Online Laboratorium Visit System using Microsoft”, dan mendapatkan sertifikasi internasional diantaranya Microsoft Office Specialist, PowerPoint Associate, Word Associate, Excel Associate, dan Power Platform Fundamentals. Pada bulan Januari – Juli 2022, penulis melaksanakan magang kampus merdeka di PT.BISA Artifisial Indonesia sebagai IoT Engineer pada proyek KIOSK Smart Campuss, KIOSK Smart Canteen, KIOSK Smart Parking, dan Tim Riset AI.

PERSEMBAHAN

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Alhamdulillah, Atas Izin Allah yang Maha Kuasa

KUPERSEMBAHKAN KARYA INI UNTUK

Bapak dan Ibu Tercinta

Suwondo dan Siti Aisah

Keluarga Besar, Dosen, Teman dan Almamater

MOTTO

Janganlah kamu bersikap lemah, dan janganlah (pula) kamu bersedih hati, padahal kamulah orang-orang yang paling tinggi (derajatnya), jika kamu orang-orang yang beriman.

(Ali-Imran ayat 139)

“Barangsiapa menempuh jalan untuk mendapatkan ilmu, Allah akan memudahkan baginya jalan menuju surga.”

(HR. Ibnu Majah no. 224)

"Every successful person loves the game. The chance to prove his worth, to excel, to win."

(Dale Carnegie – Buku “How to Win Friends and Influence People”)

"Pada dasarnya saat ini kita sedang berkembang dan kita tidak tahu batas potensi dalam diri kita. Meskipun kita sudah tahu batasnya, kita harus mengincar yang lebih tinggi."

(Daichi Sawamura – Anime “Haikyuu!”)

“Luck isn’t a result of pure coincidence. It’s an underlying element of the field that reaches only those who move by their will. If you can’t understand that, you’ve got no right to live in a competitive world.”

(Rin Itoshi – Anime “Blue Lock”)

“Bisa menghargai nyawa diri sendiri, itu juga merupakan hal yang luar biasa.”

(Armin Alert – Anime “Attack on Titan”)

SANWACANA

Segala puji bagi Allah SWT, atas limpahan nikmat-Nya yang diberikan kepada penulis sehingga dapat menyelesaikan Tugas Akhir ini. Shalawat dan salam senantiasa dicurahkan kepada Nabi Muhammad saw. suri teladan yang mampu membuka sesuatu yang terkunci, penutup dari semua yang terdahulu, penolong kebenaran dengan jalan yang benar, dan petunjuk kepada jalan-Mu yang lurus.

Tugas Akhir dengan judul “Implementasi Model Bahasa OpenAI GPT-3 untuk Aplikasi Text Content Generator berbasis Web dan Aplikasi Mobile” ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Jurusan Teknik Elektro, Fakultas Teknik, Universitas Lampung. Pada kesempatan ini, penulis mengucapkan terimakasih kepada:

1. Bapak Dr. Sofwan Effendi, M.Ed. selaku Plt Rektor Universitas Lampung.
2. Bapak Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc.. selaku Dekan Fakultas Teknik Universitas Lampung
3. Ibu Herlinawati, S.T., M.T selaku Kepala Jurusan Teknik Elektro Universitas Lampung.
4. Bapak Meizano Ardhi Muhammad, S.T.,M.T selaku Sekretaris Jurusan Teknik Elektro Universitas Lampung
5. Ibu Dr.Eng. Nining Purwasih,S.T.,M.T. selaku Kepala Prodi Teknik Elektro Universitas Lampung
6. Ibu Yetti Yuniati S.T., M.T. selaku pembimbing utama skripsi yang telah memberikan bimbingan rutin, motivasi dan arahan kepada penulis dengan baik dan ramah.
7. Bapak Misfa Susanto, S.T.,M.Sc.,Ph.D. dan Ibu Dr. Ing, Melvi, S.T., M.T. selaku penguji skripsi yang memberikan arahan untuk menyempurnakan skripsi.

8. Bapak Osea Zeboa, S.T., M.T. selaku dosen pembimbing akademik (PA) yang telah membimbing penulis mempersiapkan diri menjadi seorang Sarjana Teknik.
9. Segenap Dosen di Jurusan Teknik Elektro yang telah memberikan ilmu yang bermanfaat, wawasan, dan pengalaman bagi penulis.
10. Segenap Staff di Jurusan Teknik Elektro dan Fakultas Teknik yang telah membantu penulis baik dalam hal administrasi dan hal-hal lainnya.
11. Kepada Support System Bapak dan Ibuku tercinta yang telah memberikan dukungan doa, moril, dan materil kepada penulis, terimakasih telah menemani dan menyemangati anak tersayangnya.
12. Kepada Adik-adik asuhku yang telah memberikan doa dan semangat selama penulis menyelesaikan skripsi.
13. Rekan-rekan mahasiswa Teknik Elektro Universitas Lampung Angkatan 2018 (ELTICS'18) yang telah banyak memberi dukungan moril untuk saya.
14. Teman seperjuangan Estherina Ermi, Fani Kosasih, Maulana Fahar, dan Muhyiddin yang telah banyak membantu, berbagi pengalaman, dan memberikan saran kepada penulis dalam pembuatan skripsi ini.
15. Sahabatku Sofie Aurelia yang selalu support, doa, dan mengirim go-food agar penulis tidak kehilangan semangat dalam menyelesaikan skripsi.
16. Segenap Keluarga Besar Laboratorium Pengukuran Besaran Listrik; Bapak Qodar atas kerjasama dan nasihatnya selama studi; Rekan Asisten PBL 2018; dan adik-adik asisten.
17. Big Thanks to Nicholas Renotte as IBM Senior Data Science and his learning videos really inspired me using GPT-3, also thanks to Thomas Wolf and Kirk Borne who provided special data science learning materials which really helped me to complete my final project.

18. Semua pihak yang terlibat dalam menyelesaikan laporan Skripsi yang tidak dapat penulis sebutkan satu persatu.
19. Semua anggota keluarga besar dan kerabat yang menanyakan "Kapan skripsi-mu selesai?", yang telah meningkatkan semangat penulis untuk segera menyelesaikan skripsi.

Bandarlampung, 18 Desember 2022

Kaira Milani Fitria

DAFTAR ISI

ABSTRAK	ii
HALAMAN JUDUL	iv
LEMBAR PERSETUJUAN	v
LEMBAR PENGESAHAN	vi
SURAT PERNYATAAN	vii
RIWAYAT HIDUP	viii
PERSEMBAHAN.....	ix
MOTTO	x
SANWACANA	xi
DAFTAR ISI.....	xiv
DAFTAR GAMBAR.....	xvii
DAFTAR TABEL	xxii
DAFTAR SINGKATAN.....	xxiii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan.....	3
1.4 Batasan Masalah.....	4
1.5 Manfaat.....	4
1.6 Sistematika Penulisan.....	5
BAB II TINJAUAN PUSTAKA.....	6
2.1 Penelitian Terkait	6
2.2 <i>Copywriting</i>	15
2.2.1 Kalimat Kreatif	16
2.3 <i>Artificial Intelligence (AI)</i>	17
2.3.1 Konsep AI dalam <i>Copywriting</i>	19
2.4 <i>Machine Learning (ML)</i>	19
2.5 <i>Deep Learning (DL)</i>	21
2.6 <i>Recurrent Neural Networks (RNN)</i>	23
2.6.1 Data Teks Sekuential	23
2.6.2 Model Bahasa	24
2.6.3 Implementasi RNN.....	26

2.7 Mekanisme Atensi dan <i>Transformers</i>	26
2.7.1 Mekanisme Atensi (<i>Attention Mechanism</i>).....	27
2.7.2 <i>Self-attention</i>	28
2.7.3 <i>Transformer</i>	28
2.7.4 <i>Pre-Training</i> Skala Besar dengan <i>Transformer</i>	30
2.8 <i>Natural Language Processing (NLP)</i>	35
2.9 Pemodelan Bahasa <i>Transformer</i>	35
2.10 <i>Generative Pre-trained Transformer (GPT)</i>	37
2.10.1 Perbedaan <i>GPT</i> dan <i>BERT</i>	40
2.10.2 <i>Training GPT-3</i>	42
2.10.3 <i>Perplexity GPT-3</i>	43
2.11 <i>Application Programming Interface (API)</i>	45
2.12 <i>Python</i>	46
2.13 <i>Flask</i>	47
2.14 <i>HyperText Markup Language (HTML)</i>	47
2.15 <i>JavaScript (JS)</i>	48
2.16 <i>Cascading Style Sheets (CSS)</i>	48
2.17 <i>PythonAnywhere</i>	49
2.18 <i>Website & Mobile Application</i>	49
2.19 <i>Google Playstore</i>	50
2.20 Indikator Keberhasilan	50
2.20.1 <i>Blackbox Testing</i>	51
2.20.2 <i>ROUGE (Recall-Oriented Understudy for Gisting Evaluation)</i>	51
a. <i>Recall</i>	53
b. <i>Precision</i>	54
c. <i>F1-score</i>	55
2.20.3 <i>BERTscore</i>	55
a. <i>Recall</i>	58
b. <i>Precision</i>	59
c. <i>F1-score</i>	59
2.20.4 <i>Rating Aplikasi Playstore</i>	60
BAB III METODOLOGI PENELITIAN	61
3.1 Tempat dan Waktu Penelitian	61
3.2 Alat dan Bahan	61
3.3 Metode Penelitian.....	62
3.4 Diagram Alir Penelitian.....	67

BAB IV HASIL DAN PEMBAHASAN	84
4.1 <i>Training GPT-3</i>	84
4.2 Perancangan Aplikasi	92
4.3 Pemrograman Aplikasi	95
4.4 <i>Deployment Website</i>	99
4.5 Pemrograman Aplikasi Android.....	110
4.6 Pengujian	116
4.6.1 <i>Blackbox Testing</i>	116
4.6.2 Matriks Evaluasi	120
4.6.2.a <i>BERTscore</i>	121
4.6.2.b <i>ROUGE</i>	143
4.6.3 <i>Rating Google Play store</i>	151
BAB V KESIMPULAN	154
5.1 Kesimpulan.....	154
5.2 Saran	154
DAFTAR PUSTAKA	155

DAFTAR GAMBAR

Gambar 2.1. <i>State of the Art / Roadmap</i> Penelitian	14
Gambar 2.2. Pengembangan <i>AI</i> hingga terbentuknya <i>GPT-3</i>	18
Gambar 2.3 Korelasi antara <i>AI</i> , <i>ML</i> , dan <i>DL</i>	22
Gambar 2.4 Arsitektur <i>Transformer</i>	29
Gambar 2.5 <i>Pre-training GPT</i> dengan pemodelan bahasa	31
Gambar 2.6 <i>Zero-shot</i> , <i>one-shot</i> , <i>few-shot learning</i> dengan model bahasa	41
Gambar 2.7 Performa keseluruhan <i>GPT-3</i> untuk <i>benchmark</i> akurasi	33
Gambar 2.8 Performa model bahasa <i>transformator</i>	34
Gambar 2.9 Pelatihan model bahasa <i>Transformer</i> yang sedang berjalan	34
Gambar 2.10 Prediksi bahasa	36
Gambar 2.11 Model bahasa <i>pre-trained</i>	36
Gambar 2.12 Proses kerja model bahasa <i>pre-trained</i>	37
Gambar 2.13 Pembelajaran dalam model bahasa <i>pre-trained</i>	37
Gambar 2.14 Arsitektur <i>Transformer</i>	38
Gambar 2.15 Arsitektur <i>GPT-3</i>	38
Gambar 2.16 Blok <i>decoder Transformer</i> dan <i>GPT</i>	39
Gambar 2.17 Perbedaan <i>GPT</i> dan <i>BERT</i>	41
Gambar 2.18 Mekanisme <i>masked self-attention</i>	41
Gambar 2.19 Perbandingan akurasi proses <i>training</i> model <i>GPT-3</i>	42
Gambar 2.20 Contoh <i>few-shot learning</i> dalam penerjemahan bahasa <i>GPT-3</i>	43
Gambar 2.21 Uji <i>perplexity</i> pada beberapa model bahasa	44
Gambar 2.22 Jumlah parameter beberapa model bahasa	45
Gambar 2.23 <i>GPT-3</i> sebagai model bahasa dengan parameter terbesar	45
Gambar 2.24 Cara kerja <i>API</i>	46
Gambar 2.25 Hubungan penggunaan <i>JavaScript</i> dan <i>CSS</i> dalam <i>HTML</i>	48
Gambar 2.26 Referensi metrik evaluasi	51
Gambar 2.27 Contoh penghitungan <i>recall</i> dalam pemodelan bahasa.....	54
Gambar 2.28 Contoh penghitungan <i>precision</i> dalam pemodelan bahasa	54
Gambar 2.29 Contoh penghitungan <i>F1-score</i> dalam pemodelan bahasa.....	55

Gambar 2.30 Arsitektur <i>BERTscore</i>	57
Gambar 3.1 <i>Timeline</i> Penelitian.....	61
Gambar 3.2 Konsep Metodologi Penelitian.....	63
Gambar 3.3 Rancangan implementasi API <i>GPT-3</i> pada aplikasi	66
Gambar 3.4 Diagram Alir Penelitian	69
Gambar 3.5 Diagram alir tahap identifikasi.....	72
Gambar 3.6 Diagram alir tahap desain aplikasi	73
Gambar 3.7 <i>UML Activity Diagram</i>	74
Gambar 3.8 Pemrosesan token kata dalam <i>GPT-3</i>	75
Gambar 3.9 Diagram Alir tahap Desain Aplikasi (1)	77
Gambar 3.10 Diagram Alir tahap Desain Aplikasi (2)	79
Gambar 3.11 Diagram Alir tahap Pengujian dan Laporan.....	81
Gambar 3.12 Diagram blok pengujian <i>BERTscore</i>	83
Gambar 3.13 Diagram blok pengujian <i>ROUGE</i>	83
Gambar 4.1 <i>Setting playground GPT-3</i>	86
Gambar 4.2 Hasil <i>training</i> pertama (<i>zero-shot learning</i>).....	87
Gambar 4.3 Hasil <i>training</i> kedua (<i>one-shot learning</i>)	88
Gambar 4.4 Hasil <i>training</i> ketiga (<i>few-shot learning</i>) <i>product description</i>	88
Gambar 4.5 Hasil <i>training</i> (<i>few-shot learning</i>) <i>job description</i>	89
Gambar 4.6 Hasil <i>training</i> (<i>few-shot learning</i>) <i>social media captions</i>	89
Gambar 4.7 Hasil <i>training</i> (<i>few-shot learning</i>) <i>email template</i>	90
Gambar 4.8 Hasil <i>training</i> (<i>few-shot learning</i>) <i>advertisement</i>	90
Gambar 4.9 Hasil <i>training</i> (<i>few-shot learning</i>) <i>business pitch</i>	90
Gambar 4.10 Hasil <i>training</i> (<i>few-shot learning</i>) <i>youtube idea</i>	91
Gambar 4.11 Hasil <i>training</i> (<i>few-shot learning</i>) <i>video description</i>	91
Gambar 4.12 Hasil <i>training</i> (<i>few-shot learning</i>) <i>AI sentence</i>	91
Gambar 4.13 Hasil <i>training</i> (<i>few-shot learning</i>) <i>paragraph summary</i>	92
Gambar 4.14 Hasil <i>training</i> (<i>few-shot learning</i>) <i>key points</i>	92
Gambar 4.15 Hasil <i>training</i> (<i>few-shot learning</i>) <i>keyword extractor</i>	92
Gambar 4.16 <i>API Key GPT-3</i>	93
Gambar 4.17 Desain <i>element icon</i> untuk setiap fitur	93
Gambar 4.18 Elemen Grafis Aplikasi AI Caption Generator	94

Gambar 4.19 Desain <i>layout</i> dengan <i>Figma</i>	95
Gambar 4.20 Setting navigation dengan <i>Figma</i>	95
Gambar 4.21 Tampilan <i>prototype</i> desain <i>Figma</i>	96
Gambar 4.22 Pengunduhan komponen grafis <i>layout</i> dengan <i>Figma</i>	96
Gambar 4.23 Kode program <i>frontend</i>	97
Gambar 4.24 Kode program <i>backend</i>	98
Gambar 4.25 <i>API key</i> yang ada dalam <i>config.py</i>	98
Gambar 4.26 Setting parameter <i>GPT-3</i> dalam <i>aicontent.py</i>	99
Gambar 4.27 <i>Running</i> program di <i>localhost</i>	99
Gambar 4.28 <i>Testing</i> sistem aplikasi web	100
Gambar 4.29 <i>Login PythonAnywhere</i>	101
Gambar 4.30 Menentukan nama <i>domain</i> untuk aplikasi <i>website</i>	101
Gambar 4.31 Menentukan <i>framework python</i> (<i>Flask</i>)	102
Gambar 4.32 Menentukan versi <i>python</i> yang digunakan	102
Gambar 4.33 Membuat <i>path folder</i> program dalam <i>server</i>	102
Gambar 4.34 Konfigurasi aplikasi web	103
Gambar 4.35 Upload <i>file zip</i> ke <i>server</i>	103
Gambar 4.36 Akses <i>bash console</i> untuk <i>unzip</i> folder yang di- <i>upload</i>	103
Gambar 4.37 <i>Website</i> berhasil di <i>deploy</i> ke <i>server</i>	104
Gambar 4.38 Alamat web AI Caption Generator	104
Gambar 4.39 Segmen <i>About Projects</i> di website AI Caption Generator	104
Gambar 4.40 Segmen <i>Preview</i> 12 fitur AI Caption Generator	105
Gambar 4.41 Segmen “ <i>Features</i> ” pada website AI Caption Generator	106
Gambar 4.42 Hasil fitur “ <i>Product Description</i> ” AI Caption Generator	107
Gambar 4.43 Hasil fitur “ <i>Job Description</i> ” AI Caption Generator	107
Gambar 4.44 Hasil fitur “ <i>Social Media Captions</i> ” AI Caption Generator	107
Gambar 4.45 Hasil fitur “ <i>Advertisement</i> ” AI Caption Generator	107
Gambar 4.46 Hasil fitur “ <i>Email Templates</i> ” AI Caption Generator	108
Gambar 4.47 Hasil fitur “ <i>AI Sentence</i> ” AI Caption Generator	108
Gambar 4.48 Hasil fitur “ <i>Business Pitch</i> ” AI Caption Generator	108
Gambar 4.49 Hasil fitur “ <i>Youtube Idea</i> ” AI Caption Generator	109
Gambar 4.50 Hasil fitur “ <i>Video Description</i> ” AI Caption Generator	109

Gambar 4.51 Hasil fitur “ <i>Paragraph Summary</i> ” AI Caption Generator	109
Gambar 4.52 Hasil fitur “ <i>Key Points</i> ” AI Caption Generator.....	110
Gambar 4.53 Hasil fitur “ <i>Keyword Extractor</i> ” AI Caption Generator	110
Gambar 4.54 Pemilihan <i>activity</i> pada android <i>studio</i>	111
Gambar 4.55 Pembuatan nama aplikasi dan bahasa pemrograman android.....	112
Gambar 4.56 Kode program aplikasi android AI Caption Generator	112
Gambar 4.57 Hasil <i>testing</i> aplikasi pada <i>device</i> android	113
Gambar 4.58 Desain <i>banner</i> aplikasi untuk dirilis ke <i>Google Play store</i>	114
Gambar 4.59 Akun <i>developer</i> pada <i>Google Play Console</i>	115
Gambar 4.60 <i>Log activity</i> akun <i>developer Google Play Console</i>	115
Gambar 4.61 <i>Release overview</i> aplikasi pada <i>Google Play Console</i>	116
Gambar 4.62 Aplikasi AI Caption Generator pada <i>Google Play store</i>	116
Gambar 4.63 Hasil deskripsi produk web <i>open.ai</i>	122
Gambar 4.64 Hasil deskripsi produk AI Caption Generator.....	122
Gambar 4.65 Penghitungan <i>BERTscore</i> pada fitur <i>Product Description</i>	123
Gambar 4.66 Hasil deskripsi pekerjaan web <i>rytr.me</i>	124
Gambar 4.67 Hasil deskripsi pekerjaan AI Caption Generator.....	125
Gambar 4.68 Penghitungan <i>BERTscore</i> pada fitur <i>Job Description</i>	125
Gambar 4.69 Hasil <i>caption</i> AI Caption Generator.....	126
Gambar 4.70 Hasil <i>caption</i> web <i>Ryte.me</i>	127
Gambar 4.71 Penghitungan <i>BERTscore</i> pada fitur <i>Social Media Captions</i>	127
Gambar 4.72 Hasil <i>template email</i> pada web <i>Ryte.me</i>	129
Gambar 4.73 Hasil <i>template email</i> AI Caption Generator.....	129
Gambar 4.74 Penghitungan <i>BERTscore</i> pada fitur <i>Email Template</i>	130
Gambar 4.75 Hasil kalimat iklan pada web <i>Ryte.me</i>	131
Gambar 4.76 Hasil kalimat iklan AI Caption Generator.....	131
Gambar 4.77 Penghitungan <i>BERTscore</i> pada fitur <i>Advertisement</i>	132
Gambar 4.78 Hasil promosi bisnis pada web <i>Rytr.me</i>	133
Gambar 4.79 Hasil promosi bisnis AI Caption Generator	133
Gambar 4.80 Penghitungan <i>BERTscore</i> pada fitur <i>Business Pitch</i>	134
Gambar 4.81 Hasil ide <i>youtube</i> pada web <i>Rytr.me</i>	135
Gambar 4.82 Hasil ide <i>youtube</i> AI Caption Generator	136

Gambar 4.83 Penghitungan <i>BERTscore</i> pada fitur <i>Youtube Idea</i>	136
Gambar 4.84 Hasil deskripsi video pada web <i>Ryte.me</i>	138
Gambar 4.85 Hasil deskripsi video AI Caption Generator	138
Gambar 4.86 Penghitungan <i>BERTscore</i> pada fitur <i>Video Description</i>	139
Gambar 4.87 Hasil <i>key points</i> AI Caption Generator	140
Gambar 4.88 Penghitungan <i>BERTscore</i> pada fitur <i>Key Points</i>	141
Gambar 4.89 Hasil rangkuman pada AI Caption Generator	142
Gambar 4.90 Hasil rangkuman pada web <i>quillbot</i>	143
Gambar 4.91 Penghitungan <i>BERTscore</i> pada fitur <i>Paragraph Summary</i>	143
Gambar 4.92 Penghitungan <i>ROUGE</i> pada fitur <i>Paragraph Summary</i>	145
Gambar 4.93 Hasil <i>sentence corrector</i> pada AI Caption Generator	146
Gambar 4.94 Penghitungan <i>ROUGE</i> pada fitur <i>AI Sentence Corrector</i>	147
Gambar 4.95 Hasil <i>keyword extractor</i> pada <i>quillbot</i>	148
Gambar 4.96 Hasil <i>keyword extractor</i> pada AI Caption Generator.....	149
Gambar 4.97 Penghitungan <i>ROUGE</i> pada fitur <i>Keyword Extractor</i>	150
Gambar 4.98 <i>Performance over time</i> aplikasi AI Caption Generator.....	153
Gambar 4.99 Aplikasi AI Caption Generator pada <i>Google Play store</i>	153
Gambar 4.100 <i>User review</i> aplikasi AI Caption Generator	154

DAFTAR TABEL

Tabel 2.1 Penelitian Terkait	10
Tabel 4.1 Pengujian <i>Blackbox</i>	117
Tabel 4.2 Hasil <i>BERTscore</i> pada fitur <i>Product Description</i>	123
Tabel 4.3 Hasil <i>BERTscore</i> pada fitur <i>Job Description</i>	126
Tabel 4.4 Hasil <i>BERTscore</i> pada fitur <i>Social Media Captions</i>	128
Tabel 4.5 Hasil <i>BERTscore</i> pada fitur <i>Email Template</i>	130
Tabel 4.6 Hasil <i>BERTscore</i> pada fitur <i>Advertisement</i>	132
Tabel 4.7 Hasil <i>BERTscore</i> pada fitur <i>Business Pitch</i>	135
Tabel 4.8 Hasil <i>BERTscore</i> pada fitur <i>Youtube Idea</i>	137
Tabel 4.9 Hasil <i>BERTscore</i> pada fitur <i>Video Description</i>	139
Tabel 4.10 Hasil <i>BERTscore</i> pada fitur <i>Key Points</i>	141
Tabel 4.11 Hasil <i>BERTscore</i> pada fitur <i>Paragraph Summary</i>	144
Tabel 4.12 Hasil <i>ROUGE</i> pada fitur <i>Paragraph Summary</i>	145
Tabel 4.13 Data uji fitur <i>AI Sentence Corrector</i>	146
Tabel 4.14 Hasil <i>ROUGE</i> pada fitur <i>AI Sentence Corrector</i>	147
Tabel 4.15 Hasil <i>ROUGE</i> pada fitur <i>Keyword Extractor</i>	150
Tabel 4.16 Hasil pengujian <i>ROUGE</i> pada AI Caption Generator	151
Tabel 4.17 Hasil pengujian <i>BERTscore</i> pada AI Caption Generator.....	151

DAFTAR SINGKATAN

AI	: Artificial Intelligence
ANN	: Artificial Neural Network
API	: Application Programming Interface
ASCII	: American Standard Code for Information Interchange
BART	: Bidirectional Auto-Regressive <i>Transformers</i>
<i>BERT</i>	: Bidirectional <i>Encoder</i> Representations from <i>Transformers</i>
BLEU	: Bilingual Evaluation Understudy
CLM	: Causal Language Modeling
CMD	: Command <i>Prompt</i>
CNN	: Convolutional Neural Network
CSS	: Cascading Style Sheets
DL	: Deep <i>Learning</i>
<i>GPT</i>	: Generative Pre-trained <i>Transformer</i>
HTML	: HyperText Markup Language
JS	: JavaScript
LCS	: Longest Common Subsequence
LM	: Language Model
LSTM	: Long Short Term Memory
ML	: Machine <i>Learning</i>
NLG	: Natural Language Generation
NLP	: Natural Language Processing
NMT	: Neural Machine Translation
OS	: <i>Operating</i> System
PaaS	: Platform as a Service
PSBB	: Pembatasan Sosial Berskala Besar
<i>RNN</i>	: Recurrent Neural Network

<i>ROUGE</i>	: <i>Recall-Oriented Understudy for Gisting Evaluation</i>
SDK	: Software Development Kit
SEO	: Search Engine Optimization
T5	: <i>Text-to-Text-Transfer-Transformer Model</i>
UI	: <i>User Interface</i>
UMKM	: Usaha Mikro Kecil dan Menengah
UML	: Unified Modeling Language
URL	: Uniform Resource Locator

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Natural Language Processing (NLP) merupakan pengembangan teknologi *Artificial Intelligence (AI)* yang berfokus pada peningkatan kemampuan komputer untuk memahami teks dan kata-kata yang diucapkan. *NLP* fokus untuk memproses komputasi bahasa manusia. Pengembangan *NLP* menghasilkan pengembangan model bahasa berdasarkan jaringan saraf (*neural networks*). Pada tahun 2017, diperkenalkan *Transformer*, sebagai era baru untuk model bahasa [1]. Model bahasa *pre-trained* yang paling representatif seperti *Generative Pre-trained Transformer (GPT)-2 OpenAI* serta *GPT-3* dibentuk diatas arsitektur ini. *GPT-3* adalah model *NLP auto-regresive* yang dapat melakukan berbagai tugas seperti menanggapi pertanyaan, meringkas, mengurai teks, terjemahan, dan klasifikasi. Model bahasa ini melibatkan masukan beberapa teks sebagai *prompt* dan *GPT-3* dengan keluaran teks sesuai dengan *prompt*. *GPT-3* melakukan tugas baru hanya berdasarkan contoh (*few-shot learning*). Pengembangan *AI* dapat dikaitkan di Indonesia guna mengoptimalkan bisnis digital yang kreatif dalam membangun identitas *brand*. Adanya pandemi *COVID-19* berdampak pada pengembangan usaha, hanya sekitar 44% Usaha Mikro Kecil dan Menengah (UMKM) di Indonesia berhasil bergabung dan beradaptasi ke pasar *online* [2]. Saat program Pembatasan Sosial Berskala Besar (PSBB) pada pandemi *COVID-19*, *e-commerce* berkembang pesat hingga memunculkan persaingan pada proses pemasarannya. Hal tersebut dapat diatasi dengan *menunjang* konten penjualan produk yang baik. Permasalahan lainnya yaitu menaikkan *insight* penjualan *online* dan meningkatkan reputasi produk yang dapat diselesaikan dengan menerapkan teknik *copywriting*.

Copywriting berkaitan dengan teknik promosi suatu produk dengan menyampaikan pesan dari perusahaan kepada konsumen. *Copywriter* bertugas melakukan analisis dan riset sebelum *menulis* materi atau *caption*, penetapan kata kunci, serta penyusunan kalimat yang menarik. Pembuatan aplikasi dengan memanfaatkan kemampuan *AI* sebagai penyedia ide kalimat dapat memudahkan *copywriter*. Implementasi tersebut dapat mengkatkan produktivitas bisnis hingga 40% [3], meningkatkan produktivitas tenaga kerja, dan mengoptimalkan efisiensi bisnis sebesar 67%, mengotomatiskan komunikasi sebesar 70% [4]. Penggunaan *AI* oleh The Press Association dapat menghasilkan 30.000 berita lokal dalam sebulan, serta Washington Post menggunakan alat *Natural Language Generation (NLG)* internalnya untuk membuat artikel berita dan postingan media sosial [5]. Hal ini menunjukkan bahwa pemanfaatan *AI* berupa model bahasa sudah banyak diterapkan dan membantu beberapa usaha bisnis digital manapun untuk menghemat anggaran, serta dapat mengasah kemampuan *copywriting* diberbagai kalangan.

Pada penelitian ini dilakukan pembuatan aplikasi yang bernama “*AI Caption Generator*” yang berfungsi menghasilkan susunan kalimat yang menarik dan variatif sehingga dapat dikustomisasi sesuai dengan kebutuhan. Aplikasi ini dibuat dengan memanfaatkan model bahasa yang dikembangkan oleh *OpenAI* yaitu *GPT-3* atau *Generative Pre-trained Transformer* generasi 3 terbaru saat ini, yang merupakan model bahasa *autoregresif* dengan *deep learning* untuk menghasilkan teks seperti manusia. Beberapa fitur yang akan disediakan dalam aplikasi ini yaitu fitur deskripsi produk, deskripsi pekerjaan, *caption* media sosial, *template email*, iklan, ide bisnis, ide judul video *youtube*, deskripsi video, *sentence corrector*, rangkuman, *key point*, dan pembuat *keyword* dari suatu teks. Aplikasi ini memiliki target pasar seperti para *copywriter*, pelaku bisnis, UMKM, mahasiswa, masyarakat umum, *sales*, admin, dan lain-lain. Pengembangan aplikasi ini kedepannya diharapkan akan selalu ada penambahan fitur yang bermanfaat, dan jika aplikasi sudah mendapat *user* yang cukup banyak akan dikembangkan dengan menambahkan fitur premium yang dapat menyimpan teks yang dihasilkan oleh aplikasi sehingga dapat diakses kembali dan dapat dilakukan pengeditan pada teks yang dihasilkan oleh aplikasi.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian tugas akhir yang dibahas ialah sebagai berikut:

1. Bagaimana cara membuat aplikasi web dan aplikasi android yang bisa menjadi mesin *copywriter* untuk meningkatkan efisiensi dari segi teknikal dan bisnis.
2. Bagaimana membuat sistem yang menghasilkan susunan kalimat yang menarik dengan memanfaatkan *Application Programming Interface (API)* dari *OpenAI GPT-3*.
3. Bagaimana membuat aplikasi yang mampu mendeskripsikan produk, jenis jasa dan pekerjaan dengan bahasa yang baik dan menarik.
4. Bagaimana membuat aplikasi yang mampu menghasilkan kalimat sesuai dengan gaya bahasa sehari-hari untuk *caption* suatu konten dalam sosial media.
5. Bagaimana membuat aplikasi yang mampu menghasilkan *template email marketing* untuk menawarkan suatu produk atau jasa sesuai dengan yang diinginkan pengguna.
6. Bagaimana membuat fitur aplikasi yang mampu menghasilkan kalimat iklan untuk suatu produk atau jasa yang bersifat persuasif.
7. Bagaimana membuat fitur aplikasi yang mampu menghasilkan deskripsi suatu judul video agar sesuai konteks.
8. Bagaimana membuat fitur aplikasi yang mampu memperbaiki struktur kalimat agar lebih baik dan sesuai dengan *grammar*.
9. Bagaimana membuat fitur aplikasi yang mampu meringkas suatu paragraf agar lebih singkat dan dimuat dalam poin-poin penting.
10. Bagaimana membuat fitur aplikasi yang mampu menghasilkan *keyword* dari suatu teks untuk *keyword* tagar postingan iklan yang akan dibuat.

1.3 Tujuan

Tujuan dari penelitian skripsi ini adalah sebagai berikut:

1. Membuat aplikasi *website* dan aplikasi android yang bisa menjadi mesin *copywriter* untuk meningkatkan efisiensi dari segi teknikal dan bisnis.

2. Membuat sistem yang menghasilkan susunan kalimat yang menarik dengan memanfaatkan *API* dari *OpenAI GPT-3*.
3. Membuat aplikasi yang mampu menghasilkan kalimat : deskripsi produk dan pekerjaan, *caption* sosial media, *template email marketing*, teks iklan persuasif, deskripsi video, memperbaiki *grammar*, ringkasan, *key points*, dan *keyword*.

1.4 Batasan Masalah

Batasan Masalah dalam penelitian skripsi ini meliputi :

1. Hasil teks dari aplikasi yang dibuat sepenuhnya merupakan hasil *training* dari *OpenAI GPT-3*.
2. Teks kalimat yang dihasilkan hanya dalam bentuk Bahasa Inggris.
3. Metode yang digunakan adalah metode *few-shot learning* pada model bahasa *GPT-3*.
4. Aplikasi AI Caption Generator dijalankan pada *browser* web dan perangkat android.
5. *Website text generator* lain yang digunakan sebagai referensi pengujian yaitu *copy.ai*, *ryte.me*, dan *quillbot*.

1.5 Manfaat

Manfaat dari penelitian dari skripsi ini adalah sebagai berikut :

1. Memudahkan berbagai pihak dalam hal pekerjaan *copywriting* ringan dengan dengan bantuan teknologi *AI* melalui aplikasi hasil penelitian ini.
2. Penggunaan model bahasa untuk membuat kalimat konten diharapkan dapat menjadi solusi untuk mengatasi kesulitan pembuatan konten tulisan dalam berbagai kebutuhan.

1.6 Sistematika Penulisan

Adapun sistematika penulisan Laporan Skripsi ini adalah sebagai berikut:

I. PENDAHULUAN

Pendahuluan memuat latar belakang, tujuan penelitian, perumusan masalah, batasan masalah, manfaat penelitian, dan sistematika penulisan.

II. TINJAUAN PUSTAKA

Tinjauan pustaka memaparkan beberapa teori pendukung dan referensi materi yang diperoleh dari berbagai sumber mengenai mengenai penggunaan model bahasa untuk melakukan beberapa tugas *Natural Language Processing* dengan memanfaatkan model bahasa *GPT-3*.

III. METODOLOGI PENELITIAN

Metodologi penelitian memuat waktu dan tempat, alat dan bahan, metode penelitian, dan diagram alir penelitian.

IV. HASIL DAN PEMBAHASAN

Hasil dan pembahasan memuat perancangan dan analisis dari hasil pengujian, pembahasan hasil penelitian, dan perhitungan kinerja metode yang diusulkan.

V. PENUTUP

Memuat kesimpulan dari penelitian yang dilakukan dan saran yang didasarkan pada hasil data mengenai perbaikan dan pengembangan lebih lanjut agar didapatkan hasil lebih baik.

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Sehubungan dengan penelitian yang dilakukan, referensi terhadap penelitian sebelumnya sangat penting untuk menghindari plagiarisme atau duplikasi penelitian sebelumnya, hal ini juga dimaksudkan sebagai bahan kontribusi penelitian agar tema penelitian ini terus berkembang. Berikut beberapa ulasan penelitian terdahulu yang telah dilakukan mengenai metode yang digunakan dan hasil yang diperoleh.

Penelitian berjudul "*Improving Language Understanding by Generative Pre-Training*" dilakukan oleh Radford pada tahun 2018 yang menggunakan *unsupervised pre-training* dan *supervised fine-tuning* dengan model bahasa *generative pre-training* pada beragam teks non-label dengan *dataset BooksCorpus* dan *ELMo*. Hasil penelitian ini adalah *absolute improvement* 8.9% pada *commonsense reasoning*, 5.7% untuk menjawab pertanyaan (*RACE*), dan 1.5% pada keterlibatan tekstual/*textual entailment* [6].

Penelitian berjudul "*A Text Abstraction Summary Model Based on BERT Word Embedding and Reinforcement Learning*" dilakukan oleh Q.Wang pada tahun 2019 yaitu membuat model *extractive-abstractive* dengan mengombinasikan model bahasa *Bidirectional Encoder Representations from Transformers (BERT)* untuk *word embedding* dan *reinforcement learning*. Hasil skor *Recall-Oriented Understudy for Gisting Evaluation (ROUGE)* yang tertinggi yaitu skor *ROUGE-1*,

ROUGE-2, *ROUGE-L*, dan *R-AVG* masing-masing meningkat 1,07%, 2,46%, 0,95%, dan 1,46% [7].

Penelitian berjudul "*Customizable text generation via conditional text generative adversarial network*" dilakukan oleh Chen pada tahun 2019 yang menciptakan model *text generator* yang disebut *conditional text generative adversarial network* (*CTGAN*) dengan hasil akurasi, *recall*, dan *F1-score* masing-masing yaitu 1.0; 1.0; 1.0 untuk *dataset Yelp*, 0.75; 0.71; 0.73 untuk *negative review* dan 0.72; 0.76; 0.74 untuk *positive review* pada *dataset film review*, dan 1.00; 1.00; 1.00 pada *dataset Amazon* [8].

Penelitian berjudul "*Text summarization using transfer learning, Extractive and abstractive summarization using BERT and GPT-2 on news and podcast data*" dilakukan oleh Riisne pada tahun 2019 dengan model bahasa *BERT* dan *GPT-2* dengan hasil matriks *ROUGE F1-score* sebesar 16.28; 4.09; 12.80 pada *ROUGE-1*, *ROUGE-2*, *ROUGE-L* pada *dataset podcast*. Kemudian nilai *F1-score* 39.38; 14.64; 35.38 pada *ROUGE-1*, *ROUGE-2*, *ROUGE-L* untuk *dataset Convolutional Neural Network (CNN)* [9].

Penelitian berjudul "*Automatic Text Summarization of COVID-19 Medical Research Articles using BERT and GPT-2*" dilakukan oleh B.Tan pada tahun 2020 yang menggunakan model bahasa *BERT* dan *GPT-2* untuk membuat *dataset* penelitian *COVID-19* dengan total lebih dari 59.000 artikel ilmiah, dengan 47.000 teks lengkap tentang *COVID-19* atau penyakit terkait. Hasil evaluasi model dengan skor *ROUGE* sebesar 60% [10].

Penelitian berjudul "*BERT Transformer model for Detecting Arabic GPT2 AutoGenerated Tweets*" dilakukan oleh Fouzi pada tahun 2020 untuk pendekripsi tulisan arab dengan *GPT-2 Small Arabic* untuk menghasilkan kalimat bahasa Arab yang diuji. Hasil evaluasi model ini adalah skor akurasi hingga 98.7% [11].

Penelitian berjudul "*Deep Learning of a Pre-trained Language Model's Joke Classifier Using GPT-2*" dilakukan oleh Nur Arifin pada tahun 2021 yaitu membuat klasifikasi teks humor dengan *GPT-2* dengan metode *fine-tuning BERT* dan *dataset Short Jokes (Moudgil)* dan *WMT162*. Hasil penelitian ini adalah skor akurasi, presisi, *recall*, *F1-score* sebesar 0,983; 0,953; 0,978; 0,964 [12].

Penelitian berjudul "*Medically Aware GPT-3 as a Data Generator for Medical Dialogue Summarization*" dilakukan oleh Bharath pada tahun 2021 dengan *GPT-3* sebagai *backbone* dalam algoritma yang dibuat. Hasil penelitian ini adalah model *PEGASUS* dan *DRSUM* dengan skor *ROUGE-L* 62.45 dan 53.39 [13].

Penelitian berjudul "*Automatic Title Generation for Text with Pre-trained Transformer Language Model*" dilakukan oleh Prakhar pada tahun 2021 menggunakan *GPT-2* untuk membuat *Automatic Title Generator* yang dilatih dengan *dataset* dari *arXiv* dengan hasil penelitian skor *ROUGE-1*, *ROUGE-2*, *ROUGE-L* adalah 0.376, 0.188, 0.336, skor *BLEU-1*, *BLEU-2*, *BLEU-3*, *BLEU-4* yaitu 0.358, 0.250, 0.209, 0.124 yang tertinggi dibandingkan model lain seperti *PREFIX*, *TextRank*, *Bi-GRU w/ Attention*, dan *Zero-Shot GPT-2* [14].

Penelitian berjudul "*Prompt scoring system for dialogue summarization using GPT-3*" dilakukan oleh George pada tahun 2021 menggunakan *API GPT-3* untuk membuat sistem penilaian pada performa *few-shot training*. Hasil penelitian ini adalah rata-rata *ROUGE* untuk *setup GPT-3+SS* sebesar 0.3478 (*ROUGE-I*), 0.3296 (*ROUGE-L*), dan *setup GPT-3+random* 0.1683 (*ROUGE-I*), 0.2391 (*ROUGE-L*) [15].

Penelitian berjudul “*GPT3-to-plan: Extracting plans from text using GPT-3*” dilakukan oleh Alberto pada tahun 2021 menggunakan *GPT-3* untuk ekstraksi teks menjadi bentuk list kegiatan dengan metode *hosting GPT-3* secara *online*. Hasil penelitian ini adalah skor *F1-score* sebesar 80% [16].

Penelitian berjudul "*Abstractive Review Summarization based on Improved Attention Mechanism with Pointer Generator Network Model*" dilakukan oleh Shobana pada tahun 2021 yaitu menghasilkan ringkasan ulasan komprehensif dengan *dataset* ulasan *Amazon*. Hasil penelitian ini adalah skor *ROUGE-1*, *ROUGE-2*, *ROUGE-L* sebesar 41.2; 22.4; 37.1 [17].

Penelitian berjudul "*Sentence Augmentation for Language Translation Using GPT-2*" dilakukan oleh Ranto pada tahun 2021 menggunakan *GPT-2* untuk menggenerasi data tambahan untuk *Neural Machine Translation (NMT)*. Hasil penelitian yaitu skor *BLEU* sebesar 27.50; 24.12; 24.12 untuk *dataset Tatoeba En-Ja, WMT14 En-De, WMT18 En-Ch* [18].

Penelitian berjudul "*Detecting Hate Speech with GPT-3*" dilakukan oleh Chiu pada tahun 2022 menggunakan *GPT-3* untuk mengidentifikasi ujaran kebencian dan pengklasifikasian teks *zero-, one-,* dan *few-shot learning* dengan *dataset online haTe speeCH detectiOn dataset (ETHOS)*. Hasil penelitian ini adalah rata-rata akurasi dan *F1-score* sebesar 56%; 70% untuk *zero-shot learning*, 55%; 55% untuk *one-shot learning*, 67%; 62% untuk *few-shot learning* [19].

Penelitian berjudul "*Hate Speech Detection using OpenAI and GPT-3*" dilakukan oleh Priya pada tahun 2022 menggunakan *GPT-3* untuk menghasilkan dan memprediksi teks berbasis ujaran kebencian dan klasifikasi kalimat positif atau negatif dengan metode *zero-shot learning*, *one-shot learning*, *few-shot learning*, dan *few-shot approach* dengan *mixed category*. Hasil penelitian adalah skor akurasi sebesar 45%; 72%; 80% untuk *zero-, one-,* dan *few-shot learning* [20].

Penelitian berjudul "*Automatic Arabic Poem Generation with GPT-2*" dilakukan oleh Ghaly pada tahun 2022 yaitu menguji kelayakan model bahasa *GPT-2* untuk menghasilkan puisi berbahasa arab. Hasil penelitian ini adalah skor *BLEU-1*, *BLEU-2*, *BLEU 3*, *BLEU-4* sebesar 0.8739; 0.5369; 0.3230; 0.1871 [21].

Penelitian berjudul “*Improving Short Text Classification With Augmented Data Using GPT-3*” dilakukan oleh Salvador pada tahun 2022 yaitu menambahkan *training set* yang dihasilkan oleh *GPT-3* dengan hasil *GPT-3 Classification Endpoint* dan *GPT-3 Completion Endpoint*. Hasil penelitian ini adalah klasifikasi *endpoint* lebih konsisten akurasinya dengan nilai 0.73 setelah melalui *training set* dengan 10.000 contoh [22].

Penelitian berjudul “*An Automated Approach for the Prediction of the Severity Level of Bug Reports Using GPT-2*” dilakukan oleh Mohsin pada tahun 2022 untuk memprediksi tingkat keparahan laporan *bug* dengan menggunakan fitur *GPT-2* untuk menghindari *overfitting* dan menurunkan kompleksitas komputasi. Hasil penelitian ini adalah skor akurasi, *recall*, *precision*, *F1-score* sebesar 91%; 91%; 93%; 91% [23].

Penelitian terdahulu yang berkaitan dengan penelitian skripsi ini dapat dilihat secara keseluruhan pada Tabel 2.1.

Tabel 2.1. Penelitian Terkait

No	Judul Jurnal	Penulis & Tahun	Bidang NLP	Hasil
1	<i>Improving Language Understanding by Generative Pre-Training</i>	Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever (2018) [6].	<i>Question Answering, Text Generation</i>	Skor <i>absolute improvement</i> 8.9% pada penalaran / <i>commonsense reasoning</i> (<i>Stories Cloze Test</i>), 5.7% untuk menjawab pertanyaan (<i>RACE</i>), dan 1.5% pada keterlibatan tekstual / <i>textual entailment</i> (<i>MultiNLI</i>).
2	<i>A Text Abstraction Summary Model Based on BERT Word Embedding and Reinforcement Learning</i>	Qicai Wang, Peiyu Liu, Zhenfang Zhu (2019) [7].	<i>Content Extraction, Text Generation</i>	Skor <i>ROUGE-1</i> , <i>ROUGE-2</i> , <i>ROUGE-L</i> , dan R-AVG masing-masing meningkat 1,07%, 2,46%, 0,95%, dan 1,46%.
3	<i>Customizable text generation via conditional text generative adversarial network</i>	Jinyin Chen, Yangyang Wu, Chengyu Jia (2019) [8].	<i>Text Generation</i>	Skor akurasi 1.00, <i>recall</i> 1.00, dan <i>F1-score</i> 1.00 pada dataset <i>Yelp</i> . Pada dataset <i>film review</i> klasifikasi <i>negative review</i> mendapat akurasi 0.75, <i>recall</i> 0.71, dan <i>F1-score</i> 0.73, pada

Tabel 2.1. Penelitian Terkait (lanjutan)

No	Judul Jurnal	Penulis & Tahun	Bidang NLP	Hasil
				<i>review positive</i> akurasi 0.72, <i>recall</i> 0.76, <i>F1-score</i> 0.74. Pada dataset <i>Amazon review</i> , didapatkan akurasi 1.00, <i>recall</i> 1.00, dan <i>F1-score</i> 1.00.
4	<i>Text summarization using transfer learning, Extractive and abstractive summarization using BERT and GPT-2 on news and podcast data</i>	Victor Riisne, Adele Siitova (2019) [9].	<i>Content Extraction, Text Generation</i>	Matriks <i>ROUGE</i> untuk <i>F1-score</i> pada test <i>podcast</i> dengan nilai 16.28 pada <i>ROUGE-1</i> , 4.09 pada <i>ROUGE-2</i> , dan 12.80 pada <i>ROUGE-L</i> . Pada dataset <i>CNN / Daily Mail</i> didapatkan <i>F1-score</i> 39.38 pada <i>ROUGE-1</i> , 14.64 pada <i>ROUGE-2</i> , 35.38 pada <i>ROUGE-L</i> .
5	<i>Automatic Text Summarization of COVID-19 Medical Research Articles using BERT and GPT-2</i>	Bowen Tan, Virapat Kieuvongngam, Yiming Niu (2020) [10].	<i>Content Extraction, Text Generation</i>	skor <i>ROUGE</i> dan inspeksi visual dengan hasil skor <i>ROUGE</i> keseluruhan dari 60% kelompok abstrak lebih tinggi dari kelompok 40%.
6	<i>BERT Transformer model for Detecting Arabic GPT2 AutoGenerated Tweets</i>	Fouzi Harrag, Maria Debbah, Kareem Darwish (2020) [11].	<i>Machine Translation, Text Generation</i>	Hasil evaluasi model ini adalah skor akurasi hingga 98.7%.
7	<i>Deep Learning of a Pre-trained Language Model's Joke Classifier Using GPT-2</i>	Nur Arifin Akbar, Irma Darmayanti, Suliman Mohamed Fati, Amgad Muneer (2021) [12].	<i>Content Extraction, Classification</i>	Hasil penelitian ini adalah model <i>classifier</i> yang dibuat lebih unggul dalam hal akurasi, presisi, <i>recall</i> , dan <i>F1-score</i> dengan mencapai angka masing-masing akurasi 0,983, presisi (0,953), <i>recall</i> (0,978), dan <i>F1-score</i> (0,964).
8	<i>Medically Aware GPT-3 as a Data Generator for Medical Dialogue Summarization</i>	Bharath Chintagunta, Namit Katariya, (2021) [13].	<i>Content Extraction, Text Generation.</i>	Performa model ini dibandingkan antara model <i>PEGASUS</i> dan <i>DRSUM</i> , dengan hasil skor <i>ROUGE-L</i> adalah 62.45 dan 53.39.
9	<i>Automatic Title Generation for Text with Pre-trained Transformer Language Model</i>	Prakhar Mishra, Chaitali Diwan, Srinath Srinivasa (2021) [14].	<i>Content Extraction, Text Generation.</i>	Hasil dari penelitian ini adalah skor <i>ROUGE-1</i> , <i>ROUGE-2</i> , <i>ROUGE-L</i> adalah 0.376, 0.188, 0.336 sebagai nilai tertinggi dibandingkan model yang lainnya. Skor <i>BLEU-1</i> , <i>BLEU-2</i> , <i>BLEU-3</i> , <i>BLEU-4</i> yaitu 0.358, 0.250, 0.209, 0.124 sebagai nilai tertinggi

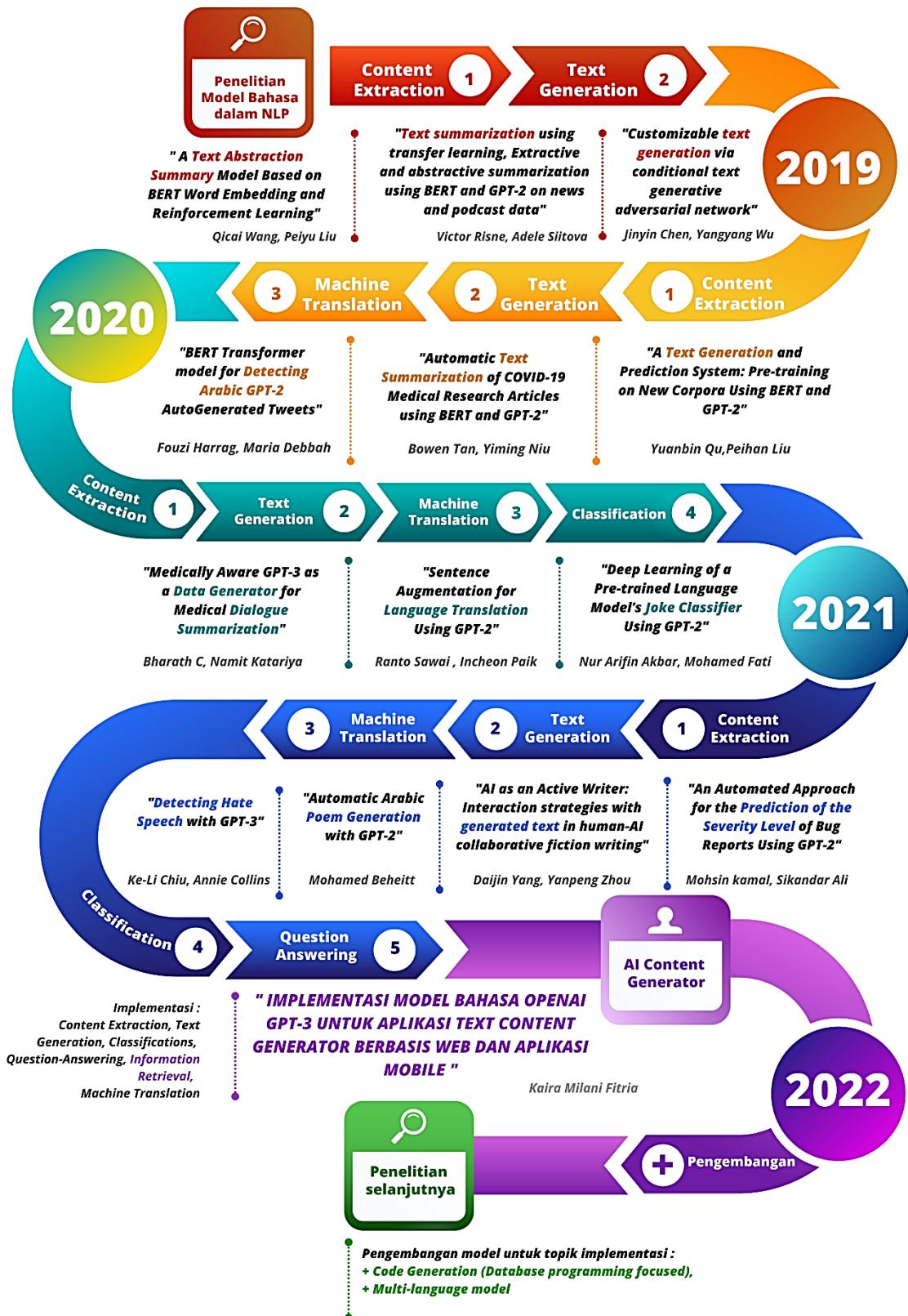
Tabel 2.1. Penelitian Terkait (lanjutan)

No	Judul Jurnal	Penulis & Tahun	Bidang NLP	Hasil
				daripada <i>PREFIX</i> , <i>TextRank</i> , <i>Bi-GRU w/ Attention</i> , dan <i>Zero-Shot GPT-2</i> .
10	<i>Prompt scoring system for dialogue summarization using GPT-3</i>	George P. Prodan, Elena Pelican (2021) [15].	<i>Content Extraction, Classification, Text Generation</i>	Pada setup GPT-3+SS rata-rata 0.3478 (<i>ROUGE-1</i>), 0.3296 (<i>ROUGE-L</i>). Pada setup <i>GPT-3+random</i> didapatkan rata-rata 0.1683 (<i>ROUGE-1</i>), 0.2391 (<i>ROUGE-L</i>).
11	<i>GPT3-to-plan: Extracting plans from text using GPT-3</i>	Alberto Olmo, Sarath Sreedharan, Subbarao Kambhampati (2021) [16].	<i>Content Extraction, Text Generation.</i>	Hasil dari penelitian ini adalah evaluasi dan metrik <i>F1-score</i> sebesar 80%. Sehingga <i>GPT-3</i> memiliki kemampuan untuk mengurutkan rencana dengan mengenali kata-kata dan menyusun kembali rencana yang sesuai.
12	<i>Abstractive Review Summarization based on Improved Attention Mechanism with Pointer Generator Network Model</i>	J.Shobana, Murali (2021) [17].	<i>Content Extraction, Text Generation.</i>	Hasil dari penelitian ini adalah skor <i>ROUGE-1</i> sebesar 41,2; <i>ROUGE-2</i> 22,4; <i>ROUGE-L</i> 37,1 pada <i>Proposed Model</i> , menjadi skor yang tertinggi dibanding metode yang lain.
13	<i>Sentence Augmentation for Language Translation Using GPT-2</i>	Ranto Sawai , Incheon Paik, Ayato Kuwana (2021) [18].	<i>Content Extraction, Text Generation.</i>	Penelitian ini menggunakan dataset <i>Tatoeba</i> , <i>WMT14</i> , dan <i>WMT18</i> . Hasil dari percobaan ini adalah skor <i>BLEU</i> untuk <i>Tatoeba En-Ja</i> yaitu 27.50; <i>WMT14 En-De</i> 24.12; dan <i>WMT18 En-Ch</i> 24.12.
14	<i>Detecting Hate Speech with GPT-3</i>	Ke-Li Chiu, Annie Collins, Rohan Alexander (2022) [19].	<i>Classification, Question Answering.</i>	Hasil <i>zero-shot learning</i> didapatkan rata-rata akurasi 56%, dengan rata-rata <i>F1-score</i> 70%. Pada <i>one-shot learning</i> rata-rata akurasi 55% dengan rata-rata <i>F1-score</i> 55%. Pada <i>few-shot learning</i> didapatkan rata-rata akurasi 67% dengan rata-rata <i>F1-score</i> 62%. Dalam <i>mixed-category few-shot setting</i> , didapatkan <i>F1-score</i> 79% dan <i>recall</i> 50%.
15	<i>Hate Speech Detection using OpenAI and GPT-3</i>	Priya, Sachin Gupta (2022) [20].	<i>Content Extraction, Classification</i>	Hasil yang didapatkan adalah pada <i>zero-shot</i> dan <i>one-shot learning</i> model mendapatkan skor akurasi antara 45% hingga 72%, dan pada <i>few-shot</i>

Tabel 2.1. Penelitian Terkait (lanjutan)

No	Judul Jurnal	Penulis & Tahun	Bidang NLP	Hasil
				<i>learning model</i> akurasinya hingga 80%.
16	<i>Automatic Arabic Poem Generation with GPT-2</i>	Mohamed El Ghaly Beheit, Moez Ben Haj Hmida (2022) [21].	<i>Machine Translation, Text Generation</i>	Hasil penelitian ini adalah skor evaluasi <i>BLEU-1</i> , <i>BLEU-2</i> , <i>BLEU 3</i> , <i>BLEU-4</i> dengan skor tertinggi daripada model bahasa lainnya, dengan skor masing-masing yaitu 0.8739, 0.5369, 0.3230, 0.1871.
17	<i>Improving Short Text Classification With Augmented Data Using GPT-3</i>	Salvador V. Balkus, Donghui Yan (2022) [22].	<i>Content Extraction, Text Generation.</i>	Hasil penelitian ini adalah klasifikasi <i>endpoint</i> lebih konsisten akurasinya dengan nilai 0.73 setelah melalui <i>training set</i> dengan 10.000 contoh.
18	<i>An Automated Approach for the Prediction of the Severity Level of Bug Reports Using GPT-2</i>	Mohsin kamal, Sikandar Ali, Anam Nasir (2022) [23].	<i>Content Extraction, Classification</i>	Skor hasil masing-masing 91,00% (<i>F1-score</i>), 93.00 (<i>precision</i>), 91.00 (<i>recall</i>), 91.41 (<i>accuracy</i>).

Dalam suatu rancangan penelitian, dapat dibuat suatu *roadmap* yang menggambarkan perkembangan penelitian dimulai dari penelitian sebelumnya dan merencanakan penelitian yang akan dilakukan serta harapan untuk pengembangan penelitian setelahnya. Penelitian dalam menyelesaikan beberapa tugas *NLP* dengan menggunakan model bahasa manapun sudah dilakukan, dimana penelitian sebelumnya berfokus pada satu tugas saja seperti *summarization*, *classification*, *machine translation*, dan yang lainnya. Sebagai pengembangan dari penelitian sebelumnya, peneliti merancang suatu aplikasi yang melakukan beberapa tugas yang sering berkaitan dengan topik *NLP*, seperti *content extracting*, *text generating*, *machine translating*, *classification*, dan juga *question answering* dapat dilihat pada *state of the art* dari penelitian yang akan dilakukan pada Gambar 2.1.



Gambar 2.1. *State of the Art / Roadmap Penelitian*

2.2 *Copywriting*

Copywriting merupakan tulisan atau naskah yang dibuat oleh seorang penulis naskah untuk kepentingan komersial. Naskah ini akan diterapkan pada berbagai media iklan. Penulis naskah disebut *copywriter*, dan kegiatan yang dilakukan oleh *copywriter* disebut sebagai *copywriting*. Naskah iklan yang dibuat disesuaikan dengan kebutuhan industri saat ini, misalnya situs *website*, brosur, *tagline*, *sales letter*, *press release*, *direct mail*, *newsletter*, dan masih banyak lagi. Berikut beberapa jenis *copywriting* :

1. *Direct Response Copywriting*, merupakan konten yang berbentuk tulisan yang sifatnya mengarahkan audiens atau pembacanya untuk mengklik suatu halaman dalam *website*.
2. *Marketing Copywriting*, adalah konten yang sifatnya informatif mengenai penjualan dan tulisannya dapat meyakinkan konsumen mengenai fungsi atau keunggulan dari produk/*brand* yang ditawarkan.
3. *Brand Copywriting*, membuat konten yang berfokus pada penyampaian deskripsi atau identitas suatu *brand*.
4. *Search Engine Optimization (SEO) Copywriting*, berfokus pada bagaimana konten dapat tampil di *search engine*, agar ketika ada calon konsumen yang mencari produk tertentu, akan muncul informasi mengenai produk yang dijual dari *keyword* yang diinputkan dalam *search engine* tersebut.

Copywriter berfokus pada konten yang bersifat informatif, edukatif, dan menghibur pembaca serta cukup familiar dengan strategi pemasaran *online* dan dapat membuat konten untuk berbagai kebutuhan seperti *video channel*, *website*, *blog*, sosial media maupun *email*. *Copywriters* dalam pandangan bisnis digital, bertugas untuk mengolah kata-kata yang bersifat meyakinkan pembaca untuk mengambil suatu tindakan, salah satunya adalah membeli produk, *download* laporan, atau berinteraksi dengan perusahaan. Beberapa kemampuan yang harusnya dimiliki oleh *copywriter* adalah dapat menentukan kata kunci yang tepat untuk memaksimalkan *views* konten dalam sistem pencarian, memproduksi konten yang menarik, menyusun konten agar mudah dibaca dan dipahami, menyusun kalimat bahasa

inggris dengan baik, mengetahui cara menyusun kalimat agar pembaca tertarik, serta mampu menghasilkan ide yang baru. Pemilihan kata serta gaya bahasa yang diterapkan dalam *copywriting* membutuhkan kreatifitas yang mampu menyampaikan esensi dari *brand* (*brand essence*) untuk menambah nilai inovasi. *Copywriting* yang menarik akan membantu meningkatkan *engagement* konsumen karena konsumen dapat mengidentifikasi produk tersebut lebih cepat. Dalam menggunakan *copywriting*, konten akan memberikan makna yang lebih mendalam pada suatu *brand*, sehingga akan terbentuk relevansi antara brand dengan audiens. Poin inilah yang akan memberikan dampak positif untuk *brand*.

Dalam hal *copywriting*, terjadi penerjemahan aktifitas *menulis* teks untuk suatu kegiatan periklanan dan pemasaran lainnya. *Copywriting* tidak menerapkan aturan yang baku yang harus diikuti, serta metode yang diterapkan sifatnya subjektif. *Copywriter* akan bertanggungjawab dalam penyusunan kata-kata sesuai konteks konten yang akan dibuat agar mengarahkan pembaca mengambil tindakan yang diharapkan sesuai tujuan pembuatan konten. Hasil kerja *copywriter* akan memberikan hasil konten yang memiliki kinerja tinggi sehingga dapat meningkatkan *engagement* dalam bisnis digital.

2.2.1 Kalimat Kreatif

Jenis konten yang difokuskan pada penelitian ini merupakan konten yang berbentuk teks. Dalam beberapa fitur yang dibuat, difokuskan untuk menghasilkan kalimat yang kreatif, dan persuasif, sehingga diharapkan teks yang dihasilkan oleh *AI* termasuk teks yang sifatnya minim plagiasi dan kalimat dengan ragam kata yang menarik. Beberapa aspek yang mengindikasikan kalimat tersebut merupakan kalimat kreatif, yaitu dengan memiliki beberapa karakteristik, diantaranya [24] :

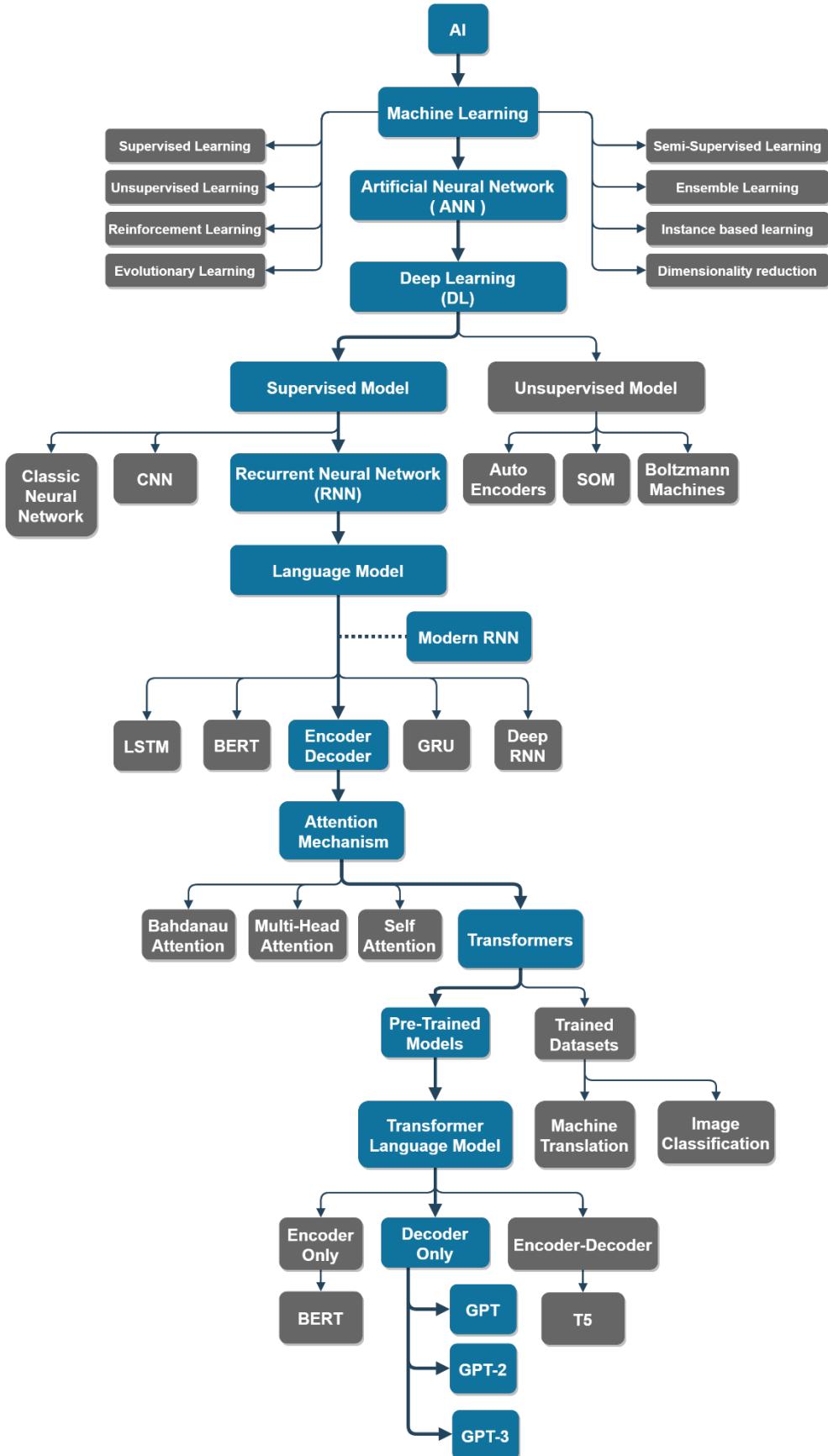
1. Kalimat tersebut bersifat menarik perhatian saat dibaca sekilas, terutama pada kalimat iklan dimana objektivitas harus difokuskan agar pembaca tertarik untuk membaca kalimat tersebut.
2. Kalimat yang mudah dipahami, yang cenderung singkat dan padat lebih baik dibanding dengan kalimat yang terlalu panjang.

3. Kalimat tersebut membangkitkan keinginan membaca, yang sifatnya mengambil perhatian pembaca untuk mengambil tindakan setelah membaca kalimat tersebut.
4. Kalimat yang memberikan kemungkinan pembaca untuk melakukan hal yang diharapkan oleh penulis.

2.3 Artificial Intelligence (AI)

Kecerdasan buatan diidentifikasi dengan mesin yang menerapkan aspek kecerdasan manusia / *human intelligence* yang mengalami perkembangan hingga saat ini. *Artificial Intelligence (AI)* termasuk salah satu subjek yang berkaitan erat antara ilmu komputer dan keteknikan. Hal ini mengindikasikan bahwa *AI* akan bekerja pada pemrosesan data melalui komputer.

Terdapat dua karakteristik yang mendominasi *AI* yaitu *self-learning* dan *connectivity*. *Self-learning* dalam *AI* mengindikasikan mesin yang secara otomatis akan meningkatkan pengetahuannya berdasarkan pengalaman yang dapat dicapai dengan melakukan pembelajaran mesin berbasis algoritma (*algorithm-based machine learning*) yang akan belajar dari *dataset* dan melakukan prediksi dengan menggunakan jaringan saraf tiruan (*artificial neural networks*) [25]. *AI* mampu untuk melakukan peningkatan kemampuan mandiri (*self-improve*). *AI* dalam konteks *connectivity*, berhubungan dengan komunikasi internet yang secara signifikan sehingga kemampuannya meningkat dibandingkan dengan mesin individu. *AI* terdiri atas seperangkat algoritma yang meniru kecerdasan manusia dengan pembelajaran mesin (*machine learning*) yang didalamnya ada teknik pembelajaran mendalam (*deep learning*). *Machine learning* mengolah data komputer dan menggunakan teknik statistik untuk melakukan pembelajaran agar kinerja sistem semakin baik dalam tugas tertentu tanpa harus melalui pemrograman secara khusus untuk tugas tersebut, sehingga menghilangkan kebutuhan akan banyaknya baris kode yang tertulis. Pengembangan topik *AI* hingga terbentuknya model bahasa *GPT-3* yang diterapkan oleh peneliti dapat dilihat pada Gambar 2.2.



Gambar 2.2 Pengembangan *AI* hingga terbentuknya *GPT-3*

2.3.1 Konsep *AI* dalam *Copywriting*

Pengaplikasian *AI* dalam konsep *copywriting* yaitu program *AI* dapat membantu *copywriter* untuk meningkatkan konten orisinil mereka atau menghasilkan (*generate*) konsep baru untuk konten. Secara sederhana, *AI* mungkin tidak akan menggantikan *copywriter*, tetapi *AI* berfungsi sebagai alat yang berguna untuk membantu pertumbuhan industri *copywriting* di masa depan.

Pemanfaatan *AI* untuk tugas *copywriting* ini dapat dilakukan dengan memanfaatkan adanya model bahasa yang sudah ada saat ini, salah satunya adalah *GPT-3*, suatu model bahasa yang dilatih untuk menghasilkan data bentuk teks. Cara kerja utama dari model bahasa *GPT-3* ini adalah model mampu memprediksi kata berikutnya dengan memahami konteks kalimat yang diberikan. Dalam penerapannya sebagai mesin penulis *copywriting*, maka model harus menerima informasi atau petunjuk mengenai topik yang ingin ditulis. Semakin banyak panduan yang diberikan pada model bahasa, maka *AI* akan semakin baik dalam memberikan ide dan *menulis* draf konten. Keuntungan penggunaan model bahasa ini selain untuk menghasilkan draf konten, juga untuk membantu *copywriter* memilih kata kunci yang tepat sehingga konten yang dihasilkan akan terstruktur dengan susunan kalimat yang bervariasi. Penggunaan aplikasi *copywriter* membantu untuk menghemat waktu dalam menentukan ide hanya dengan memberikan topik atau konteks kalimat pada model *AI* dan *AI* akan menghasilkan ide dan garis besar yang dapat dimodifikasi oleh *copywriter*.

2.4 *Machine Learning (ML)*

Machine learning memungkinkan suatu komputer untuk meniru serta menyesuaikan perilaku seperti manusia. Dengan *machine learning*, setiap interaksi atau tindakan yang dilakukan akan menjadi sesuatu yang dapat menjadi bahan pembelajaran, dan nantinya akan digunakan oleh sistem sebagai *experience* untuk waktu selanjutnya. Kategori tugas yang diluar kemampuan manusia dapat diselesaikan secara efektif dengan *machine learning*, yang menganalisis kumpulan data yang besar dan kompleks seperti *remote sensing* / penginderaan jauh, perkiraan

cuaca, *e-commerce*, pencarian *website*, dan masih banyak lagi. Dengan jumlah data yang besar tersebut, akan sangat kompleks jika manusia secara langsung melakukan prediksi data secara manual [26].

Berdasarkan pada cara kerja algoritma dan ketersediaan *output* saat *training*, paradigma pembelajaran mesin dapat diklasifikan dalam beberapa kategori, salah satu kategori yang diterapkan dalam penelitian ini yaitu *Artificial Neural Networks (ANN)*.

Artificial Neural Networks (ANN) atau Jaringan Saraf Tiruan menggunakan mekanisme berdasarkan jaringan saraf biologis. Jaringan saraf (*Neural Network*) adalah interkoneksi sel-sel neuron yang membantu impuls listrik untuk menyebar melalui otak. Sebuah neuron terdiri dari empat bagian, yaitu *dendrit* (reseptör), *soma* (pengolah sinyal listrik), *nucleus* (inti *neuron*) dan *akson* (ujung transmisi *neuron*). Dengan analogi saraf biologis, *ANN* bekerja pada tiga lapisan: lapisan *input* (*input layer*), lapisan tersembunyi (*hidden layer*), dan lapisan *output* (*output layer*). Berdasarkan tipe pembelajarannya, *ANN* diklasifikan lebih lanjut menjadi:

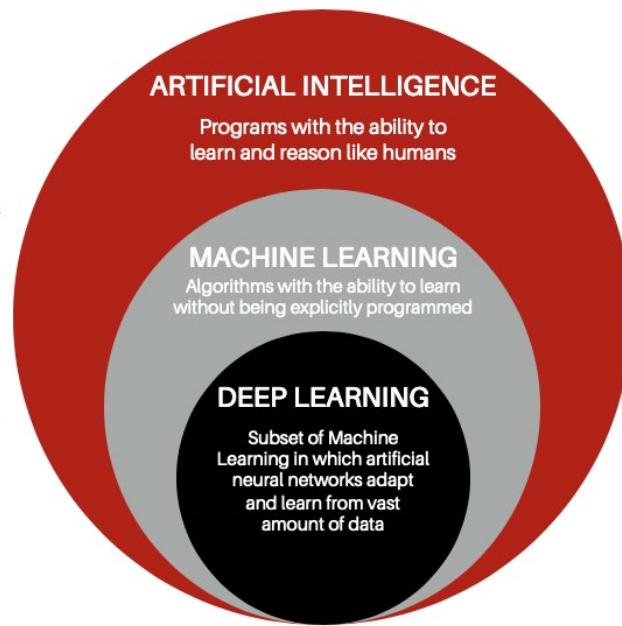
- a) *Supervised Neural Network*, dimana *input* dan *output* disajikan ke jaringan sebagai *data training*. Jaringan dilatih dengan data ini dengan menyesuaikan bobot untuk mendapatkan hasil yang akurat. Ketika sepenuhnya dilatih, akan disajikan data yang tidak terlihat untuk memprediksi *output*.
- b) *Unsupervised Neural Network*, dimana jaringan tidak dilengkapi dengan keluaran apa pun. *Unsupervised Neural Network* mencoba untuk menemukan beberapa struktur atau korelasi antara data *input* dan mengelompokkan data tersebut bersama-sama dalam sebuah kelompok atau kelas. Ketika data baru disajikan sebagai *input*, mesin akan mengidentifikasi fitur-fiturnya dan mengklasifikasikannya dalam salah satu kelompok berdasarkan kesamaan.
- c) *Reinforcement Neural Network*, dengan asumsi saat manusia berinteraksi dengan lingkungan mereka dan belajar dari kesalahan, maka *Reinforcement Neural Network* juga belajar dari keputusan masa lalunya melalui penalti untuk keputusan yang salah dan penghargaan untuk keputusan yang baik.

Bobot koneksi yang menghasilkan *output* yang benar diperkuat, sedangkan bobot koneksi yang menghasilkan respons yang salah akan dilemahkan.

Sebagai *sub-bidang* dari *AI*, *machine learning* memiliki kaitan yang erat dengan perancangan dan pengembangan suatu algoritma. *Machine learning* mencakup teknik yang digunakan untuk meningkatkan performa suatu sistem pada beberapa tugas tertentu melalui pengalaman (*training*). Fokus utama dalam *machine learning* yaitu menghasilkan *insight* yang dapat membuat keputusan atas beberapa *dataset* yang dipelajari, karena *machine learning* menggunakan *dataset* untuk menjawab pertanyaan dan membuat keputusan. Pada beberapa tugas yang kompleks, *machine learning* membutuhkan metode tambahan agar mesin yang dibuat dapat meniru kerja otak manusia, hal ini dapat disebut sebagai pembelajaran mendalam (*deep learning*).

2.5 *Deep Learning (DL)*

Deep learning dapat menyelesaikan permasalahan yang lebih kompleks daripada algoritma *machine learning*, seperti *computer vision* (pengenalan objek gambar), *speech recognition* (pengenalan suara), dan *natural language processing* (pengenalan data teks) dengan menggunakan *artificial neural network (ANN)*. *Deep learning* menggunakan *ANN* yang sifatnya fleksibel dan mampu beradaptasi secara mandiri untuk menyelesaikan permasalahan kompleks yang tidak dapat secara jelas digambarkan dengan model matematika seperti halnya pengenalan pola dan klasifikasi atau pendekatan fungsi dan kontrol [27]. *Deep learning* berfokus untuk meniru cara kerja otak manusia melalui jaringan *neuron (neural network)* dengan berbagai keragaman arsitektur. Hubungan antara *AI*, *ML*, dan juga *DL* dapat dilihat pada Gambar 2.3.



Gambar 2.3 Korelasi antara *AI*, *ML*, dan *DL* [28]

Dalam algoritma *deep learning* perlu data yang sesuai untuk mendapatkan model yang stabil dan bertahan lama. Korelasi antara data sampel, transformasi data sampel, serta hasil yang diinginkan perlu diperhatikan dalam algoritma *deep learning*. *Dataset* yang digunakan dibagi menjadi dua kelompok yaitu data pelatihan (*data training*) dan rangkaian uji (*test set*). *Training set* merupakan data yang diterapkan dalam pembangunan model, dengan memasangkan *input* dan *output* yang ditentukan, kemudian mencoba menghubungkan antar variabel dengan meminimalkan kesalahan. Setelah model diselesaikan, dilakukan rangkaian uji coba (*test set*). Karena *test set* tidak terlihat oleh model, ini digunakan untuk mendapatkan hasil akhir mengenai seberapa baik model *deep learning* sesuai dengan data, dimana ukuran ini menunjukkan seberapa baik model yang dibuat akan berperan dalam data yang sebenarnya.

Klasifikasi jenis model dalam *deep learning* terbagi menjadi 2 yaitu *supervised models* dan *unsupervised models*. Dalam *supervised models*, terdiri dari tiga jenis model yaitu *Classic Neural Networks*, *Convolutional Neural Networks (CNN)*, dan *Recurrent Neural Networks (RNN)*. Beberapa jenis algoritma yang digunakan dalam *supervised learning* pada model *deep learning* adalah :

- *Convolutional Neural Networks (CNN)*, terdiri atas beberapa *layer*, yang digunakan untuk melakukan pemrosesan data gambar dan deteksi objek.
- *Recurrent Neural Networks (RNN)*, menghasilkan suatu koneksi yang akan membentuk siklus teratur, siklus ini akan memproses *output* yang berasal dari *Long Short Term Memory (LSTM)* untuk dijadikan *input* di fase selanjutnya. RNN digunakan untuk memberikan *caption* pada gambar, analisis *time series*, pemrosesan bahasa alami (*natural language*), dan juga pengenalan tulisan tangan.
- *Long Short Term Memory (LSTM)*, menjadi salah satu jenis algortima RNN yang mempelajari ketergantungan dalam jangka panjang, dengan hasilnya yaitu model yang mampu mengingat kembali informasi dari waktu lampau.

Algoritma yang digunakan dalam pekerjaan ini termasuk dalam *supervised model*, yaitu *Recurrent Neural Networks (RNN)*.

2.6 Recurrent Neural Networks (RNN)

Recurrent neural networks (RNN) adalah model *deep learning* dengan arsitektur *Artificial Neural Networks (ANN)* untuk memproses data yang berurutan, yang dapat dianggap sebagai siklus dalam jaringan node. RNN menyimpan memori (*feedback loop*) yang memungkinkan model ini dapat mengenali pola data dengan baik untuk membuat prediksi akurat.

2.6.1 Data Teks Sekuential

Teks adalah salah satu bentuk data sekuelial yang paling umum ditemui dalam metode *deep learning*. Data teks direpresentasikan dalam bentuk urutan kata, karakter, ataupun potongan kata. Tugas ini memerlukan beberapa proses dasar yang mampu mengubah teks mentah (*raw text*) menjadi bentuk urutan (*sequential*) yang sesuai. Pemrosesan kata dimulai dengan memuat teks sebagai *string* kedalam memori, memisahkan (*split*) *string* tersebut menjadi bentuk token (kata, karakter),

membuat kamus kosakata (*dictionary*) untuk mengaitkan setiap elemen kosakata dengan indeks numerik, kemudian mengubah teks menjadi urutan indeks numerik.

Token merupakan unit teks yang atomik (sudah tidak dapat dibagi lagi). Setiap urutannya setara dengan 1 token, namun hal tersebut tergantung pilihan perancangan untuk tokenisasi kata yang dipilih. Sebagai contohnya, dalam merepresentasikan kalimat “*Baby need a new pair of shoes*” dapat dipahami kalimat tersebut terdiri dari urutan 7 kata, kumpulan semua kata terdiri dari kosa kata yang besar. Namun kalimat tersebut juga bisa direpresentasikan dalam rangkaian 30 karakter yang lebih panjang, menggunakan kosakata yang jauh lebih kecil, karena ada 256 karakter *American Standard Code for Information Interchange (ASCII)* yang berbeda.

Token yang dihasilkan masih berbentuk *string*. Sedangkan *input* token ke model harus terdiri dari *input* numerik. Sehingga dibentuklah kelas yang dapat membangun kosakata, dimana objek akan mengaitkan setiap nilai token yang berbeda dengan indeks yang unik. Pertama yang dilakukan adalah menentukan set token unik dalam *training corpus*, kemudian menetapkan indeks numerik pada masing-masing token unik. Elemen kosakata yang cukup langka atau jarang digunakan akan dihilangkan untuk alasan kenyamanan. Jika pada saat *training* dan *test* ditemukan token yang belum pernah dilihat sebelumnya, maka akan direpresentasikan sebagai token spesial "<unk>" atau nilai yang tidak diketahui (*unknown value*).

2.6.2 Model Bahasa

Setelah memetakan urutan teks menjadi bentuk token sebagai urutan dari observasi diskrit seperti kata atau karakter, diasumsikan bahwa token dalam urutan teks (T) seperti x_1, x_2, \dots, x_T . Maka dari itu tujuan dari dibentuknya model bahasa (*model language*) adalah untuk memperkirakan probabilitas penggabungan kata dari seluruh urutan kata, menjadi bentuk $P(x_1, x_2, \dots, x_T)$. Penggunaan model bahasa sangat berguna, contohnya model bahasa yang ideal akan mampu menghasilkan

teks natural dengan sendirinya, cukup dengan merancang satu token pada waktu tersebut. Jika ditinjau lebih jauh, penggunaan model bahasa mampu menghasilkan dialog yang memiliki makna, hanya dengan mengkondisikan teks pada fragmen dialog sebelumnya. Model bahasa juga bermanfaat dalam pengenalan frasa dalam ucapan yang ambigu.

Pembelajaran mengenai model bahasa juga membahas bagaimana memodelkan suatu dokumen dan urutan tokennya. Frekuensi kata diasumsikan dengan kumpulan data *training* merupakan kumpulan *dataset* dari teks dengan jumlah besar seperti data *wikipedia*, atau semua teks yang diposting di internet. Probabilitas kata dapat dihitung dari frekuensi kata relatif yang didapatkan dalam *training dataset* yang digunakan. Salah satu cara untuk mengukur kualitas model bahasa yaitu dengan mengevaluasi model dengan memeriksa seberapa mengejutkan teks yang dibuat oleh model bahasa tersebut. Model bahasa yang baik mampu memprediksi dengan token akurasi tinggi apa yang akan ditampilkan di kata selanjutnya. Jika ingin mengkompresi teks, dapat dilakukan dengan prediksi token selanjutnya dengan kumpulan token yang dimiliki saat ini. Model bahasa yang baik akan memungkinkan untuk memprediksi token berikutnya dengan akurat, sehingga akan membutuhkan lebih sedikit bit dalam mengkompresi urutan teksnya.

Model bahasa memperkirakan probabilitas gabungan kata dari urutan suatu teks. Maka dari itu, dalam perancangan suatu model bahasa dilakukan dengan adaptasi jaringan saraf (*neural networks*) dan menggunakan standar *perplexity* untuk mengevaluasi seberapa baik model bahasa dan memprediksi token berikutnya dari kumpulan token yang dalam bentuk teks. Dalam proses latihan (*training*) model bahasa, dapat dilakukan dengan mengambil sampel urutan kata *input* yang acak dengan urutan target yang diharapkan dalam *minibatch*. Data yang dimuat secara acak akan menghasilkan *minibatch* dalam setiap prosesnya.

Recurrent Neural Network merupakan suatu model jaringan saraf (*neural network*) yang menggunakan komputasi berulang untuk keadaan tersembunyi (*hidden state*). *Hidden state* dalam *RNN* dapat menangkap informasi historis dari urutan data

hingga *step* yang terjadi saat ini. Dengan begitu, jumlah parameter model *RNN* tidak akan bertambah seiring dengan pertambahan jumlah waktu. Dengan menggunakan *RNN*, dapat dibuat suatu model bahasa dengan level karakter (*character-level language models*) yang kemudian kualitas model bahasa dapat dievaluasi dengan dengan *perplexity*.

2.6.3 Implementasi *RNN*

Model bahasa tidak hanya digunakan untuk memprediksi token / kata berikutnya, namun dapat juga untuk terus memprediksi setiap token selanjutnya, dengan asumsi token yang diprediksi sebelumnya sebagai token *input* untuk token berikutnya. Contoh kasusnya adalah terkadang penulis hanya ingin membuat teks penuh dengan mempersiapkan bagian awal dokumennya saja. Hal ini akan mengembangkan fitur pelengkapan otomatis untuk suatu *search engine* dalam membantu pengguna. Mesin mencoba mencari tahu awalan teks (*prefix*) yang diinputkan oleh pengguna, dan mesin akan menghasilkan prediksi kata selanjutnya. Proses *training* model bahasa berbasis *RNN* dilakukan untuk menghasilkan teks yang mengikuti awalan (*prefix*) yang disiapkan oleh pengguna. Model bahasa *RNN* secara sederhana terdiri dari pengkodean *input*, kemudian memodelkan *RNN*, dan menghasilkan *output*. Dalam pelatihannya, model bahasa *RNN* akan melatih pada urutan teks yang diberi token pada tingkat karakternya. Pengkondisian awalan (*prefix*) oleh *user* akan membantu untuk menghasilkan kemungkinan kelanjutan kata hingga menyelesaikan teks secara keseluruhan, dimana hal ini banyak berguna pada berbagai aplikasi seperti fitur pelengkapan otomatis (*autocomplete*).

2.7 Mekanisme Atensi dan *Transformers*

Pada tahun 2014, *Google* memperkenalkan model *Sequence-to-sequence* yang menggunakan metode *Multilayered Long Short-Term Memory (LSTM)* untuk melakukan pemetaan *fixed-length input* dengan *fixed-length output* dengan kemungkinan panjang *input* dan *output* berbeda [29].

Model *seq2seq* memiliki arsitektur yang terdiri atas *encoder* dan *decoder*. *Encoder* bertugas untuk memproses urutan *input* kemudian memadatkan informasi ke bentuk vektor konteks (*sentence embedding*) yang panjang urutannya tetap. *Decoder* akan diinisialisasi dengan vektor untuk menghasilkan *output* yang diubah, yaitu dengan menggunakan bagian akhir dari *encoder* sebagai bagian awal dalam *decoder*. *Encoder* dan *decoder* dalam teknik ini merupakan model *Recurrent Neural Network (RNN)* yang menggunakan *LSTM*. Namun, model *seq2seq* tidak mampu untuk mengingat kalimat yang panjang, sehingga model ini akan melupakan bagian pertama/awal jika telah memproses keseluruhan *input* yang diberikan, untuk itu dibuatkan mekanisme atensi atau *attention mechanism* [30].

2.7.1 Mekanisme Atensi (*Attention Mechanism*)

Attention merupakan *interface* yang akan menghubungkan *encoder* dan *decoder*, dimana *attention* akan memberikan informasi penting pada *decoder* dari setiap bagian tersembunyi dalam *encoder*. *Framework* ini akan membantu suatu model agar bisa selektif menentukan titik fokus ke bagian-bagian penting dari urutan *input* yang kemudian akan mempelajari hubungan diantara *input* tersebut. Adanya mekanisme atensi akan membantu model untuk mengatasi *input* kalimat yang panjang dengan lebih efisien [31].

Mekanisme atensi (*attention mechanism*) adalah suatu algoritma yang menggunakan model *neural network* dengan fokus *local feature* untuk memperbaiki performanya pada saat pelatihan (*training*). Desain *multi-head attention* dan *self-attention* yang baru, bentuk arsitektur suatu *transformer* didasarkan pada mekanisme atensi (*attention mechanism*). Konsep dasar mekanisme atensi yaitu menambahkan *weighted access* pada setiap tahapan waktu (*timestep*) dengan tujuan memperkuat model tersebut dalam memproses data sekuel. Permasalahan dalam *neural network* yaitu model harus menghafalkan data sekuel yang cukup panjang. Analogi yang diterapkan yaitu manusia yang melakukan penerjemahan dari satu bahasa ke bahasa yang lain akan dilakukan bertahap dimulai dari anak kalimat pertama, dilanjutkan ke anak kalimat kedua,

hingga seterusnya. Analogi inilah yang mendasari solusi ketika *neural network* akan menerjemahkan suatu kalimat panjang, yaitu dengan mekanisme atensi (*attention mechanism*).

2.7.2 Self-attention

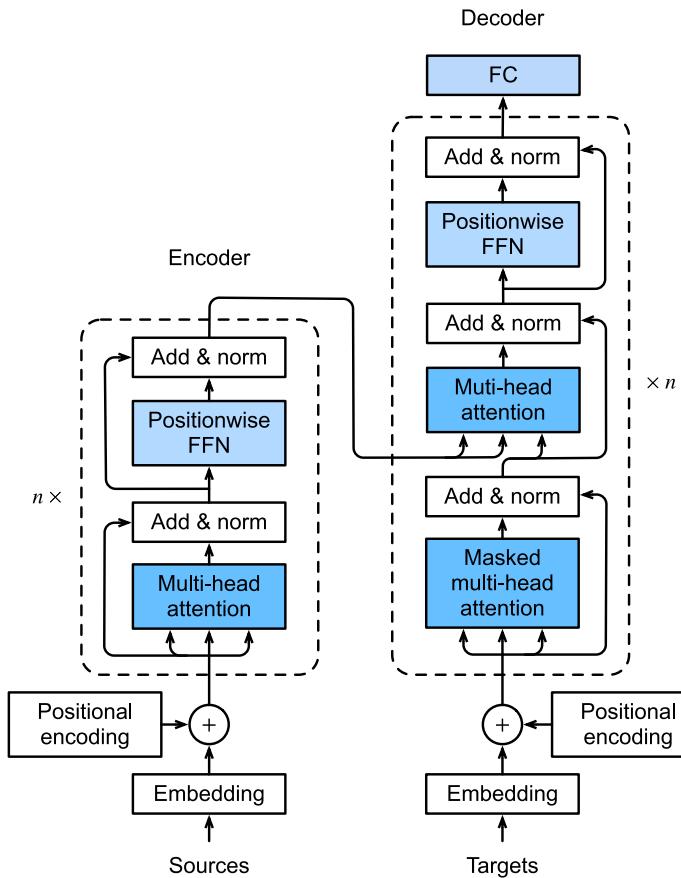
Dalam metode *deep learning*, sering digunakan *CNN* atau *RNN* untuk mengkodekan suatu urutan. Namun kini dengan adanya mekanisme atensi (*attention mechanism*), dapat dianalogikan dengan memasukkan urutan token dalam pengumpulan atensi sehingga kumpulan token yang sama akan berperan sebagai kueri (*query*), kunci (*key*), dan nilai (*value*). Setiap kueri akan memperhatikan semua pasangan *key-value* dan menghasilkan satu *output* atensi. Karena kueri, kunci, dan nilainya berasal dari tempat yang sama, akan dilakukan *self-attention* [32]. *Self-attention* membandingkan semua anggota urutan *input* satu sama lain, dan memodifikasi posisi urutan *output* agar sesuai. *Self-attention* yang termasuk dalam mekanisme atensi (*attention mechanism*) sangat bermanfaat dalam *machine reading*, *abstractive summarization*, atau *image captioning*.

2.7.3 Transformer

Revolusi penerapan *attention* dengan mengeluarkan pengulangan (*recurrence*) dan konvolusi (*convolutions*) dengan hanya mengandalkan mekanisme *self-attention*. Model *transformer* hanya didasarkan pada mekanisme atensi (*attention mechanism*) tanpa menggunakan konvolusi atau *recurrent layer*. *Transformer* mengubah satu urutan ke urutan lain hanya dengan bantuan *encoder* dan *decoder*.

Pada tingkat tinggi, *encoder transformer* adalah tumpukan beberapa lapisan identik, di mana setiap lapisan memiliki dua *sub-lapisan* (*sublayer*). *Sublayer* yang pertama adalah *multi-head self-attention pooling* dan yang kedua adalah *positionwise feed-forward network*. Secara khusus, di *encoder self-attention*, kueri, kunci, dan nilai semuanya berasal dari *output* lapisan *encoder* sebelumnya. Dalam *transformer*, untuk setiap masukan pada posisi urutan yang acak, akan membentuk sambungan kata. Penambahan dari sambungan sisa ini segera diikuti dengan normalisasi

lapisan. Sebagai hasilnya, *encoder transformer* akan mengeluarkan representasi dimensi vektor untuk setiap posisi urutan *input*. Dalam *self-attention decoder*, adanya kueri, kunci, dan nilai semuanya berasal dari *output* lapisan *dekompor* sebelumnya. Namun, setiap posisi di *dekompor* hanya diperbolehkan untuk memperhatikan semua posisi di *dekompor* hingga posisi itu. *Masked attention* akan mempertahankan properti *auto-regressive*, memastikan bahwa prediksi hanya bergantung pada token keluaran yang telah dihasilkan.



Gambar 2.4 Arsitektur *Transformer* [33]

Berdasarkan Gambar 2.4 mengenai arsitektur *Transformer*, *Transformer* memiliki komponen utama yaitu *multi-head self-attention mechanism*. *Transformer* merepresentasikan *decoder* dari *input* sebagai pasangan *key-value*. Jika dalam konteks *natural machine translation (NMT)*, kedua aspek ini termasuk dalam *encoder hidden-states*. Untuk representasi *decoder*, *output* yang dihasilkan sebelumnya akan dikompresi ke bentuk *query* kemudian *output* bentuk teksnya dihasilkan dari pemetaan *query*, *key*, dan *values* [34].

2.7.4 Pre-Training Skala Besar dengan Transformer

Model umum yang lebih baik dan general serta lebih kompeten dapat melakukan banyak tugas, baik dengan atau tanpa adaptasi sehingga jenis model *pre-training* ini perlu menggunakan ukuran data yang besar, hal ini semakin umum diperlukan untuk saat ini. Dengan digunakannya jenis data besar untuk *pre-training*, arsitektur *transformer* akan memiliki performa yang lebih baik dengan meningkatnya ukuran model dan komputasi *training*-nya, hal ini menunjukkan perilaku skalabilitas yang unggul. Kinerja model bahasa (*language model*) berbasis *transformer* memiliki kaitan erat dengan parameter model, *training token*, dan komputasi *training* [35].

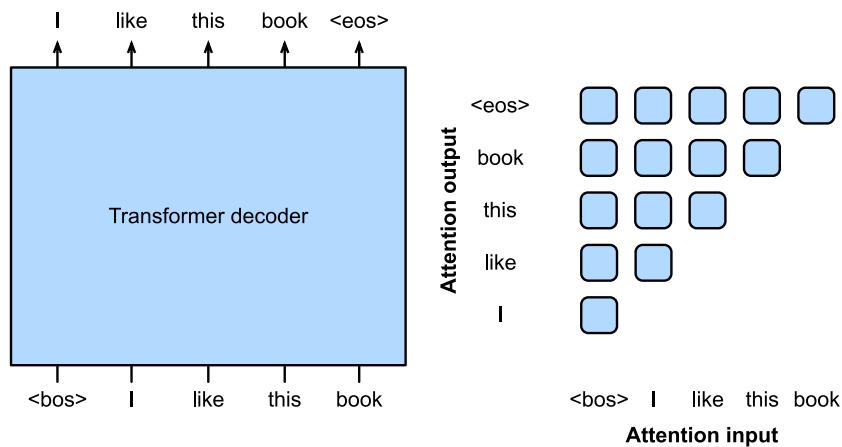
Sebelum mencapai keberhasilan dan kompilasi model, *pre-training transformer* telah dilatih untuk data *multi-modal*, secara ekstensif dilatih dengan banyak jenis teks. Arsitektur *transformer* terdiri atas *encoder* untuk mewakili urutan *input* dan *decoder* untuk menghasilkan urutan target. *Transformer* dapat digunakan dalam tiga mode berbeda yaitu *encoder-only*, *encoder-decoder*, dan *decoder-only*.

Mode *encoder-only* dapat dikategorikan ketika hanya bagian *encoder* dari *transformer* yang digunakan, urutan token *input* diubah menjadi jumlah representasi yang sama untuk diproyeksikan menjadi *output*. *Encoder* dalam *transformer* terdiri dari *self-attention layer*, dimana semua token *input* berhubungan satu sama lain. Karena representasi ini bergantung pada semua token *input*, maka representasi ini akan diproyeksikan dalam klasifikasi. Aplikasi dari *transformer* khusus *encoder* ini dilakukan pada *BERT (Bidirectional Encoder Representations from Transformers)* [36].

Karena *encoder transformer* mengubah urutan token *input* menjadi jumlah representasi *output* yang sama, maka mode *encoder-only* tidak dapat menghasilkan urutan yang panjangnya variatif dalam tejemahan mesin. Walaupun awalnya *transformer* bertujuan untuk menyelesaikan terjemahan mesin, namun arsitektur *transformer* juga memiliki *decoder* yang dapat memprediksi ukuran target dengan

panjang variatif, dari token ke token, dan tergantung pada *output encoder* dan *output decoder*. Mode *encoder-decoder* yaitu pengkondisian pada *output decoder*, akan memungkinkan token target untuk memperhatikan keseluruhan *input* token. Pengkondisian pada *output decoder* dimana setiap token target hanya akan memperhatikan token sebelumnya dan saat ini dalam target *sequence*. Pelatihan *transformer* dengan mode *encoder-decoder* dilakukan pada *BART (Bidirectional Auto-Regressive Transformers)* dan *T5 (Text-to-Text Transfer Transformer)* yang merupakan dua *Transformer encoder-decoder* yang dilatih dengan teks skala besar.

Mode *decoder-only* menghapus keseluruhan *encoder* dan *sub-layer decoder*. Saat ini, *transformer* khusus *decoder* telah menjadi arsitektur yang memanfaatkan data teks tak berlabel yang berlimpah jumlahnya melalui *self-supervised learning*. Contoh pengaplikasian *mode decoder only* yaitu pada *GPT*, *GPT-2* dan juga *GPT-3*. Menggunakan pemodelan bahasa sebagai objek *training*-nya, model *GPT* memilih untuk menggunakan *decoder* sebagai *backbone* [6].

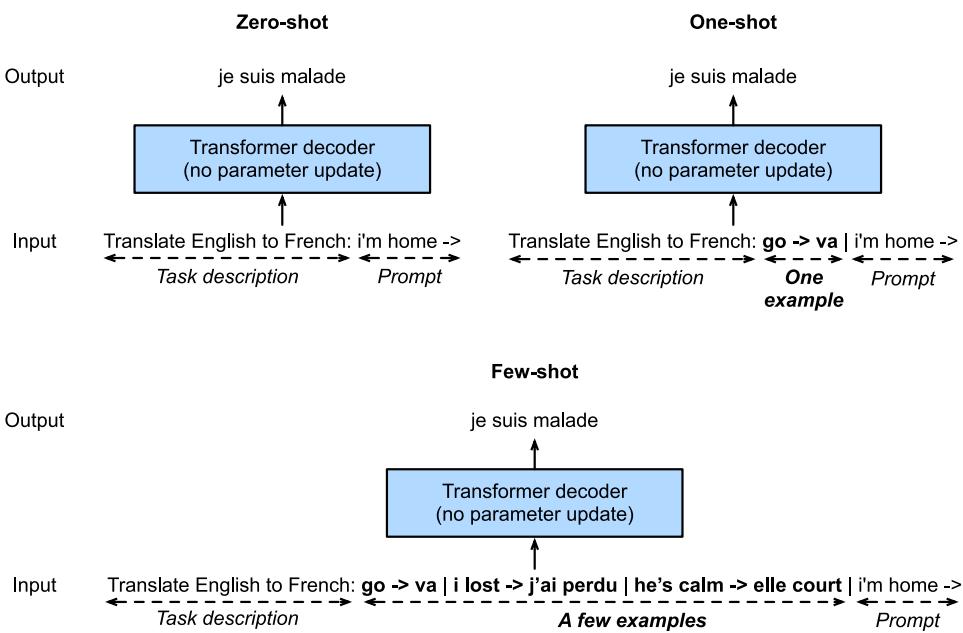


Gambar 2.5 *Pre-training GPT* dengan pemodelan bahasa [37].

Gambar 2.5 bagian kiri menjelaskan *pre-processing GPT* dengan pemodelan bahasa. Urutan target adalah urutan *input* yang digeser oleh satu token. Baik “<bos>” dan “<eos>” masing-masing adalah token khusus yang menandai awal dan akhir urutan. Gambar kanan menjelaskan pola atensi pada mode *transformer decoder*. Setiap tanda di sepanjang sumbu vertikal hanya memperhatikan tanda sebelumnya di sepanjang sumbu horizontal. Mengikuti pelatihan model bahasa

autoregresif, Gambar 2.5 mengilustrasikan pra-pelatihan *GPT* dengan *encoder transformer*, di mana urutan target adalah urutan *input* yang digeser oleh satu token. Perhatikan bahwa pola atensi dalam *decoder transformer* akan mengatur bahwa setiap token hanya dapat memperhatikan token yang sebelumnya (prediksi token per token tidak dapat menangani token yang selanjutnya).

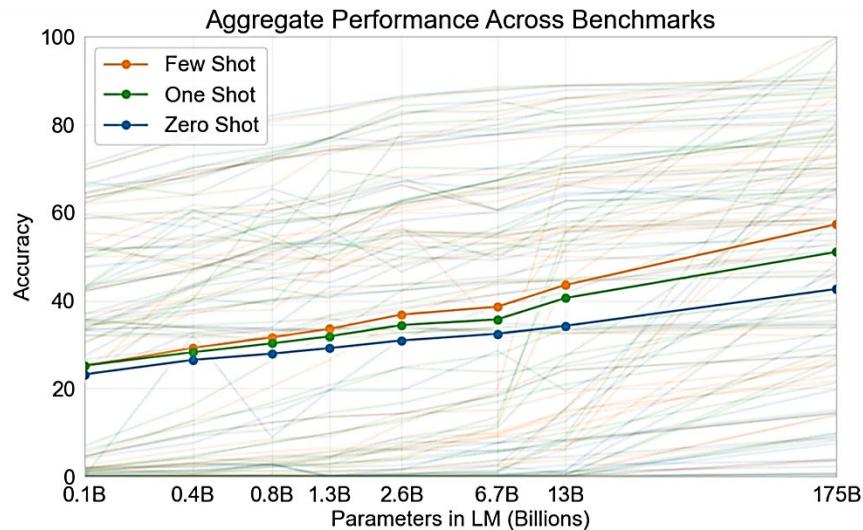
GPT memiliki 100 juta parameter dan perlu disesuaikan untuk tugas individu. Model bahasa *transformer-decoder GPT-2*, diperkenalkan satu tahun kemudian. Dibandingkan dengan *decoder transformer* asli di *GPT*, pra-normalisasi dan peningkatan inisialisasi dan penskalaan bobot diadopsi di *GPT-2*. Dilatih pada total teks berukuran 40 GB, *GPT-2* memiliki 1,5 miliar parameter sebagai hasil mutakhir pada tolok ukur pemodelan bahasa dan hasil yang menjanjikan pada beberapa tugas lain tanpa memperbarui parameter atau arsitektur. *GPT-2* menunjukkan potensi penggunaan model bahasa yang sama untuk banyak tugas tanpa memperbarui model. Model ini lebih efisien secara komputasi daripada *fine-tuning*, yang membutuhkan pembaruan model melalui komputasi gradien.



Gambar 2.6 *Zero-shot, one-shot, few-shot learning* dengan model bahasa [37].

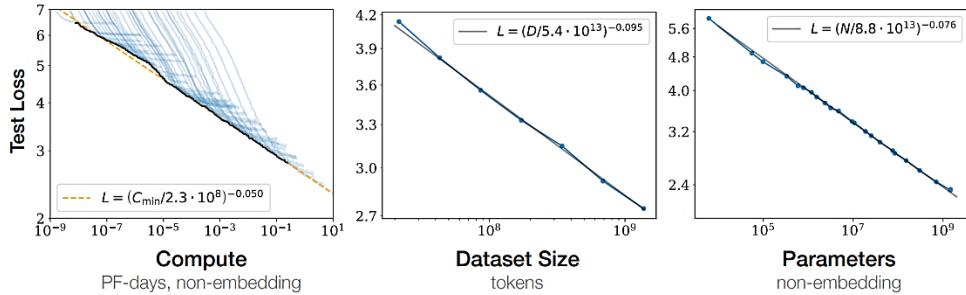
Gambar 2.6 menjelaskan bahwa model bahasa pra-latihan (*pre-trained*) dapat menghasilkan *output* tugas sebagai urutan tanpa pembaruan parameter, tergantung pada urutan *input* dengan deskripsi tugas, contoh *input-output* khusus tugas, dan *prompt* (*input* tugas). Paradigma pembelajaran ini selanjutnya dapat dikategorikan menjadi *zero-shot*, *one-shot*, dan *few-shot*.

Ketiga metode ini diuji pada *GPT-3* dengan versi terbesarnya menggunakan data dan model yang dua kali lebih besar daripada *GPT-2*. *GPT-3* menggunakan arsitektur *decoder transformer* yang sama seperti *GPT-2* kecuali pada pola atensi yang lebih ada jarak antar *layer* berseling-seling. *GPT-3* dilatih dengan 300 miliar token sehingga memiliki kinerja yang lebih baik dan ukuran modelnya lebih besar, terutama pada metode *few-shot* akan bekerja dengan cepat (Gambar 2.7). *GPT-3* telah mendukung berbagai aplikasi di seluruh *web* dengan menghasilkan 4,5 miliar kata setiap harinya setelah sembilan bulan sejak *API* dari *GPT-3* dirilis.



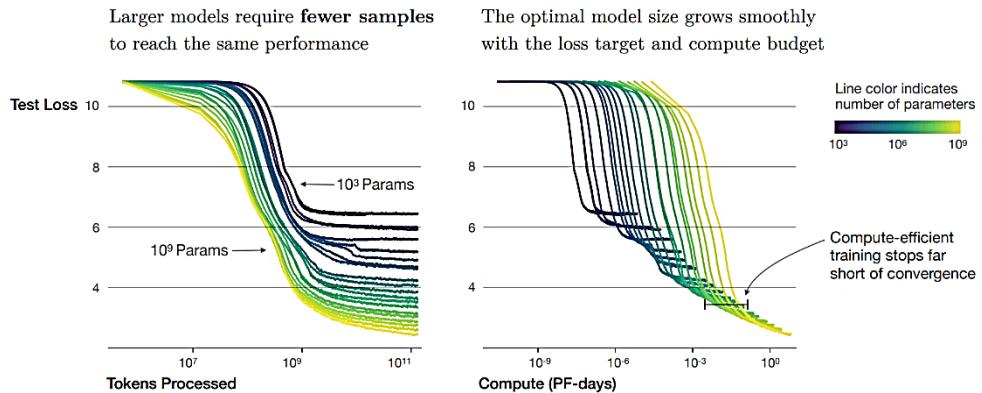
Gambar 2.7 Performa keseluruhan *GPT-3* untuk benchmark akurasi [38].

Gambar 2.7 secara empiris menunjukkan skalabilitas *transformer* dalam model bahasa *GPT-3*. Untuk meningkatkan skalabilitas *transformer* disarankan untuk melakukan *training transformer* yang lebih besar dengan lebih banyaknya data dan komputasi yang digunakan.



Gambar 2.8 Performa model bahasa *transformer* meningkat dengan signifikan seiring peningkatan ukuran model ukuran *set data*, dan jumlah komputasi yang digunakan untuk pelatihan [35].

Pada Gambar 2.8, kinerja suatu model bahasa berhubungan dengan ukuran model (jumlah parameter), ukuran *set data* (jumlah token pelatihan), dan jumlah komputasi pelatihan. Secara umum, dengan meningkatkan ketiga faktor ini akan menghasilkan kinerja yang lebih baik.



Gambar 2.9 Pelatihan *transformer* mulai 10^3 hingga 10^9 parameter [35].

Gambar 2.9 menunjukkan bahwa model yang besar akan membutuhkan lebih sedikit sampel pelatihan (token yang diproses) untuk bekerja pada tingkat yang sama dengan pencapaian model ukuran kecil, serta kinerja model bahasa ini diskalakan dengan lancar dengan komputasi.

Jenis *transformer pre-trained* yang *encoder-only* yaitu *BERT*, *encoder-decoder* yaitu *T5*, dan *decoder-only* yaitu seri *GPT*. Model *pre-trained* dapat disesuaikan untuk melakukan tugas yang berbeda dengan pembaruan model (misalnya *fine-*

tuning) atau tanpa pembaruan model (misalnya *few-shot*). Skalabilitas *transformer* menunjukkan bahwa kinerja yang lebih baik didapatkan dari model yang lebih besar, lebih banyak data pelatihan, dan lebih banyak komputasi pelatihan. Karena *transformer* pertama kali dirancang dan dilatih untuk data teks, maka akan condong ke pemrosesan bahasa alami. Meskipun demikian, model-model yang dikembangkan saat ini dapat sering ditemukan dalam model yang lebih baru di berbagai aspek.

2.8 Natural Language Processing (NLP)

Pemrosesan bahasa alami (*Natural Language Processing*) mempelajari interaksi antara komputer dan manusia menggunakan bahasa alami. Dalam praktiknya, sangat umum menggunakan teknik pemrosesan bahasa alami untuk memproses dan menganalisis data teks (bahasa alami manusia). Untuk memahami teks, kita bisa mulai dengan mempelajari representasinya. Memanfaatkan urutan teks yang ada dari kumpulan data, *self-supervised learning* yang banyak digunakan untuk melatih representasi teks, seperti dengan memprediksi beberapa bagian teks yang tersembunyi menggunakan beberapa bagian lain dari teks di sekitarnya. Dengan cara ini, model belajar melalui pengawasan (*supervised learning*) dari data teks besar tanpa upaya pelabelan dapat dimanfaatkan. Ketika memperlakukan setiap kata atau subkata sebagai token individual, representasi setiap token dapat dilatih sebelumnya menggunakan model penyisipan *word2vec*, *GloVe*, atau yang lainnya. Setelah *pre-training*, representasi setiap token dapat menjadi vektor, namun tetap sama terlepas dari konteksnya. Misalnya, representasi vektor "bank" sama dalam "pergi ke bank untuk menyimpan sejumlah uang" dan "pergi ke bank untuk duduk". Dengan demikian, banyak model pra-pelatihan yang lebih baru mengadaptasi representasi dari token yang sama ke konteks yang berbeda.

2.9 Pemodelan Bahasa *Transformer*

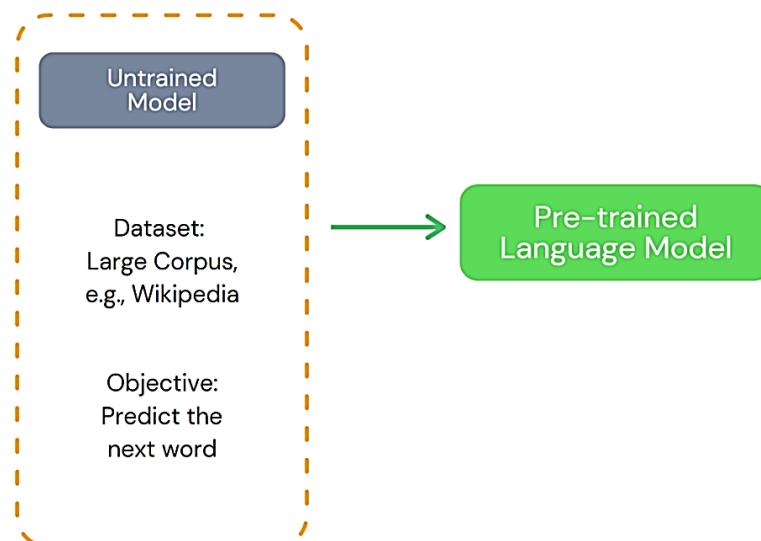
Model bahasa atau *Language Model (LM)* merupakan salah satu model *Machine Learning (ML)* yang memprediksi kata selanjutnya berdasarkan kata-kata yang telah dipelajari di teks sebelumnya. Model bahasa ini juga disebut *Causal Language*

Model (CLM). CLM memiliki cara kerja memprediksi probabilitas kata tertentu dalam suatu urutan kata. Analogi prediksi suatu kata dapat dilihat pada Gambar 2.10.



Gambar 2.10 Prediksi bahasa [39].

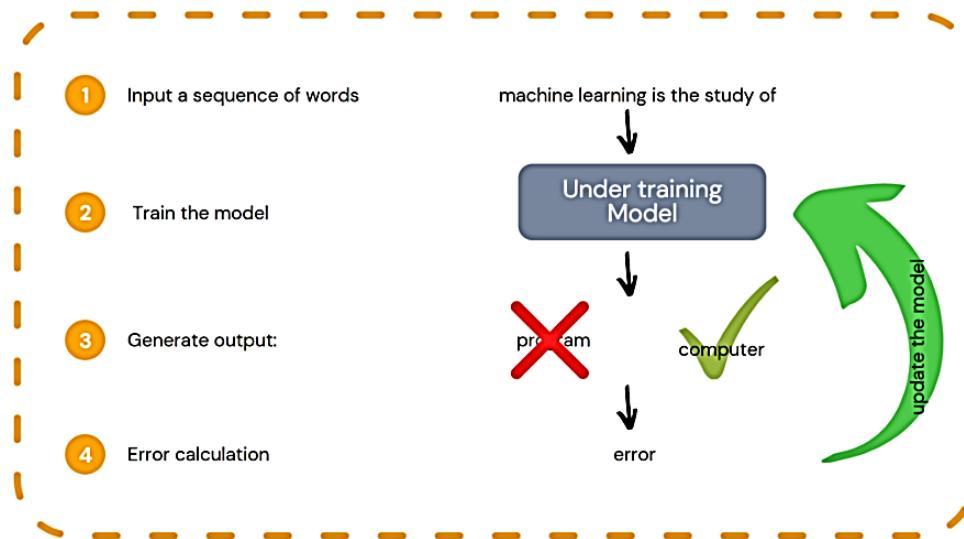
Model bahasa diharapkan dapat memprediksi kata dengan benar dan berguna pada beberapa tugas lain, sehingga model ini perlu dilatih berulang kali dengan data ukuran besar. Agar dapat melakukan *training* pada model bahasa, maka diperlukan suatu tugas tertentu. Pada model bahasa, *task* yang dilakukan adalah *task* untuk memprediksi kata selanjutnya. Model yang belum melalui *training* akan dikembangkan menjadi model bahasa *pre-trained* seperti pada Gambar 2.11.



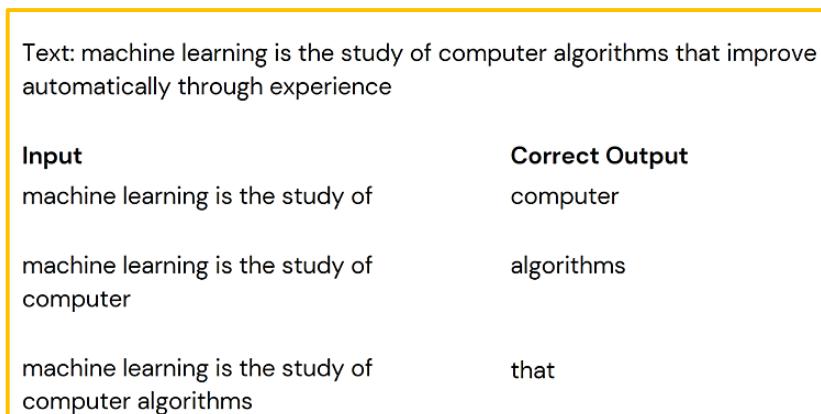
Gambar 2.11 Model bahasa *pre-trained* [40].

Training suatu model bahasa dimulai dengan memberikan *input* suatu kalimat, kemudian model akan dibangun dari *input* kalimat yang diberikan. Selanjutnya model akan menghasilkan *output* tertentu, dimana model akan terus diperbarui

berdasarkan *output* yang telah dihasilkan sebelumnya. Gambar 2.12 menjelaskan bagaimana konsep *training* dalam suatu model bahasa. Begitupun dengan Gambar 2.13 menunjukkan contoh pengolahan kata dalam proses *training*.



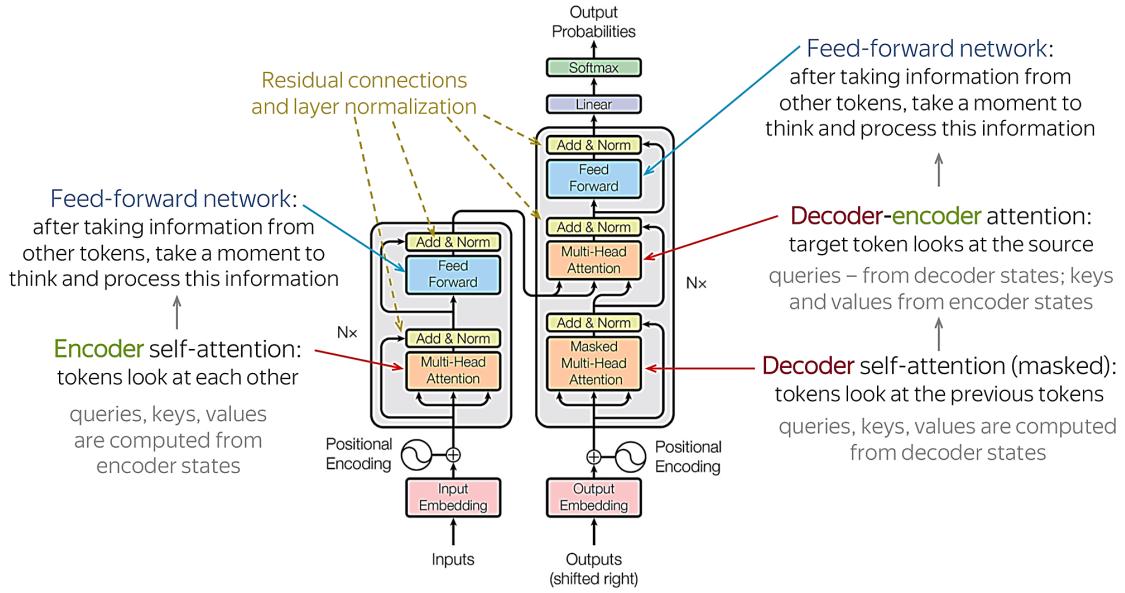
Gambar 2.12 Proses kerja model bahasa *pre-trained* [40].



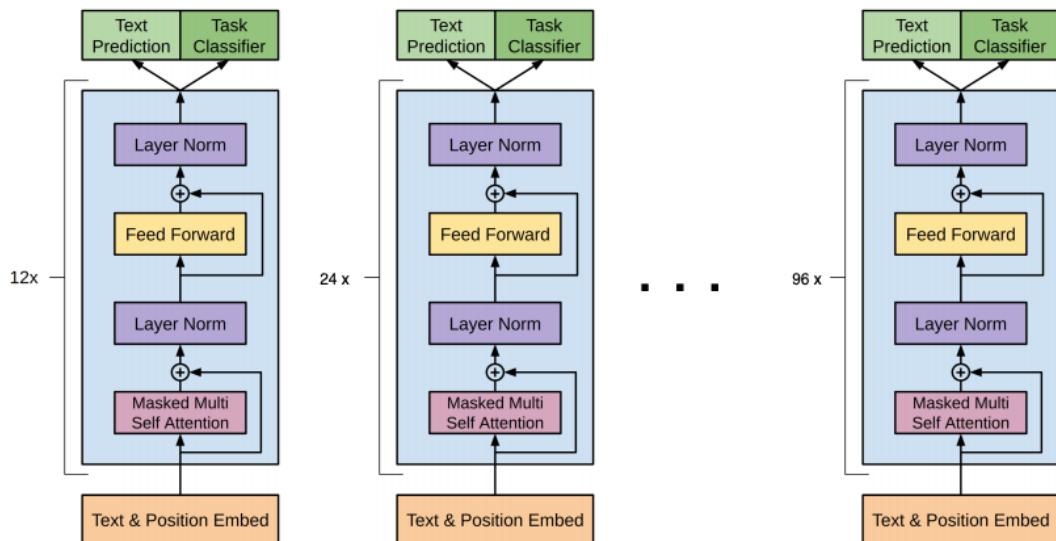
Gambar 2.13 Pembelajaran dalam model bahasa *pre-trained* [40].

2.10 Generative Pre-training Transformer (GPT)

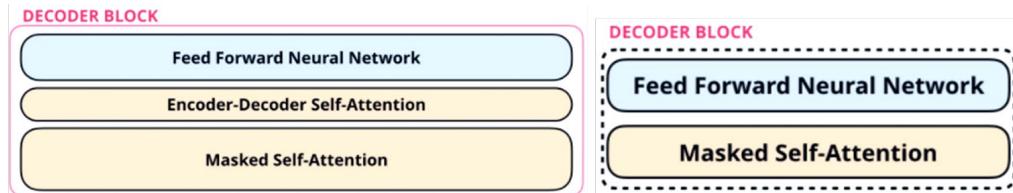
GPT merupakan singkatan dari *Generative Pre-training Transformer*. Model *GPT* menggunakan mode *encoder-only transformer*, yaitu hanya menggunakan bagian *block decoder* saja dari *Transformer*.

Gambar 2.14 Arsitektur *Transformer* [33].

Bila dibandingkan dengan arsitektur *transformer*, maka arsitektur milik *GPT-3* lebih sederhana karena hanya menggunakan bagian *encoder*-nya saja. Gambar 2.14 menjelaskan tentang arsitektur *transformer* dan bagaimana proses kerjanya, sedangkan Gambar 2.15 menjelaskan tentang arsitektur *GPT-3* yang hanya menggunakan sisi *encoder* saja, sehingga arsitekturnya lebih sederhana dibanding arsitektur *transformer* sebelumnya.

Gambar 2.15 Arsitektur *GPT-3* [6].

Baik *GPT-2* maupun *GPT-3* memiliki arsitektur yang sama, dan perbedaan utama antara kedua model adalah jumlah *layer*-nya, pada pengembangan *GPT* digunakan berbagai ukuran model antara 125 juta parameter hingga 175 miliar parameter (*GPT-3* asli). Model yang terkecil (125 juta parameter) memiliki 12 lapisan atensi, dengan masing-masing memiliki 12 *attention head*, dan masing-masing memiliki 64 dimensi. Model yang terbesar memiliki 96 lapisan atensi, dengan 96 *attention head*, dan 128 dimensi. Pengembangan yang dilakukan adalah *training dataset* yang lebih besar dengan parameter pelatihan yang lebih banyak. Karena *GPT* tidak memiliki *encoder*, maka *GPT* hanya menggunakan *decoder* yang ditumpuk lebih banyak dengan berbagai ukuran mulai dari 12-48 blok *decoder*.



Gambar 2.16 Blok *decoder transformer* (kiri), blok *decoder GPT* (kanan) [41].

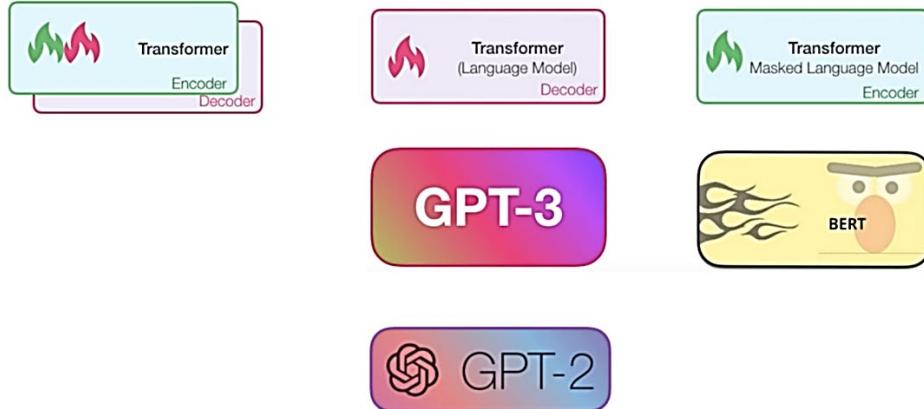
Blok *decoder* dalam *GPT* hampir sama dengan *Transformer*. Seperti pada Gambar 2.16 dimana blok *decoder Transformer* menggunakan *feed forward neural network*, *encoder-decoder attention*, dan *masked self-attention*. Namun *GPT* hanya menggunakan *masked self-attention* dan *feed forward neural network*. Seri *GPT-2* memiliki 1.5 miliar parameter yang dilatih dengan menggunakan 40 GB *file* teks dari internet (yang setara dengan 10 miliar token, dimana satu token terdiri dari 4 karakter), seri selanjutnya yaitu *GPT-3* memiliki jumlah parameter yang lebih besar yaitu 175 miliar yang dilatih dengan menggunakan 499 miliar token, dimana 175 miliar parameter ini membutuhkan memori sebesar 700 GB. Sebagai model bahasa terbesar saat ini, *GPT-3* dengan jumlah parameter 175 miliar dilatih pada *dataset* teks sebesar 45 TB.

Model bahasa *GPT-3* memiliki jumlah jumlah parameter sebanyak 175 miliar yang mampu menghasilkan teks seperti manusia dan telah di-*training* pada kumpulan data teks berjumlah besar yang memiliki ratusan miliar kata. Pemanfaatan *GPT-3*

untuk menghasilkan teks ini dapat dilakukan dengan menggunakan *API*, developer dapat memanfaatkan layanan *AI GPT-3* melalui pemanggilan *API* dalam programnya. Model *GPT-3* yang digunakan yaitu yang berjenis “*text-davinci-002*”, karena model tersebut merupakan model yang terkuat pada *GPT-3* untuk saat ini. Jumlah kata atau token yang dihasilkan oleh *GPT-3* dapat diatur pada opsi “*max token*” untuk mengatur berapa fragmen kata yang diperlukan. Mengenai jumlah *input* dan *output*-nya, *GPT-3* memiliki ukuran yang terbatas yaitu 2048 token yang diterima dan yang dihasilkan sekitar 1500 kata.

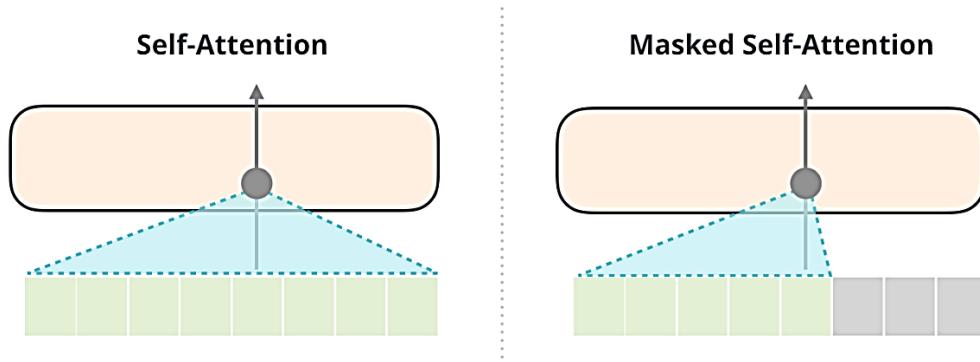
2.10.1 Perbedaan *GPT* dan *BERT*

Model *transformer* memiliki beberapa jenis lagi selain *GPT*, yaitu *Bidirectional Encoder Representations from Transformers (BERT)*. Perbedaan yang mendasari *GPT* dan *BERT*, dimana *BERT* hanya menggunakan salah satu bagian *transformer* yaitu *encoder*. Berdasarkan istilah *bidirectional* yang ada dalam *BERT*, mengarah pada model *BERT* dilatih dari dua arah (kiri-kanan dan kanan-kiri). Perbedaan yang mendasari *BERT* dan *GPT* adalah penggunaan *transformer* untuk pemodelan bahasa, dimana *OpenAI GPT* menggunakan *decoder*, dan *BERT* menggunakan *encoder*. Alasan yang mendasari model bahasa *Transformer* menggunakan salah satu bagian dari *Transformer* saja dapat dilihat dari penelitian yang dilakukan oleh Peter J. Liu mengenai pengajuan model *Transformer* dengan mode *decoder-only*, dimana penelitian ini menyatakan bahwa dugaan untuk tugas *text-to-text monolingual*, informasi yang berlebihan akan dipelajari kembali di bagian *encoder* dan *decoder*. Dimana hal ini memungkinkan adanya pengoptimalan yang lebih mudah dan pengamatan empiris dengan urutan kata yang lebih panjang [42]. Alasan lainnya bisa terjadi pengulangan (*redundant*) karena *encoder* dan *decoder* memiliki *input* yang sama. Hal ini menjadikan kedua komponen *transformer* tersebut akan memperlajari hal yang sama. Maka dari itu, jika hanya menggunakan bagian *encoder* atau *decoder*-nya saja, akan menjadikan model dapat lebih fokus dan bisa dibuat lebih kompleks melalui penambahan jumlah blok *encoder* atau *decoder*.



Gambar 2.17 Perbedaan *GPT* dan *BERT* [41].

Aplikasi penggunaan *decoder-only* atau *encoder-only* disesuaikan pada saat *pre-training*. Model bahasa *GPT* menggunakan model bahasa standar (*Causal Language Model*) yang memungkinkan *decoder* sebagai pilihan terbaik untuk *pre-training* model *CLM*. *Decoder* akan memproses token satu persatu dari kiri ke kanan hingga menutupi token yang berada di sebelah kanan, metode ini disebut sebagai *masked self-attention*. Sedangkan *BERT* lebih tepat jika menggunakan *encoder* karena mekanisme *bidirectional* yang melatih model bahasa dari dua arah (token sebelum dan selanjutnya), dimana proses ini tidak dapat dilakukan di *decoder* karena ada *masked self-attention*.



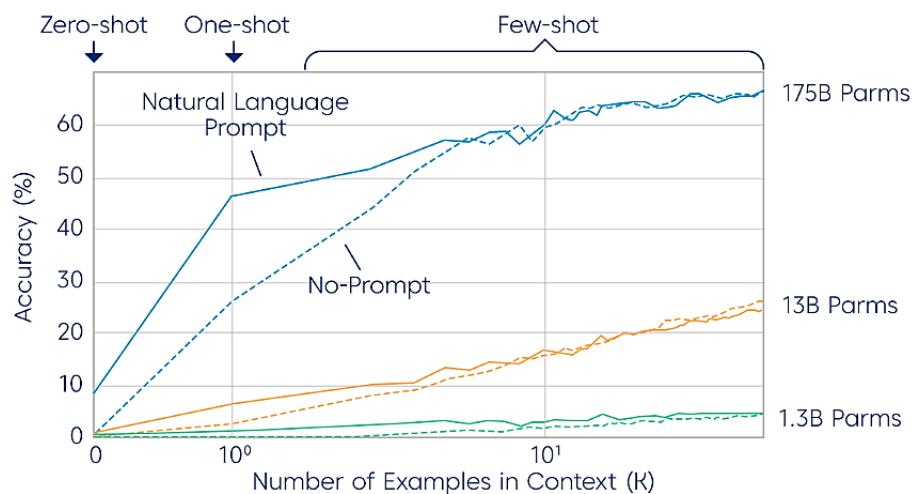
Gambar 2.18 Perbandingan mekanisme *self-attention* pada *BERT* dan *masked self-attention* pada *GPT* [41].

Auto-regressive memiliki arti bahwa setiap token yang di-*training* mendapatkan konteks melalui kata-kata yang sebelumnya. *GPT* sebagai model bahasa yang *auto-regressive*, karena token yang dilatih oleh *GPT* akan dilatih satu persatu. Sedangkan

BERT tidak termasuk *auto-regressive* karena konteks kata berasal dari sebelum dan sesudah kata yang ditutupi (*masked*). Hal yang cukup membedakan antara *GPT* dan *BERT* juga ada pada aspek ukuran *vocabulary*. *Vocabulary size* milik *GPT* sebesar 50.257 [43]. Sedangkan untuk *BERT* sebesar 30.522 [44]. Sehingga dari perbedaan tersebut, penulis memilih *GPT* sebagai model bahasa yang akan digunakan.

2.10.2 Training *GPT-3*

Suatu model bahasa, dalam hal ini model bahasa *GPT-3* jika telah melalui proses *training* maka akan menghasilkan suatu teks. Secara opsional, proses *training* dapat dilakukan dengan memberikan beberapa teks sebagai *input* yang nantinya akan mempengaruhi hasil *output*. *Output* yang dihasilkan didapatkan dari apa yang model bahasa tersebut pelajari selama periode *training*, dimana model bahasa akan memindai sejumlah besar teks. *GPT-3* menggunakan beberapa bentuk kompresi data sembari melakukan konsumsi jutaan teks sampel untuk mengubah kata-kata tersebut menjadi vektor (representasi numerik). Kemudian model bahasa akan membongkar teks yang terkompresi tersebut dalam kalimat yang dimengerti manusia. Sehingga, proses kompresi dan dekompresi teks ini akan mengembangkan akurasi model bahasa saat memprediksi probabilitas kata.

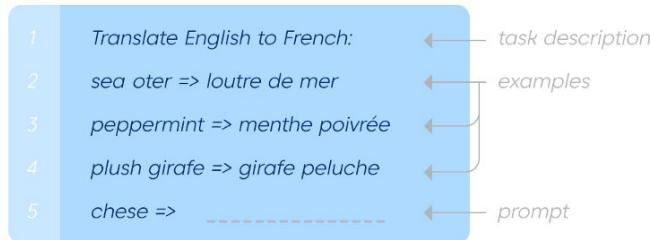


Gambar 2.19 Perbandingan akurasi proses *training* model *GPT-3* [38].

Model bahasa *GPT-3* secara efisien akan menjadi model bahasa yang lebih besar melalui informasi yang sesuai dengan konteks, sehingga meningkatkan akurasi model bahasa *GPT*.

Few-shot

In addition to the task description, the model sees a few examples of the tasks. No gradient updates are performed.

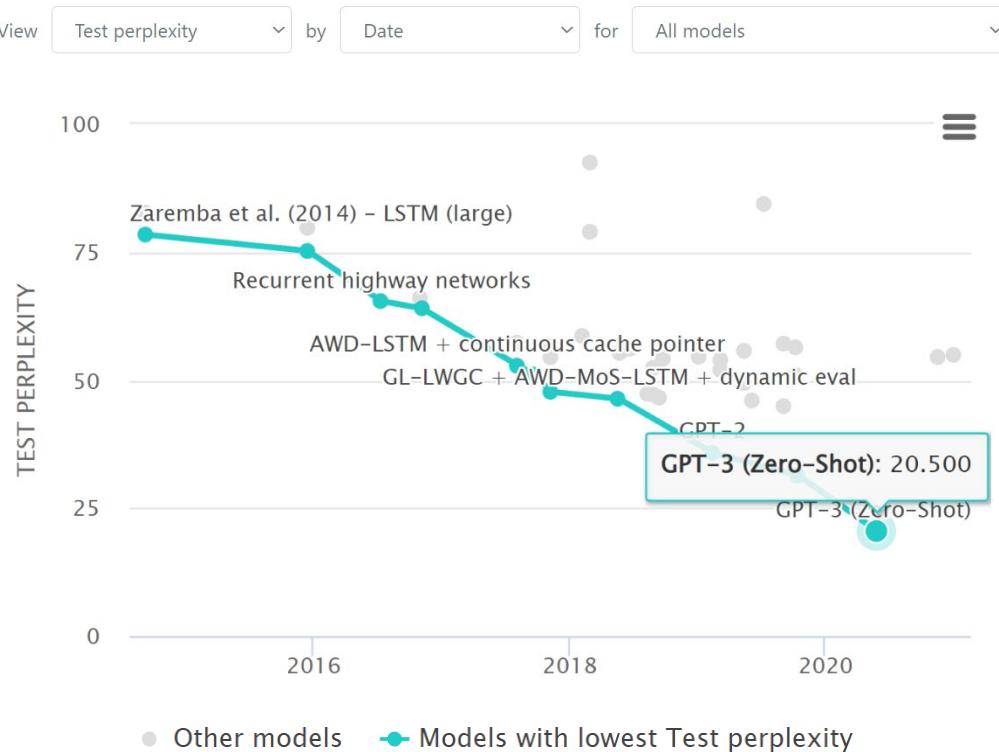


Gambar 2.20 Contoh *few-shot learning* dalam penerjemahan bahasa *GPT-3* [38].

Dikarenakan *GPT-3* memiliki kinerja tinggi dalam *few-shot learning*, maka *GPT-3* dapat merespons dengan cara yang konsisten dengan contoh teks yang diberikan yang belum pernah diekspos sebelumnya. Jadi, hanya perlu beberapa contoh untuk menghasilkan respon yang relevan, karena telah dilatih pada banyak contoh teks. Model *few-shot* hanya membutuhkan beberapa contoh untuk menghasilkan respons yang relevan, karena model ini telah dilatih pada banyak sampel teks. Setelah *training*, ketika probabilitas model bahasa sudah akurat, maka model bahasa dapat memprediksi kata berikutnya saat diberi masukan kata, kalimat, atau fragmen kalimat yang semua itu disebut *prompt*.

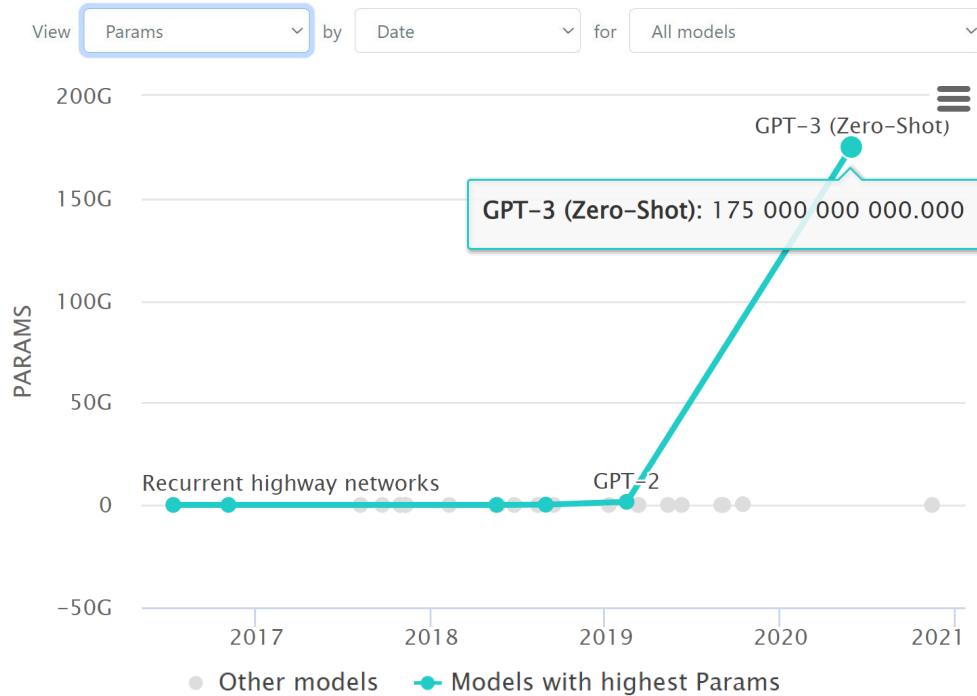
2.10.3 Perplexity GPT-3

Uji *perplexity* merupakan pengujian yang dilakukan pada model bahasa untuk melihat tingkat kebingungan suatu model bahasa dalam memahami konteks yang diberikan. Semakin sedikit skor *perplexity* maka model bahasa dinyatakan tidak mengalami kebingungan dalam memahami konteks kalimat yang diberikan.



Gambar 2.21 Uji *perplexity* pada beberapa model bahasa [45].

Berdasarkan Gambar 2.21, *GPT-3* menunjukkan skor *perplexity* sebesar 20,5 (mendefinisikan seberapa baik model bahasa memprediksi probabilitas kata dalam sampel) di bawah model *zero-shot* di *Penn Tree Bank (PTB)* [45]. *GPT-3* menjadi model bahasa nomor satu dalam Pemodelan Bahasa di *Penn Tree Bank* dengan skor *perplexity* 20,5 dan juga memiliki akurasi 86,4% (meningkat 18% dari model yang sebelumnya) dalam model *few-shot learning* saat melakukan uji *dataset LAMBADA*. Untuk tes ini, model memprediksi kata terakhir dalam kalimat, dimana model diharuskan untuk "membaca" keseluruhan paragraf. Hasil uji *perplexity* yang menentukan kualitas model bahasa dipengaruhi oleh banyaknya parameter yang dimiliki oleh model bahasa tersebut, dengan begitu *GPT-3* memiliki parameter terbesar dibandingkan dengan model bahasa tersebut, dengan begitu *GPT-3* memiliki parameter terbesar dibandingkan dengan model bahasa yang lainnya. Pada Gambar 2.22 dan Gambar 2.23 menjelaskan bahwa *GPT-3* yang memiliki jumlah parameter sebanyak 175 miliar, menjadikan *GPT-3* menempati model bahasa nomor satu dengan memiliki jumlah parameter terbesar [46].



Gambar 2.22 Jumlah parameter beberapa model bahasa [46].

Rank	Model	Test perplexity	Validation perplexity	Params ↑ Training Data	Paper	Code	Result	Year	Tags
1	GPT-3 (Zero-Shot)	20.5		175000M ✓	Language Models are Few-Shot Learners	🔗	🔗	2020	
2	GPT-2	35.76		1542M ✓	Language Models are Unsupervised Multitask Learners	🔗	🔗	2019	
3	BERT-Large-CAS	31.3	36.1	395M ✓	Language Models with Transformers	🔗	🔗	2019	
4	AWD-LSTM-DOC x5	47.17	48.63	185M ✗	Direct Output Connection for a High-Rank Language Model	🔗	🔗	2018	LSTM
5	GL-LWGC + AWD-MoS-LSTM + dynamic eval	46.34	46.64	26M ✗	Gradual Learning of Recurrent Neural Networks	🔗	🔗	2018	LSTM

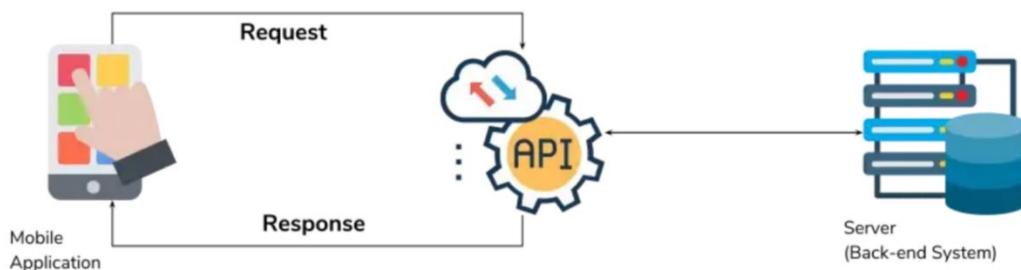
Gambar 2.23 GPT-3 sebagai model bahasa dengan parameter terbesar [46].

2.11 Application Programming Interface (API)

Application Programming Interface (API) memiliki peran sebagai perantara berbagai aplikasi yang berbeda-beda dalam satu *platform* yang sama atau *platform*

yang berbeda. *API* banyak digunakan sebagai perantara komunikasi dengan berbagai bahasa pemrograman yang berbeda yang akan memudahkan bagi pengembang (*developer*). *API* membantu *developer* agar tidak perlu menyediakan semua datanya secara mandiri, karena *API* yang akan mengambil data dari *platform* yang dibutuhkan. Dalam pengembangan suatu *website*, *developer* dapat melakukan integrasi layanan berbagai *platform* dengan menggunakan *API*, melalui *API key* yang disematkan dalam kode program yang dibuat oleh *developer*.

Cara kerja *API* dimulai dengan *user* yang mengakses suatu *web* atau aplikasi, kemudian *API* akan melakukan *request* ke *server* hingga *server* memberikan respon terhadap *API* dalam bentuk data atau informasi maupun layanan, dilanjutkan dengan *API* kembali meneruskan informasi dari *server* untuk ditampilkan di aplikasi yang diakses oleh *user*.



Gambar 2.24 Cara kerja *API* [47].

OpenAI menyediakan *API* yang dapat diakses oleh *developer* untuk melakukan beberapa tugas pemrosesan bahasa seperti pencarian semantik, ringkasan, analisis sentimen, membuat konten, terjemahan, dan beberapa tugas yang bisa dipersonalisasi. *API* akan bekerja dengan pemberian *prompt* teks apapun, kemudian *API* akan mengembalikan teks hasil pemrosesan ke aplikasi yang dibuat oleh *developer*.

2.12 Python

Python merupakan suatu bahasa pemrograman yang digunakan untuk pembuatan aplikasi, melakukan perintah komputer, maupun analisis data. Bahasa pemrograman *Python* memiliki definisi *high-level*, *interpreted*, dan *general*

purpose. Python sebagai bahasa pemrograman *high-level* yang berarti bahasa pemrograman tingkat tinggi yang diaplikasikan pada berbagai bagian seperti *IT*, *back-end developer*, hingga *data scientist*. Python sebagai bahasa *general-purpose* yang berarti bahasa pemrograman ini dapat digunakan untuk pembuatan aplikasi apapun untuk menyelesaikan berbagai tugas dan permasalahan. Python sebagai *interpreted-language* yang berarti bahwa kode pemrograman *python* diproses ketika *runtime*, sehingga tidak perlu kompilasi program sebelum menjalankannya.

2.13 Flask

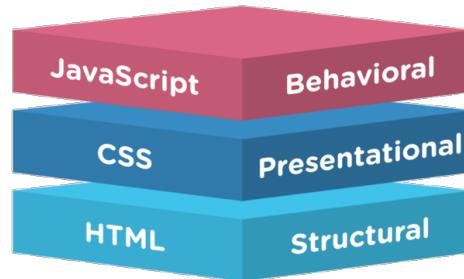
Pengembangan web (*web development*) merupakan proses pembangunan dan pemeliharaan situs web. Pengembangan web, erat kaitannya dengan pengembangan halaman (*page*) dalam web yang diakses, sehingga muncul alat yang dapat membantu tugas *web development*, salah satunya adalah *web framework* yang dibuat dengan bahasa pemrograman *python* yang disebut dengan *Flask*.

Flask adalah *library* yang dibuat oleh *Python* untuk pengembangan perangkat lunak (*software*) maupun pengelolaan *website* dalam berbagai kebutuhan. Seorang *developer* dapat menggunakan *flask* dan bahasa *python* untuk membuat *web* yang terstruktur dengan lebih mudah. *Flask* dapat berperan sebagai kerangka dalam pembuatan aplikasi atau pembuatan tampilan situs *web*. *Flask* tergolong jenis *microframework* karena tidak memerlukan alat atau *library* tertentu untuk pemakaiannya. *Microframework* yang diterapkan dalam *library flask* memiliki tujuan untuk membuat *core* aplikasi sesederhana mungkin dan tetap mudah untuk ditambahkan, agar meningkatkan nilai fleksibilitas dan skalabilitas dibanding *framework* lainnya. Selain ringan untuk dijalankan, *flask* dapat menerima fungsi *HTTP request* dengan mudah, mudah dalam pemasangan dan proses *deployment*-nya.

2.14 HyperText Markup Language (HTML)

HyperText Markup Language (HTML) merupakan kumpulan *script* yang digunakan untuk membuat halaman web agar menampilkan data bentuk teks maupun gambar

yang dirancang dalam web yang dibuat. *HyperText Markup Language (HTML)* adalah *markup* standar untuk dokumentasi suatu tampilan *browser* web. HTML dibantu oleh teknologi *Cascading Style Sheets (CSS)* dan bahasa *scripting* seperti *JavaScript*. Pembuatan *web* dengan bahasa pemrograman *python* dan *framework flask* memerlukan *template HTML* yang diletakkan dalam folder *templates* yang ada dalam direktori *python*. Proses pembuatan aplikasi atau *web* dengan struktur yang kompleks dan komponen dinamis yang beragam akan membutuhkan penggunaan *templates* dalam bentuk *layout HTML*.



Gambar 2.25 Hubungan penggunaan *JavaScript* dan *CSS* dalam *HTML* [48].

2.15 *JavaScript (JS)*

JavaScript merupakan bahasa *scripting* yang bekerja dalam banyak *browser* seperti *Internet Explorer*, *Mozilla*, *Firefox*, *Netscape*, *Opera*. Kegunaan *JavaScript* terdapat pada halaman web untuk meningkatkan kualitas desain web, memvalidasi form, mendeteksi *browser*, dan membuat *cookies*. *JavaScript* tidak dapat dijalankan sendiri tanpa didasari oleh *HTML*. *JavaScript* mengatur logika agar tampilan *web* lebih dinamis.

2.16 *Cascading Style Sheets (CSS)*

Cascading Style Sheet (CSS) adalah bahasa *style sheet* yang mengatur tampilan dengan mengendalikan beberapa komponen web agar terstruktur dan seragam. CSS mengendalikan gambar, warna teks atau tabel, besar ukuran *border* dan warnanya, tampilan *hyperlink*, spasi antar segmen, maupun parameter lainnya. Penggunaan CSS memungkinkan *developer* untuk menampilkan suatu halaman yang sama

dengan format yang berbeda. *CSS* tidak dapat dijalankan sendiri, melainkan harus dalam struktur pemrograman lain, salah satunya adalah *HTML*.

2.17 PythonAnywhere

PythonAnywhere adalah *environment* berbasis *online cloud PaaS (platform-as-a-service)* untuk layanan *website hosting* berdasarkan bahasa pemrograman *python*. *PythonAnywhere* didirikan oleh Giles Thomas dan Robert Smithson pada tahun 2012, yang menyediakan akses dalam *browser* ke *interface* dengan baris program *python* dan *bash* berbasis *server* dengan editor kode melalui penyorotan sintaks.

2.18 Website & Mobile Application

Aplikasi web merupakan program yang dibuat dengan memanfaatkan *browser* web dan teknologi web untuk mengerjakan suatu tugas melalui jaringan internet. Aplikasi web juga termasuk *software* yang dikembangkan dan berjalan di sisi *client* yang membutuhkan *web server* yang menggunakan bahasa pemrograman misalnya *HTML*, *JavaScript*, *CSS*, maupun bahasa lainnya yang membutuhkan *browser* web untuk menjalankannya seperti *Chrome*, *Firefox* atau *Opera*.

Aplikasi *mobile* atau *mobile apps* merupakan aplikasi yang dibangun untuk beroperasi di perangkat bergerak (*mobile*), misalnya *smartphone* dan *tablet* serta memiliki *Operating System (OS)* yang mendukung *software* secara mandiri. Pendistribusian aplikasi *mobile* dapat melalui *platform* seperti *Apple App*, *Google Play*, maupun *Windows Phone*. Aplikasi *mobile* dapat terpasang di perangkat *mobile* melalui sistem pengunduhan (*download*) yang disediakan oleh *platform* distribusi aplikasi *mobile*. Selain aplikasi berbasis web, aplikasi akan lebih mudah dicapai oleh lebih banyak *user* jika dibuat versi aplikasi *mobile*-nya. Hal ini sesuai dengan data statistik global oleh statcounter bahwa mulai bulan Mei 2022, lebih dari 60% lalu lintas web di seluruh dunia berasal dari perangkat seluler [49]. Selain itu, pengguna perangkat seluler dapat mengunduh dan memasang aplikasi ke perangkat mereka sehingga memungkinkan aplikasi android memberikan pengalaman yang lebih cepat dan lebih responsif dibanding mengakses *Uniform*

Resource Locator (URL) situs web pada perangkat seluler. Aplikasi android yang dibuat menggunakan fungsi *web view*, dimana ketika aplikasi dibuka akan diarahkan *menuju* ke situs web dengan tampilan *mobile*. Hal ini dilakukan karena memberikan keuntungan bahwa pengguna yang memasang aplikasi tidak perlu memperbarui atau menghabiskan waktu menginstal versi baru jika ada pembaruan atau peningkatan pada aplikasi web yang dibuat.

2.19 Google Playstore

Google Play store merupakan platform layanan distribusi aplikasi digital berbasis sistem android yang dikembangkan oleh *Google*. *Google Play store* memungkinkan *user* untuk melakukan penelusuran serta pengunduhan aplikasi yang disediakan oleh *platform* dengan *Software Development Kit (SDK)* yang diterbitkan oleh *Google*. Terdapat dua jenis aplikasi tersedia dalam *Google Play store*, yaitu aplikasi berbayar dan aplikasi gratis, kedua aplikasi dapat diunduh secara langsung dengan perangkat android yang tersedia di aplikasi *play store*.

2.20 Indikator Keberhasilan

Aplikasi yang dibuat harus melalui tahap analisis melalui berbagai cara yang dilakukan untuk mengukur kinerja yang diperoleh dari model aplikasi selama pengembangan. Analisis yang dilakukan untuk mengukur keberhasilan aplikasi yang dibuat terdiri dari beberapa metode analisis, yaitu:

- a) *Blackbox Testing*
- b) *ROUGE Metrics*
 - *ROUGE-1 : Precision, Recall, F1-score*
 - *ROUGE-2 : Precision, Recall, F1-score*
 - *ROUGE-L : Precision, Recall, F1-score*
- c) *BERTscore*
 - *Precision*
 - *Recall*
 - *F1-score*
- d) *Rating Google Playstore*

2.20.1 Blackbox Testing

Blackbox testing merupakan salah satu metode pengujian aplikasi yang dilakukan dengan cara pengamatan *input* dan hasil *output* aplikasi tersebut tanpa memperhatikan struktur kode dari aplikasi tersebut dan melaporkan apakah aplikasi dapat berfungsi dengan baik. Teknik *blackbox testing* yang akan diterapkan oleh peneliti yaitu *all-pair testing* atau *pairwise testing* dengan menguji semua probabilitas kombinasi dari seluruh hubungan antar bagian aplikasi berdasarkan *input* parameternya. Jika hasil pengujian parameter lebih didominasi nilai yang berhasil, maka nilai pengujian *blackbox testing*-nya baik.

2.20.2 ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*)

ROUGE atau *Recall-Oriented Understudy for Gisting Evaluation* merupakan kelompok yang terdiri dari beberapa metrik *ROUGE*. Metrik evaluasi *ROUGE* kelompok metode evaluasi yang sering digunakan pada sistem *Natural Language Generation (NLG)* dengan mengukur tingkat kesesuaian atau *matching* antara teks yang dihasilkan mesin dan teks yang ditulis oleh manusia (*ground-truth*).

Berdasarkan pada penelitian mengenai survei evaluasi beberapa metrik evaluasi yang baik di tugas *NLP*, lebih tepatnya tugas *summarization* (peringkasan). Didapatkan tabel referensi metrik evaluasi yang bisa digunakan sebagai acuan metrik evaluasi untuk setiap bidang *NLP* seperti pada Gambar 2.26.

Metric	Property	MT	IC	SR	SUM	DG	QG	RG
F-SCORE	precision and recall	✓	✓	✓	✓	✓	✓	✓
BLEU	<i>n</i> -gram precision	✓	✓			✓	✓	✓
METEOR	<i>n</i> -gram w/ synonym match	✓	✓			✓		
CIDER	<i>tf-idf</i> weighted <i>n</i> -gram sim.			✓				
NIST	<i>n</i> -gram precision		✓					
GTM	<i>n</i> -gram metrics		✓					
HLEPOR	unigrams harmonic mean			✓				
RIBES	unigrams harmonic mean							
MASI	attribute overlap							
WER	% of insert, delete, replace					✓		
TER	translation edit rate			✓				
ROUGE	<i>n</i> -gram recall				✓	✓		
DICE	attribute overlap							

Gambar 2.26 Referensi metrik evaluasi [50].

Keterangan Gambar 2.26 yaitu **MT**: *Machine Translation*, **IC**: *Image Captioning*, **SR**: *Speech Recognition*, **SUM**: **Summarize**, **DG**: **Document Generation**, **QG**: *Question Generation*, **RG**: *Response Generation*. Aplikasi yang dirancang oleh peneliti memiliki 12 Fitur, dimana terdapat dua fitur dalam aplikasi tersebut yang mengaplikasikan penggunaan *NLP* untuk tugas *summarizing* atau meringkas yaitu *Paragraph Summary* dan *Keyword Extractor*. Satu fitur yang mengaplikasikan *Document Generation*, yaitu *AI Sentence Corrector*. Kedua bidang yang diaplikasikan ini termasuk ke dalam tugas *NLP* yang termasuk dalam Gambar 2.27. Sehingga untuk fitur *Paragraph Summary*, *Keyword Extractor*, dan *AI Sentence Corrector* akan diukur keberhasilannya dengan menggunakan metrik evaluasi *ROUGE*.

Dalam penelitian ini, metrik *ROUGE* yang digunakan yaitu *ROUGE-1*, *ROUGE-2*, dan *ROUGE-L*. *ROUGE-N* mengukur jumlah pencocokan '*n-gram*' antara teks yang dihasilkan model bahasa dan teks referensi yang dibuat oleh manusia. *N-gram* merupakan rangkaian kata yang ada dalam suatu kalimat atau dokumen. Sebuah *unigram* (1-gram) akan terdiri dari satu kata. Sedangkan *bigram* (2 gram) terdiri dari dua kata berurutan. Dengan *ROUGE-N*, *N* mewakili *n-gram* yang digunakan. Untuk *ROUGE-1* berarti akan dilakukan pengukuran tingkat kecocokan *unigram* antara keluaran model dan referensi yang dibuat. Begitupun pada *ROUGE-2* akan menggunakan *bigram*. Metrik *ROUGE-L* mengukur urutan umum terpanjang atau *Longest Common Subsequence (LCS)* antara keluaran model dan referensi yang dibuat manusia. Hal ini berarti bahwa metrik ini menghitung urutan token terpanjang yang dibagikan di antara kedua data teks yang diuji. Setiap metrik *ROUGE* yang digunakan, akan menghasilkan nilai *precision*, *recall*, dan juga *F1-score*.

Metrik *ROUGE* memberikan beberapa metode evaluasi otomatis yang mengukur kesamaan antara ringkasan [51]. *ROUGE* dapat diaplikasikan untuk melakukan pengujian dengan membuat kode program (*python*) yang menggunakan *library ROUGE*, yaitu dengan cara menginstall *library ROUGE* pada terminal program.

Library Python ROUGE bersifat *open-source*, sehingga dapat dijalankan pada semua program *python*.

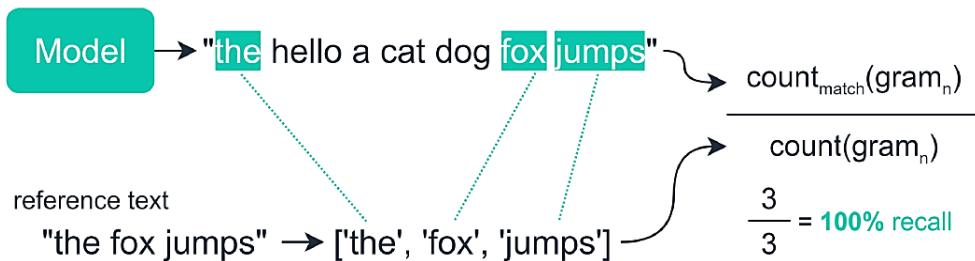
Dalam pengaplikasiannya, peneliti menggunakan *ROUGE* dengan meng-*install library python ROUGE* pada kode program yang dibuat, kemudian menambahkan dua sumber teks yang akan diuji, yaitu teks yang dihasilkan oleh model bahasa *GPT-3* dan teks yang dihasilkan oleh aplikasi perangkum teks yang lainnya seperti *quillbot* yang dapat diakses dengan akun premium berbayar. Teks yang dihasilkan oleh model bahasa *GPT-3* yang sudah di-*training* dikelompokkan dalam variabel “*predictions*”, dan teks yang dihasilkan oleh aplikasi pembuat teks lainnya dimasukkan kedalam variabel “*reference*”. Kedua variabel tersebut akan dibandingkan oleh *library ROUGE* untuk mengukur tingkat kesamaan kata hasil dari aplikasi yang dibuat dengan aplikasi yang lainnya. Metrik *ROUGE* digunakan untuk mengukur kesamaan hasil pada fitur *Paragraph Summary*, *AI Sentence Corrector*, dan *Keyword Extractor*. Pengukuran kemiripan dengan menggunakan *ROUGE* pada ketiga fitur yang dibuat dilakukan dengan tujuan untuk melihat kinerja dari aplikasi yang dibuat apakah memberikan hasil peringkasan teks dan pengambilan *keyword* yang kualitasnya setara dengan aplikasi pembuat teks berbayar yang sudah diakui kinerjanya. Hasil dari pengukuran dengan metrik *ROUGE* ini memberikan tiga nilai yaitu *precision*, *recall*, dan *F1-score*.

a. *Recall*

Recall menghitung jumlah *n-gram* yang tumpang tindih yang ditemukan di kedua data teks yaitu data hasil keluaran model bahasa (mesin) dan data teks referensi, kemudian membagi angka tersebut dengan jumlah total *n-gram* dalam referensi. Perumusannya seperti berikut ini :

$$\frac{\text{number of } n\text{-grams found in model and reference}}{\text{number of } n\text{-grams in reference}} \dots \dots \dots \quad (2.1)$$

Nilai *recall* bagus untuk memastikan model yang dibuat mampu menangkap semua informasi yang terkandung dalam referensi.



Gambar 2.27 Penghitungan *recall* dalam pemodelan bahasa [52].

b. *Precision*

Nilai presisi dihitung dengan cara yang hampir sama dengan *recall*, namun perbedannya terletak pada hitungan pembagian yang dilakukan. Jika pada *recall* jumlah *n-gram* yang sama dibagi dengan jumlah total *n-gram* dalam referensi, pada *precision* jumlah *n-gram* yang sama dibagi dengan jumlah *n-gram* yang dihasilkan oleh model mesin.

$$\frac{\text{number of } n - \text{grams found in model and reference}}{\text{number of } n - \text{grams in model}} \dots \dots \dots \dots \dots \dots \dots \quad (2.3)$$

Jika contoh sebelumnya dihitung nilai presisinya, maka didapatkan hasil sebagai berikut.



Gambar 2.28 Penghitungan *precision* dalam pemodelan bahasa [52].

c. *F1-score*

Nilai *FI-score* menggabungkan nilai presisi dan *recall* menjadi satu metrik dengan mengambil rata-rata. *FI-score* digunakan untuk membandingkan kinerja dua model yang misalnya model A memiliki nilai *recall* yang lebih tinggi, namun model B memiliki nilai presisi yang lebih tinggi, maka pilihannya adalah melihat skor *FI-score* dari kedua model untuk menentukan mana yang memberikan hasil lebih baik.

Persamaan untuk menghitung *F1-score* yaitu :

Jika diaplikasikan pada contoh sebelumnya, maka didapatkan nilai *F1-score* sebagai berikut.



$$2 * \frac{0.43 * 1.0}{0.43 + 1.0} = 0.6$$

Gambar 2.29 Penghitungan *F1-score* dalam pemodelan bahasa [52].

2.20.3 BERTscore

BERTscore merupakan metrik evaluasi otomatis yang ditujukan untuk menguji kemampuan sistem pembuatan teks (*text generator*). Penggunaan *BERTscore* berbeda daripada komputasi kesamaan sintaks (*syntactical similarity*), *BERTscore* lebih berfokus pada komputasi untuk menilai kesamaan semantik (*semantic similarity*) antara token referensi dengan teks yang dihasilkan mesin. Sederhananya, *BERTscore* menghitung kesamaan makna dari teks yang dihasilkan oleh mesin dengan teks referensi yang disiapkan walaupun kedua teks memiliki kata yang

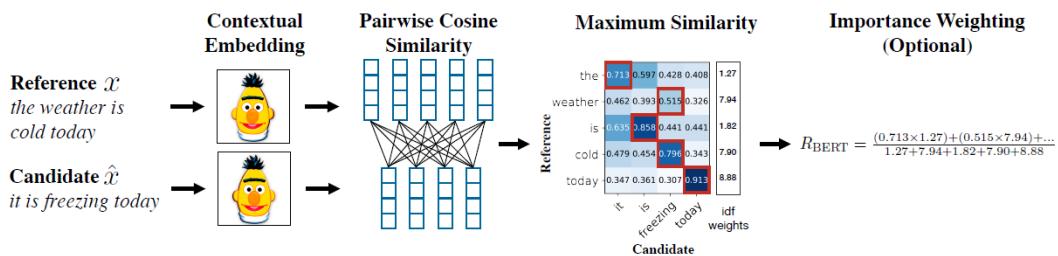
berbeda namun maknanya sama, berbeda dengan metrik lain seperti *ROUGE* dan *BLEU* yang bergantung pada konsep *n-gram overlap* atau tumpang tindih sintaks kata yang sama seperti *unigram* atau *bigram*. *BERTscore* muncul untuk menjawab keterbatasan metrik evaluasi sebelumnya dengan cara memahami arti dari apa yang dihasilkan oleh mesin dan apa yang seharusnya dihasilkan sebagai perbandingan. *BERTscore* menghitung skor kesamaan untuk setiap token dalam kalimat kandidat dengan setiap token dalam kalimat referensi. Namun, alih-alih kecocokan persis, penghitungan kesamaan token menggunakan penyematan kontekstual (*contextual embeddings*) [53]. Hasil dari evaluasi metrik *BERTscore* yaitu nilai *precision*, *recall*, dan *F1-score*.

BERTscore berfungsi sebagai metrik evaluasi baru yang menghitung kesamaan makna suatu kalimat yang dihasilkan oleh mesin model bahasa dengan teks referensi dengan cara mencocokkan kata-kata dalam kalimat yang dihasilkan oleh model bahasa dan referensi melalui kesamaan kosinus (*cosine similarity*) [53]. Konsep kesamaan kosinus (*cosine similarity*) adalah metrik yang digunakan untuk mengukur seberapa mirip dokumen terlepas dari ukurannya maupun banyaknya token kata. Beberapa fitur aplikasi yang lainnya seperti *Product Description*, *Job Description*, *Social Media Captions*, *Email Templates*, *Advertisement*, *Business Pitch*, *Youtube Idea*, *Video Description*, *Key Points* dan *Paragraph Summary* akan menggunakan metrik evaluasi *BERTscore*.

BERTscore dapat diaplikasikan untuk melakukan pengujian dengan membuat kode program (*python*) yang menggunakan *library BERT-score*, yaitu dengan cara menginstall *library BERTscore* pada terminal program. *Library Python BERTscore* bersifat *open-source*, sehingga dapat dijalankan pada semua program *python*. Dalam pengaplikasianya, peneliti menggunakan *BERTscore* dengan menginstall *library python BERT-score* pada kode program yang dibuat, kemudian menambahkan dua sumber teks yang akan diuji, yaitu teks yang dihasilkan oleh model bahasa *GPT-3* dan teks yang dihasilkan oleh aplikasi pembuat teks yang lainnya seperti *copy.ai* dan *ryte.me* yang dapat diakses dengan akun premium berbayar. Teks yang dihasilkan oleh model bahasa *GPT-3* yang sudah di-training

dikelompokkan dalam variabel “*predictions*”, dan teks yang dihasilkan oleh aplikasi pembuat teks lainnya dimasukkan kedalam variabel “*reference*”. Kedua variabel tersebut akan dibandingkan oleh *library BERTscore* untuk mengukur tingkat kesamaan hasil dari aplikasi yang dibuat oleh penulis dengan aplikasi yang sudah ada sebelumnya, dengan tujuan untuk melihat kinerja dari aplikasi yang dibuat apakah memberikan hasil yang kualitasnya setara dengan aplikasi pembuat teks berbayar lainnya. Hasil dari pengukuran dengan metrik *BERTscore* terdapat tiga nilai yang ditampilkan yaitu *precision*, *recall*, dan *F1-score*.

Mekanisme *BERTscore* yaitu kalimat referensi sebagai $x = \langle x_1, \dots, x_k \rangle$ dan kalimat yang diuji sebagai $\hat{x} = \langle \hat{x}_1, \dots, \hat{x}_l \rangle$, dengan menggunakan *contextual embeddings* untuk merepresentasikan token kata dan mengkomputasi kesamaan dengan *cosine similarity*, sehingga secara opsional dapat dihasilkan bobot skor kemiripannya.



Gambar 2.30 Arsitektur *BERTscore* [53].

Ilustrasi pada Gambar 2.30 menunjukkan contoh komputasi dari metrik *recall* dalam *BERTscore* yang disebut R_{BERT} . Dengan kalimat referensi sebagai x dan kalimat yang diuji sebagai \hat{x} , dihitung penyematan *BERT* (*BERT embeddings*) dan pasangan *cosine similarity* yaitu kemiripan konteks antar kata, kemudian menyorot skor tertinggi (*highlight* warna merah) dari penghitungan *similarity maximum* dan menyertakan pembobotan skornya.

Token referensi sebagai $x = \langle x_1, \dots, x_k \rangle$, dan model penyematan akan menghasilkan urutan vektor $\langle x_1, \dots, x_k \rangle$. Dengan prinsip yang sama, token kandidat sebagai $\hat{x} = \langle \hat{x}_1, \dots, \hat{x}_l \rangle$ akan dipetakan ke vektor $\langle \hat{x}_1, \dots, \hat{x}_m \rangle$. Model utama *BERTscore* akan melakukan tokenisasi teks *input* ke bentuk urutan kata, dimana

kata-kata yang tidak diketahui dibagi menjadi beberapa urutan karakter. Representasi untuk setiap kata dikomputasikan oleh *encoder Transformer* dengan mengaplikasikan *self-attention* secara berulang dan transformasi non-linear.

Representasi vektor memungkinkan ukuran kemiripan (*similarity measure*) yang lebih lembut dibandingkan pencocokan kata yang benar-benar tepat. *Cosine similarity* dari token referensi x_i dan token kandidat \hat{x}_j adalah $\frac{x_i^T \hat{x}_j}{\|x_i\| \|\hat{x}_j\|}$. Dengan menggunakan vektor pra-normalisasi, yang mengurangi perhitungan ini menjadi lebih sederhana yaitu $x_i^T \hat{x}_j$. Sementara pengukuran kemiripan (*similarity measure*) mempertimbangkan token secara terpisah, penyematan kontekstual (*contextual embeddings*) mengandung informasi dari kalimat.

Skor kecocokan secara lengkap dari setiap token dalam x pada token \hat{x} dihitung sebagai *recall*, dan setiap token dalam \hat{x} pada token x dihitung sebagai *precision*. Penghitungan ini menggunakan pencocokan maksimum untuk memaksimalkan skor kesamaan pada token yang cocok, dimana setiap token dicocokkan dengan token yang paling mirip di kalimat lainnya. Kombinasi dari *precision* dan *recall* dihitung sebagai pengukuran *F1 (F1-score)*. Untuk token referensi x dan token kandidat \hat{x} , maka nilai *precision*, *recall*, dan *F1-score* yaitu [53] :

a. *Recall*

Recall adalah fraksi dari contoh positif yang diberi label dengan benar oleh model sebagai positif. Persamaan matematis dalam penghitungan *recall* pada *BERTscore* yaitu:

Keterangan :

$$R_{BERT} = Recall BERTscore$$

x = Token referensi

\hat{x} = Token kandidat (yang diuji)

x_i = urutan vektor x

\hat{x}_j = urutan vektor \hat{x}

$\sum_{x_i \in x}$ = jumlah x_i yang ada dalam x

$\max_{\hat{x}_j \in \hat{x}}$ = nilai maksimum \hat{x}_j yang ada dalam \hat{x}

$x_i^\top \hat{x}_j$ = cosine similarity x dan \hat{x}

b. *Precision*

Presisi adalah bagian dari contoh positif yang diberi label dengan benar dari semua contoh yang diberi label positif. Persamaan matematis dalam penghitungan *precision* pada *BERTscore* yaitu:

Keterangan :

$$P_{BERT} = Precision BERTscore$$

x = Token referensi

\hat{x} = Token kandidat (yang diuji)

x_i = urutan vektor x

\hat{x}_j = urutan vektor \hat{x}

$\sum_{\hat{x}_j \in \hat{x}}$ = jumlah \hat{x}_j yang ada dalam \hat{x}

$\max_{x_i \in x}$ = nilai maksimum x_i yang ada dalam x

$x_i^\top \hat{x}_j$ = cosine similarity x dan \hat{x}

c. *F1-score*

Nilai *F1-score* menggabungkan nilai presisi dan *recall* menjadi satu metrik dengan mengambil rata-rata. Persamaan matematis dalam penghitungan *F1-score* pada *BERTscore* yaitu:

$$F_{BERT} = F1\text{-score } BERTscore$$

$P_{BERT} = Precision BERTscore$

$$R_{BERT} = Recall BERTscore$$

2.20.4 Rating Aplikasi Playstore

Rating aplikasi *playstore* merupakan salah satu cara untuk melakukan uji lapangan pada aplikasi yang sudah dibuat. Dengan adanya *rating* dan ulasan yang didapatkan pada *platform Google Play store* berguna untuk menentukan seberapa bermanfaat aplikasi tersebut. Peneliti memilih parameter ini untuk menguji kelayakan aplikasi yang telah digunakan oleh *user* dari berbagai kalangan. Tidak ada kriteria baik buruknya *rating* aplikasi pada *playstore*, namun selama aplikasi tersebut mendapatkan banyak ulasan bagus dan nilai *rating* aplikasinya diatas 4.5 bintang (dari nilai maksimal 5 bintang) maka aplikasi tersebut sudah dapat dianggap berhasil melalui uji lapangan.

Ada beberapa aspek yang membantu pengembang aplikasi untuk mendapatkan rating yang baik melalui kepuasan *user*, yaitu dengan melakukan optimasi aplikasi dan mengajak *user* untuk memberikan ulasan agar aplikasi menjadi lebih baik. Dengan adanya ulasan dan saran oleh *user*, akan membantu *developer* untuk semakin mengembangkan sistem aplikasi dan memperbaiki kekurangan pada aplikasi yang dibuat.

BAB III

METODOLOGI PENELITIAN

3.1 Tempat dan Waktu Penelitian

Penelitian dimulai pada Juli 2022 sampai dengan Oktober 2022. Penelitian ini dilakukan di Laboratorium Terpadu Jurusan Teknik Elekro, Fakultas Teknik, Universitas Lampung.

3.2 Alat dan Bahan

Alat dan bahan yang digunakan pada penelitian tugas akhir ini adalah sebagai berikut :

3.2.1 Perangkat Keras (*Hardware*)

1. Satu unit laptop sebagai media perancangan sistem *web* dan aplikasi yang dibangun dengan sistem operasi Windows 11 Pro 64-bit, dengan spesifikasi minimum :
 - a) Processor AMD Ryzen 5 3500U / Intel Core i5
 - b) RAM DDR4 berkapasitas 8 GB
 - c) SSD NVMe berkapasitas 500 GB
2. Smartphone Android untuk *testing* / simulasi aplikasi android

3.2.2 Perangkat Lunak (*Software*)

1. *Software Visual Studio Code* sebagai *text editor* dan *running script program*.
2. *Command Prompt (CMD)* untuk *running script program*.
3. *Software Figma* untuk pembuatan *prototype* desain *interface website*.
4. *Software Adobe Illustrator* untuk pembuatan animasi *element website*.

5. *Software Android Studio* untuk pembuatan aplikasi berbasis *mobile android*.

6. *Software Jupyter Notebook* untuk melakukan analisis evaluasi metrik

Adapun bahasa pemrograman beserta modul *library* yang digunakan yaitu :

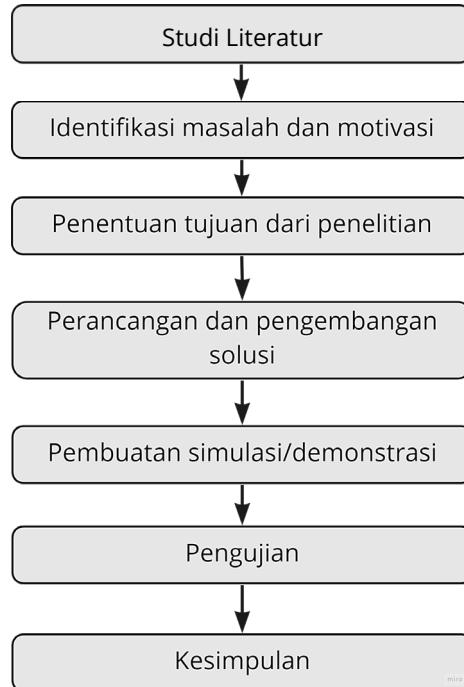
- a) *Python*
- b) *HTML*
- c) *CSS*
- d) *JavaScript*
- e) *Flask (framework)*
- f) *OpenAI (library)*
- g) *os (python module)*
- h) *ROUGE (library)*
- i) *BERTscore (library)*
- j) *Evaluate (library)*

3.2.3 Subscription Account

1. *OpenAI Developer Subscription* untuk kode *API Developer* yang dimasukkan ke program
2. *PythonAnywhere Subscription* untuk *web deployment*
3. *Google Play Console* untuk perilisan aplikasi android.

3.3 Metode Penelitian

Metode penelitian yang dilakukan terdiri dari studi literatur, identifikasi masalah, pembuatan simulasi, pengujian sistem, pembahasan, dan pengambilan kesimpulan. Berdasarkan konsep yang diterapkan, maka metode penelitian disesuaikan dengan memiliki tahapan studi literatur, identifikasi masalah, fokus penelitian, perancangan pembuatan *prototype* aplikasi, demonstrasi aplikasi, pengujian aplikasi, analisis capaian, pelaporan hasil penelitian. Berikut bagan alir penelitian yang dilakukan. Konsep metode penelitian yang diterapkan dapat dilihat pada Gambar 3.2.



Gambar 3.2 Konsep Metodologi Penelitian

1. Studi Literatur

Metode ini merupakan teknik mengumpulkan referensi dan pustaka pendukung yang akan digunakan sebagai bahan acuan mengenai objek yang diteliti. Studi dilakukan terhadap jurnal penelitian internasional, tesis dan disertasi nasional dan internasional. Peneliti melakukan analisis, interpretasi, dan generalisasi fakta-fakta dari literatur yang didapatkan. Beberapa referensi utama yang menjadi acuan dalam penelitian ini antara lain:

- Vaswani [1] dalam penelitian “*Attention is All you Need, Advances in neural information processing systems*”.
- J. Devlin [36] dalam penelitian “*BERT: Pre-training of deep bidirectional Transformers for language understanding*”.
- Radford [54] dalam penelitian “*Language Models are Unsupervised Multitask Learners*”.
- T. B. Brown [38] dalam penelitian “*Language models are few-shot learners*”.

2. Identifikasi Masalah dan Motivasi

Identifikasi masalah dalam penelitian ini dimulai dari maraknya penggunaan situs *e-commerce* yang semakin berkembang, memunculkan suatu permasalahan yaitu adanya persaingan pada proses pemasarannya. Pemasaran digital perlu ditingkatkan untuk menghadapi kondisi bisnis saat ini dengan meningkatkan peranan *copywriter* untuk *menunjang* konten penjualan produk yang baik. Selama ini teknik *copywriting* hanya dilakukan secara manual oleh para ahli dibidang penulisan kreatif seperti *copywriter*. Sehingga tidak semua orang bisa memiliki dasar untuk membuat konten penulisan yang baik dan menarik. Beberapa usaha kecil seperti UMKM masih perlu banyak pertimbangan dalam mengeluarkan biaya untuk jasa *copywriter*. Suatu bidang usaha yang bergerak di bisnis digital pastinya membutuhkan kemampuan *copywriting*. Solusi untuk meningkatkan kemampuan *copywriting* bagi orang awam, salah satunya dengan membuat aplikasi yang mampu menyediakan ide kalimat yang menarik agar memudahkan *copywriter* menyusun kalimat dengan memanfaatkan *Artificial Intelligence (AI)*.

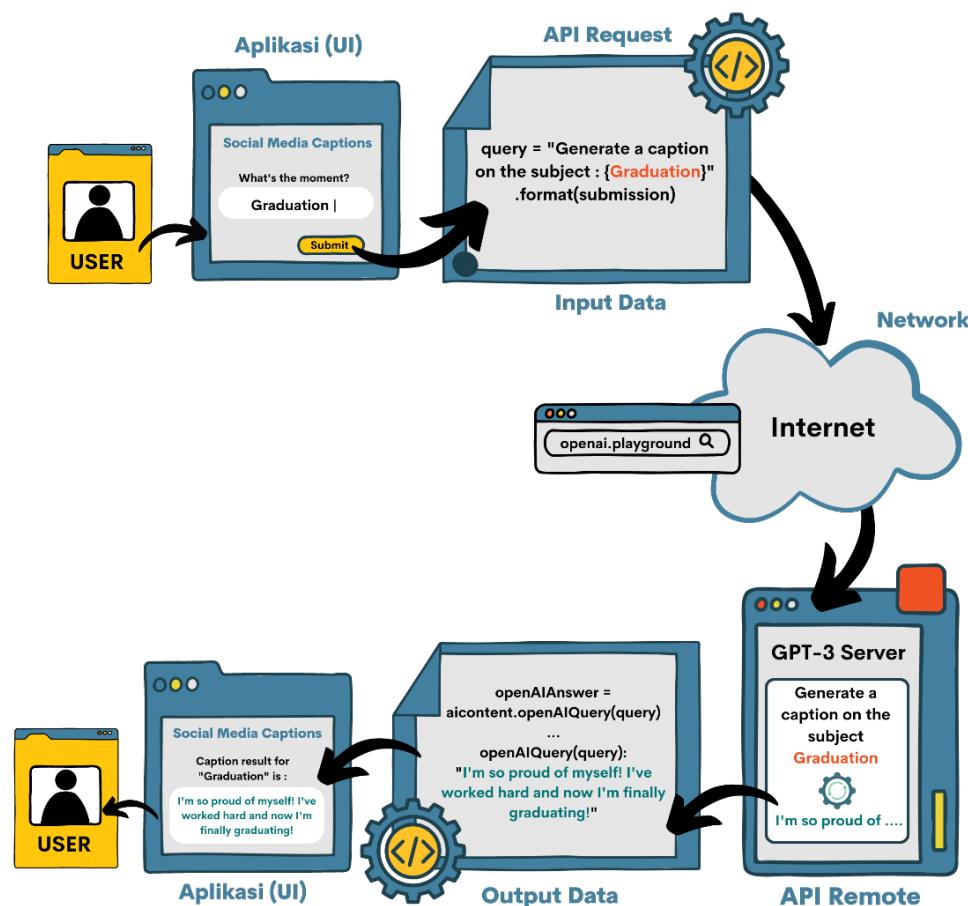
3. Penentuan Fokus dari Penelitian

Penentuan fokus ditentukan berdasarkan hasil identifikasi masalah dan motivasi yang mendorong dilakukannya penelitian. Fokus penelitian adalah perancangan dan pembuatan aplikasi web dan aplikasi berbasis android yang bisa diakses oleh berbagai lapisan masyarakat, terutama bisnis kecil yang belum memiliki tenaga *copywriter*, sehingga aplikasi ini dapat dimanfaatkan sebagai mesin *copywriting* untuk meningkatkan efektivitas bisnis. Aplikasi web dan aplikasi berbasis android yang dibuat dapat dimanfaatkan sebagai mesin *copywriting* untuk meningkatkan efektivitas bisnis. Aplikasi yang dirancang juga memuat berbagai fitur yang mendukung seperti deskripsi produk/jasa, deskripsi pekerjaan, *caption* sosial media, template *email marketing*, iklan, ide promosi bisnis. Tanpa meninggalkan kebutuhan akan metode *marketing* yang lebih menarik,

dibangun juga fitur untuk menghasilkan deskripsi yang baik untuk video promosi, ide judul video youtube, perbaikan susunan kata dan kalimat, peringkasan paragraf, pembuat *keyword* dari suatu deskripsi agar konten masuk di pencarian global untuk meningkatkan *insight* konten promosi.

4. Perancangan dan Pembuatan Aplikasi

Perancangan solusi berdasarkan fokus dari penelitian dikerjakan dengan metode pengembangan aplikasi *copywriting* dalam melakukan desain sistem dan pembuatan aplikasi.



Gambar 3.3 Rancangan implementasi *API GPT-3* pada aplikasi

Berdasarkan Gambar 3.3, implementasi model bahasa *GPT-3* dalam aplikasi yang dibuat yaitu dengan melakukan *training* pada model bahasa dan menggunakan kode *API* atau layanan dari model bahasa ini kedalam kode program web yang telah dibuat sebelumnya. Sehingga prinsip kerja

utamanya adalah aplikasi web yang dibuat akan menjadi penghubung antara layanan *API* model bahasa dengan *user* melalui *User Interface* (UI), dimana tampilan *website* akan menerima *input* dari *user* kemudian mengambil *inputan* tersebut dan dijadikan sebagai *prompt* pada layanan model bahasa *GPT-3* yang kemudian hasil dari model bahasa akan diterima oleh web tersebut dan ditampilkan kembali ke tampilan web yang terlihat oleh *user*.

Komponen pembangun aplikasi yang dibuat yaitu dengan menggunakan *API* dari *OpenAI GPT-3* yang disematkan ke kode program. Kode *API* disediakan oleh *OpenAI* melalui akun *developer* milik penulis. Pembuatan aplikasi secara keseluruhan dilakukan dengan *software Visual Studio Code* dengan referensi dari berbagai sumber seperti *GitHub*, *StackOverFlow*, *Youtube*, *OpenAI*, *arXiv*, *Papers with Code*, dan lainnya. Kode program yang digunakan yaitu *Python 3.8* dengan menggunakan beberapa *library* pendukung seperti *Flask* dan *OpenAI*. Untuk tampilan web (*frontend*) menggunakan bahasa pemrograman *HTML*, *CSS*, dan *JavaScript*. Dalam pembuatan desain *layout website*, digunakan aplikasi *Adobe Illustrator* untuk membuat detail komponen grafis, dan penyusunan *prototype* desain *layout website* dengan menggunakan aplikasi *Figma*. *Testing* aplikasi sebelum *deployment* dilakukan di *localhost* dengan *running script program* pada *Command Prompt* atau *Visual Studio Code* yang terbuka pada *browser Chrome*. Setelah aplikasi sudah berjalan dengan baik di *localhost*, maka folder program di-*deploy* ke *website PythonAnywhere* sehingga *website* memiliki *URL* yang dapat diakses berbagai *device*. Pembuatan aplikasi android dilakukan dengan *Android Studio*, dengan mengkonversikan *layout website* ke bentuk *mobile*. Perilisan aplikasi yang berbasis android dilakukan melalui akun *developer* penulis di *platform Google Play store*.

5. Demonstrasi

Berdasarkan rancangan solusi yang dibuat, demonstrasi dibangun dengan tujuan menguji aplikasi yang dibuat untuk melihat kesesuaian rancangan dengan harapan yang ingin dicapai

6. Pengujian

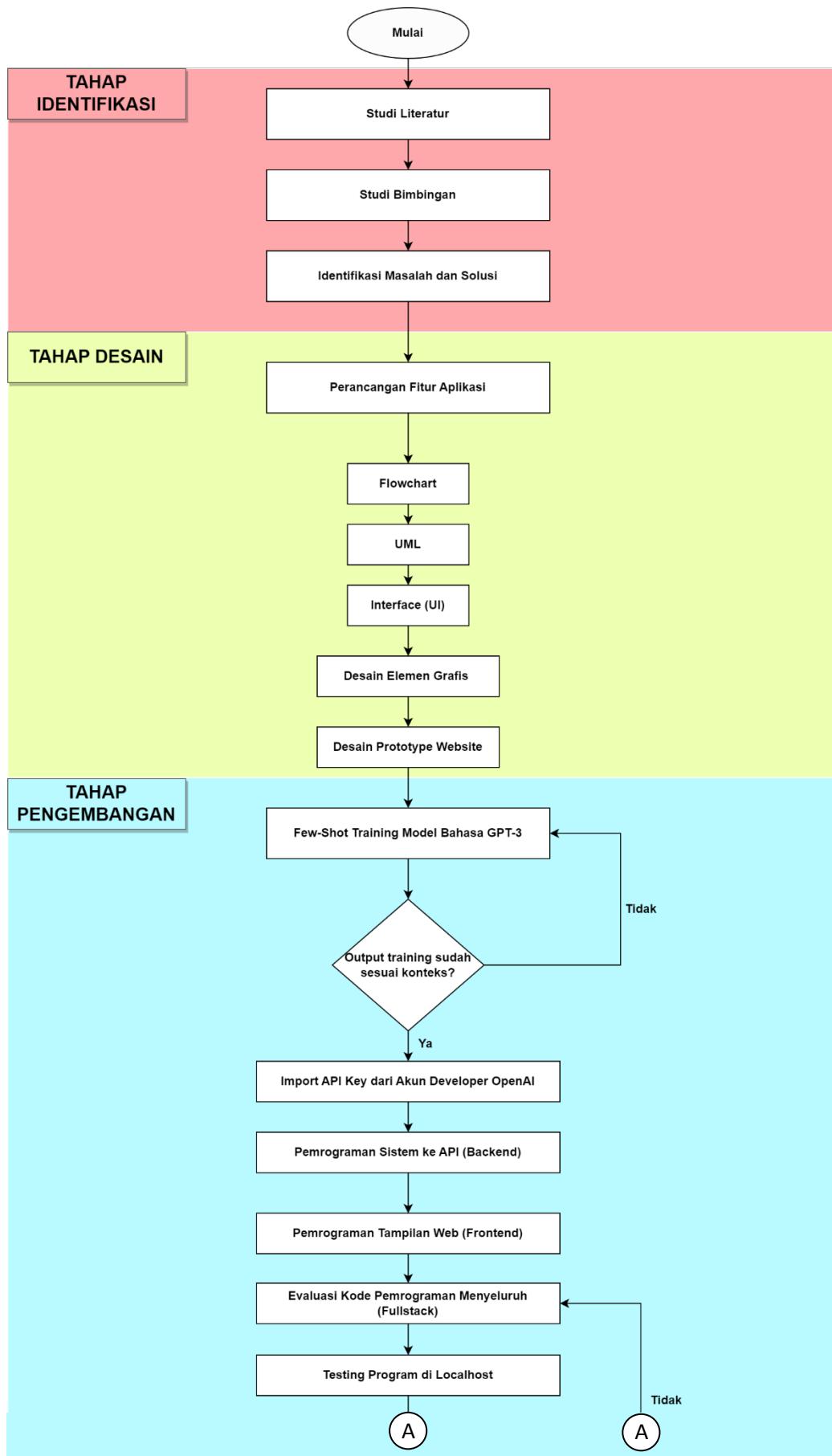
Setelah rancangan dan demonstrasi/simulasi didapatkan, pengujian terhadap aplikasi dilakukan dalam 3 jenis pengujian. Pengujian pertama dengan melakukan *evaluation testing* terhadap aplikasi menggunakan metode *blackbox*. Tahapan pengujian selanjutnya melalui matriks evaluasi, disini peneliti menggunakan dua matriks evaluasi yang paling sesuai dengan model bahasa yang digunakan, yaitu matriks *ROUGE* dan *BERTscore*. Pengujian yang ketiga yaitu menilai kepuasan pengguna dalam menggunakan aplikasi, yang dapat dilihat melalui *rating* penilaian aplikasi di *platform Google Play store*.

7. Laporan

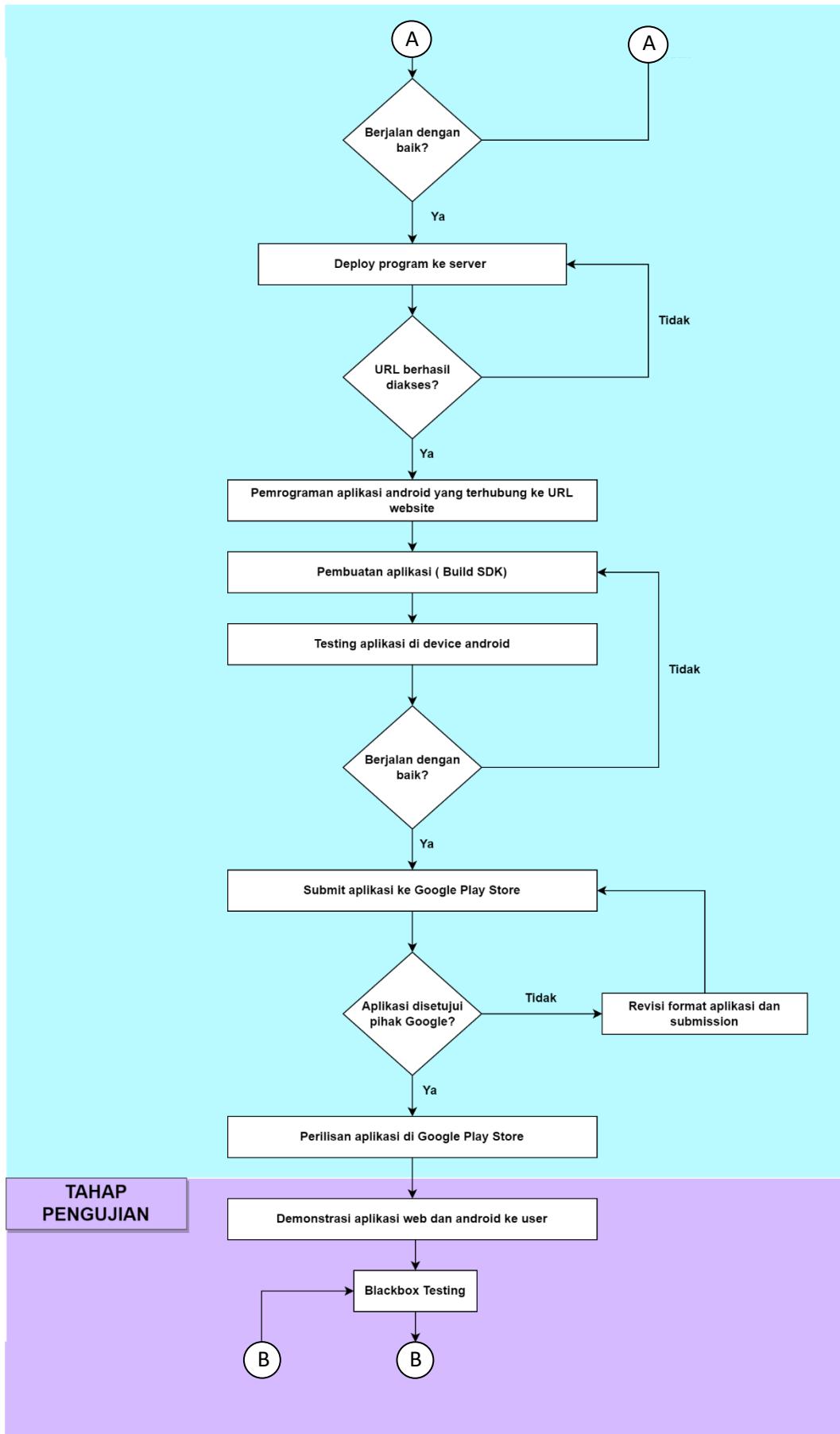
Laporan temuan penelitian berdasarkan data dan hasil analisis yang ada dibuat dan dilaporkan. Diharapkan hasil penelitian dapat memberikan kontribusi dalam mengatasi permasalahan yang ada.

3.4 Diagram Alir Penelitian

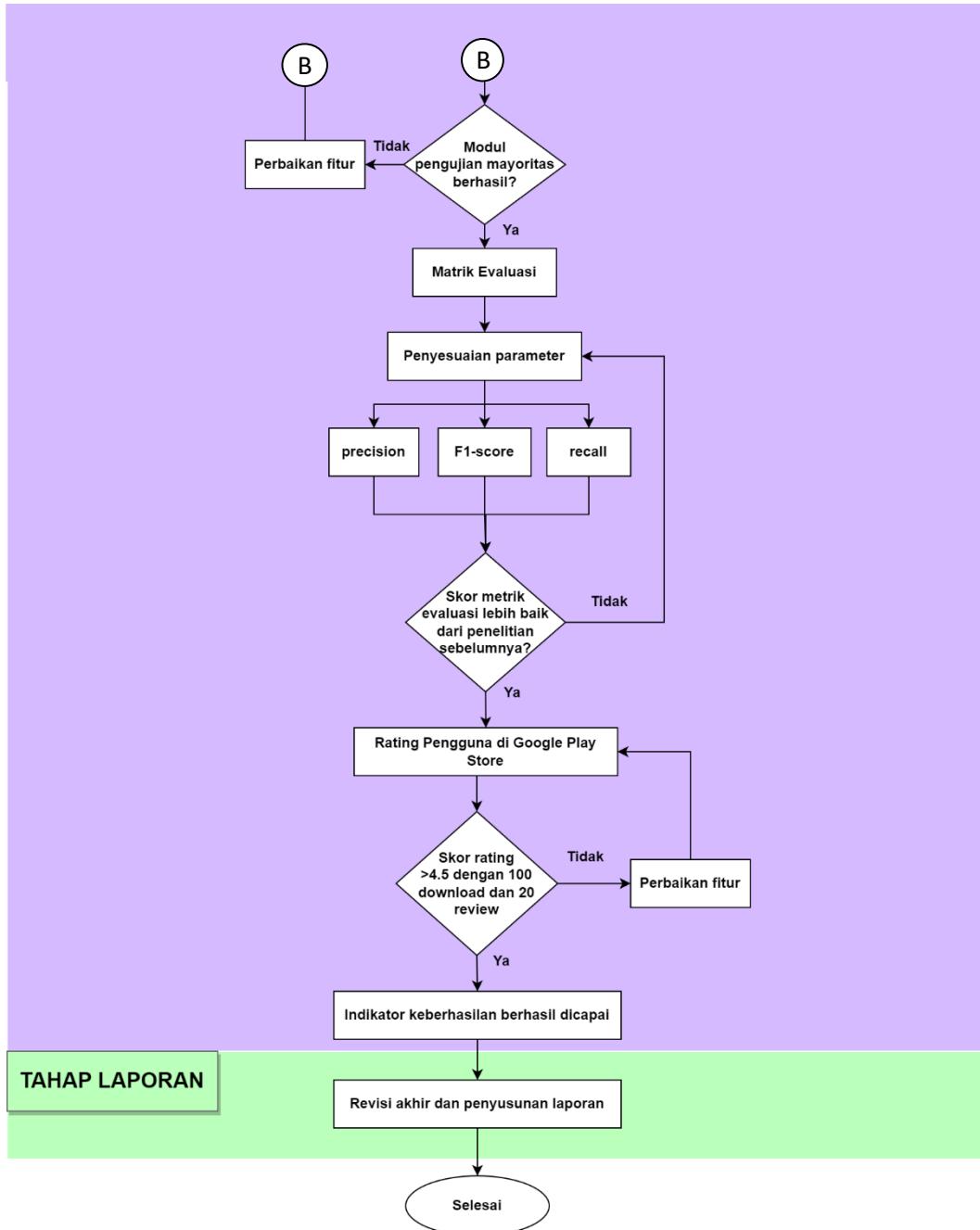
Prosedur penelitian digambarkan secara visual melalui diagram alir pada Gambar 3.4. Berikut merupakan tahapan penelitian yang dijelaskan melalui diagram alir (*flowchart*) penelitian sebagai berikut:



Gambar 3.4 Diagram Alir Penelitian



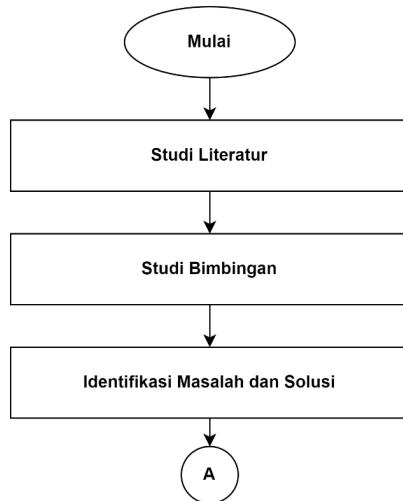
Gambar 3.4 Diagram Alir Penelitian (lanjutan)



Gambar 3.4 Diagram Alir Penelitian (lanjutan)

1. Tahap Identifikasi

Tahap ini memiliki peranan penting dalam perancangan suatu aplikasi agar terjamin nilai keefektifan perancangan aplikasinya, serta menentukan tujuan yang tepat sasaran. Pada tahap ini penulis melakukan beberapa pendekatan terhadap objek yang dikaji.

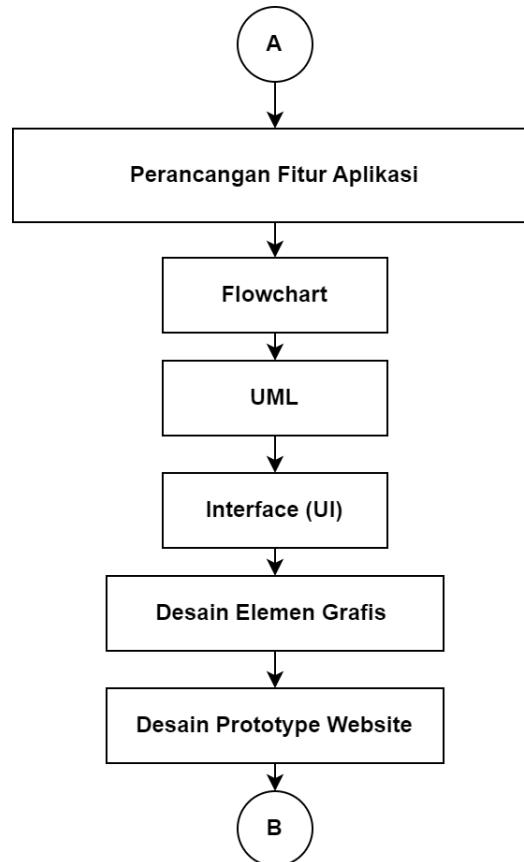


Gambar 3.5 Diagram alir tahap identifikasi

- a) Studi Literatur dilakukan dengan tujuan meningkatkan pemahaman pandangan para ahli dan para praktisi teknologi yang memiliki peranan aktif dalam mengembangkan teknologi. Kegiatan ini dilakukan dengan mempelajari dan memahami topik yang berkaitan dari berbagai literatur yang berasal dari beberapa macam referensi maupun sumber ilmiah yang resmi seperti penelitian terdahulu, jurnal, *e-book*, dan artikel resmi.
- b) Studi Bimbingan dilakukan melalui kegiatan diskusi dengan dosen pembimbing untuk mengulas materi dan permasalahan yang dapat ditemukan dari penelitian terkait. Kegiatan ini membantu penulis dalam mendapatkan wawasan tambahan dan pengetahuan yang lebih dalam setelah melakukan studi literatur.
- c) Identifikasi Masalah dan Solusi merupakan tahap mendapatkan gambaran permasalahan yang akan diangkat setelah melakukan pengamatan teori dan merancang langkah solutif yang digunakan untuk menyelesaikan permasalahan terkait penelitian yang dilakukan.

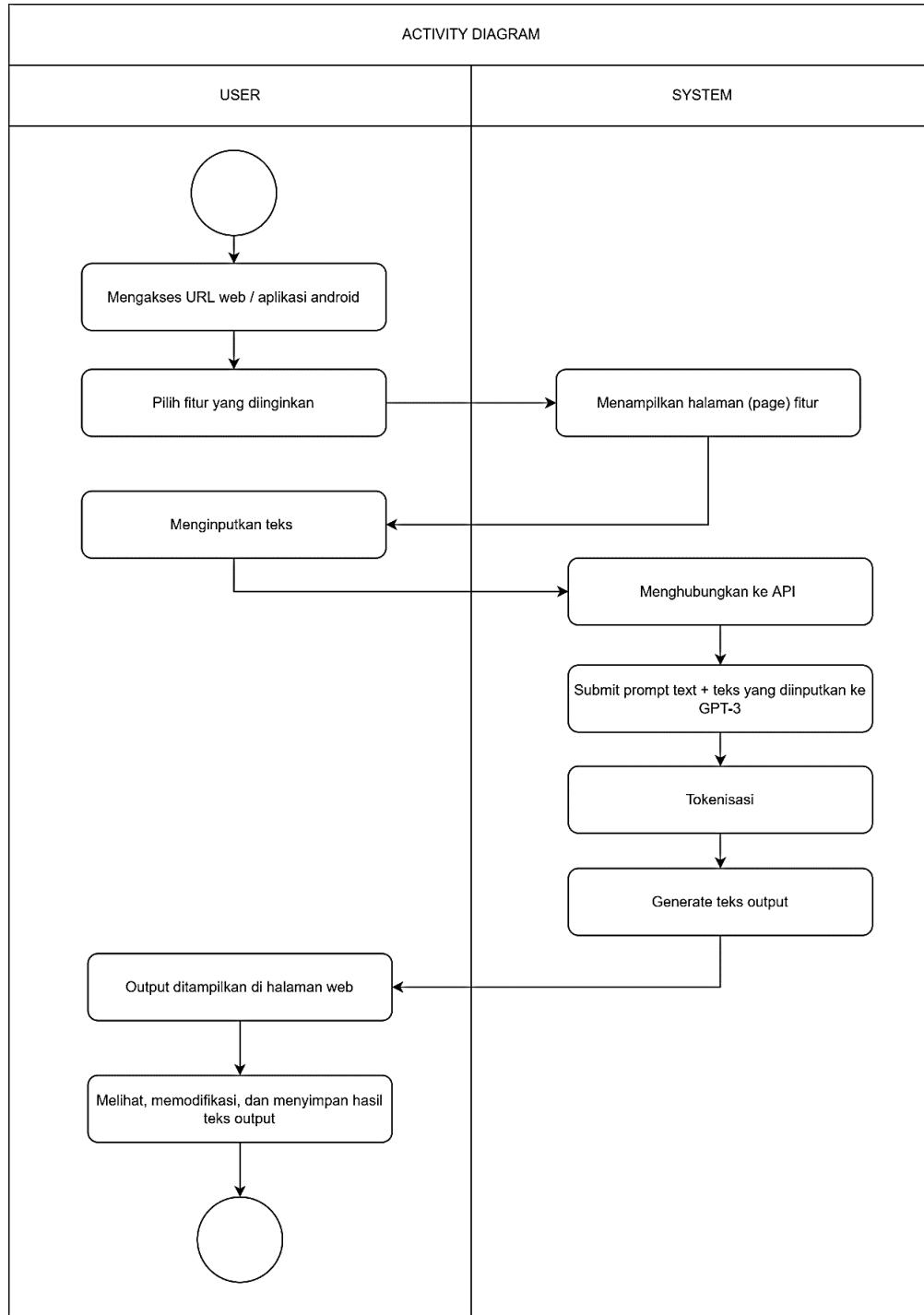
2. Tahap Desain

Tahap desain suatu aplikasi meliputi beberapa rancangan yang dibuat sebelum proses pemrograman, dan lebih berfokus pada tampilan aplikasi dan keterkaitan antar proses perancangan dan pemodelan aplikasi secara visual.



Gambar 3.6 Diagram Alir Tahap Desain Apikasi

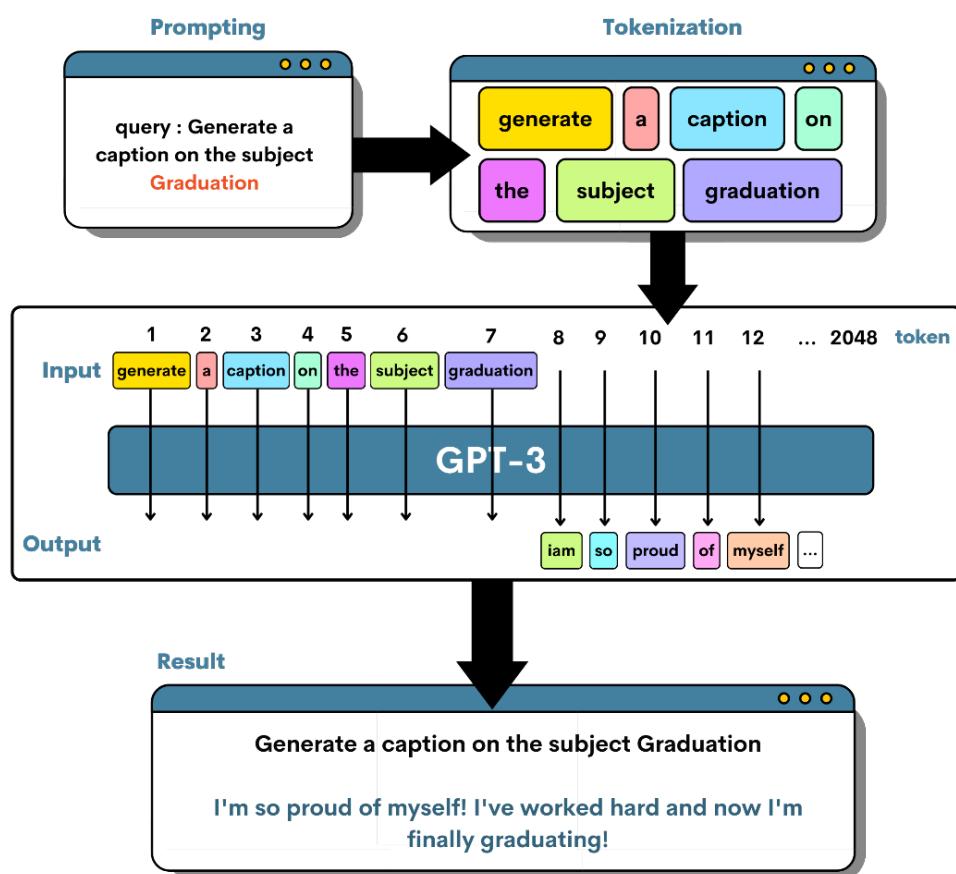
- a) Perancangan Fitur Aplikasi dilakukan dengan membuat rancangan cara kerja aplikasi, menentukan fitur yang ada pada aplikasi, membuat rancangan tampilan aplikasi, membuat pemodelan secara spesifik sebelum aplikasi diprogram.
- b) *Unified Modeling Language (UML)* diagram dibuat dengan tujuan untuk memodelkan secara visual sistem rancangan aplikasi yang dibuat, dengan menyatukan beberapa informasi dalam arsitektur *software*. Jenis diagram *UML* yang akan diterapkan di perancangan aplikasi ini yaitu *activity diagram*. Diagram *UML* pada Gambar 3.7 menjelaskan tentang pemodelan bagaimana aplikasi bekerja dari sisi *user* dan sistem.



Gambar 3.7 UML Activity Diagram

Diagram aktivitas dimulai dengan *user* mengakses *url* web atau membuka aplikasi, dilanjutkan dengan *user* memilih fitur yang akan digunakan. Saat *user* telah menekan fitur yang dipilih, akan dilanjutkan dengan respon sistem yaitu menampilkan halaman web sesuai fitur yang dipilih. Tampilan web tersebut

diisi dengan teks *input* oleh *user*, ketika *user* menekan *submit*, maka akan dilanjutkan oleh sistem untuk proses penghubungan ke *API GPT-3* untuk dilakukan proses *prompting* pada layanan *GPT-3*, kemudian dilanjutkan dengan proses tokenisasi yaitu pemrosesan kalimat dengan memisahkan setiap katanya untuk memahami konteks kalimat dan memberikan prediksi respon *output* atas teks yang diproses sebelumnya. Setelah teks hasil pemrosesan pada layanan *API*, maka teks tersebut diteruskan untuk ditampilkan ke halaman web yang dapat dilihat oleh *user*. Hasil teks yang ditampilkan pada halaman web dapat dimodifikasi oleh *user* secara manual sesuai dengan kebutuhan.



Gambar 3.8 Pemrosesan token kata dalam *GPT-3*

Konsep tokenisasi lebih detailnya dapat dilihat pada Gambar 3.8 yang menjelaskan lanjutan proses pada layanan *API* yang diakses, yaitu dengan membuat *prompting* (penyusunan *input*) berdasarkan kalimat

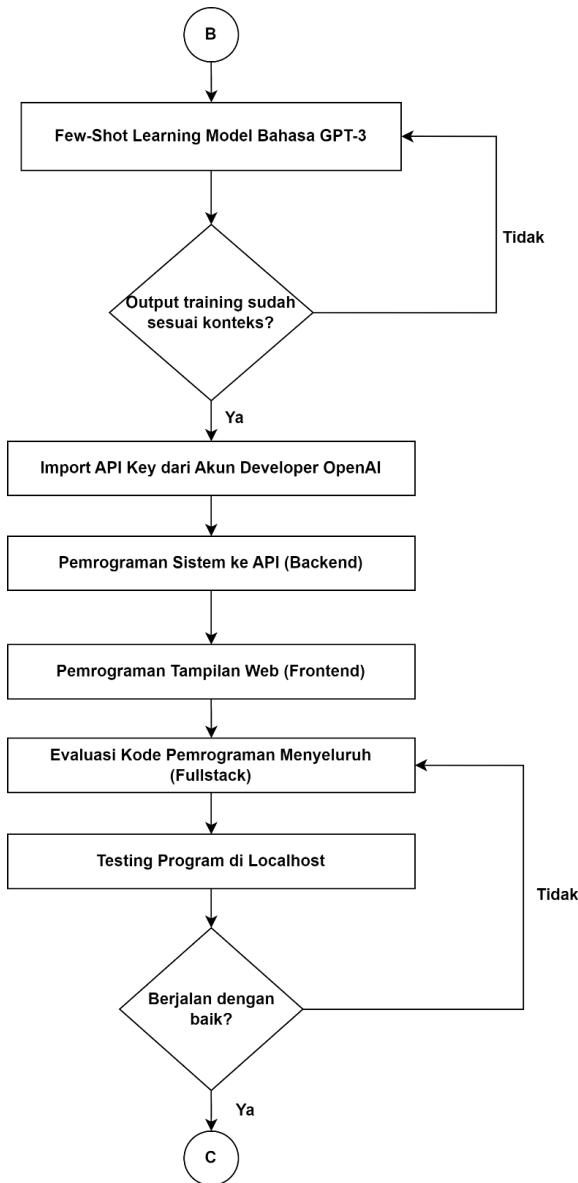
input yang diberikan oleh *user* sebelumnya melalui tampilan web, sehingga kalimat tersebut disatukan dengan format *prompting* yang ada dalam kode pemrograman untuk diproses oleh *GPT-3*. Hasil kalimat pada *prompting* dilanjutkan pada proses tokenisasi dengan memisahkan setiap kata dalam kalimat untuk diubah menjadi vektor agar dapat diproses oleh *GPT-3*. Setiap kata mewakili satu buah token, sehingga dalam proses kerjanya, *GPT-3* dapat memproses hingga 2048 token (termasuk token *input* dan *output*). Hasil dari pemrosesan token oleh *GPT-3* yaitu susunan kalimat yang sesuai dengan perintah *input* yang diberikan.

- c) *Flowchart* merupakan diagram alur bagan yang menampilkan tahapan rinci serta pembuatan keputusan dimulai dari awal hingga akhir perancangan aplikasi yang dibuat. *Flowchart* digunakan dalam penelitian ini sebagai acuan dalam menyelesaikan beberapa tahapan realisasi perancangan aplikasi secara urut sesuai dengan yang direncanakan.
- d) *User Interface* diperlukan dalam penelitian ini untuk merancang tampilan aplikasi dari sisi *user*. Tahapan ini menentukan elemen grafis dalam tampilan *web* sesuai dengan fungsinya, penulis menggunakan aplikasi *Adobe Illustrator* untuk membuat komponen grafis pembangun tampilan *web* untuk dilanjutkan ke tahapan desain *prototype website* menggunakan aplikasi *Figma*. Prototype *website* dibuat dengan tujuan agar desain yang direncanakan dapat disimulasikan letak komponen grafisnya dan juga perpindahan ke halaman lain setelah *button* di klik.

3. Tahap Pengembangan

Tahap pengembangan merupakan tahap dalam merealisasikan hasil desain suatu aplikasi secara logis kedalam kode program komputer. *Flowchart*, *UML*, dan *UI* yang telah dibuat sebelumnya akan diimplementasikan ke baris-baris kode dalam bahasa pemrograman dan akan dibangun ke bentuk *website* dan aplikasi. Tahapan pengembangan ini diharapkan dapat memastikan

apakah aplikasi yang dibuat memberikan hasil sesuai yang diinginkan atau tidak setelah melakukan pengetesan.



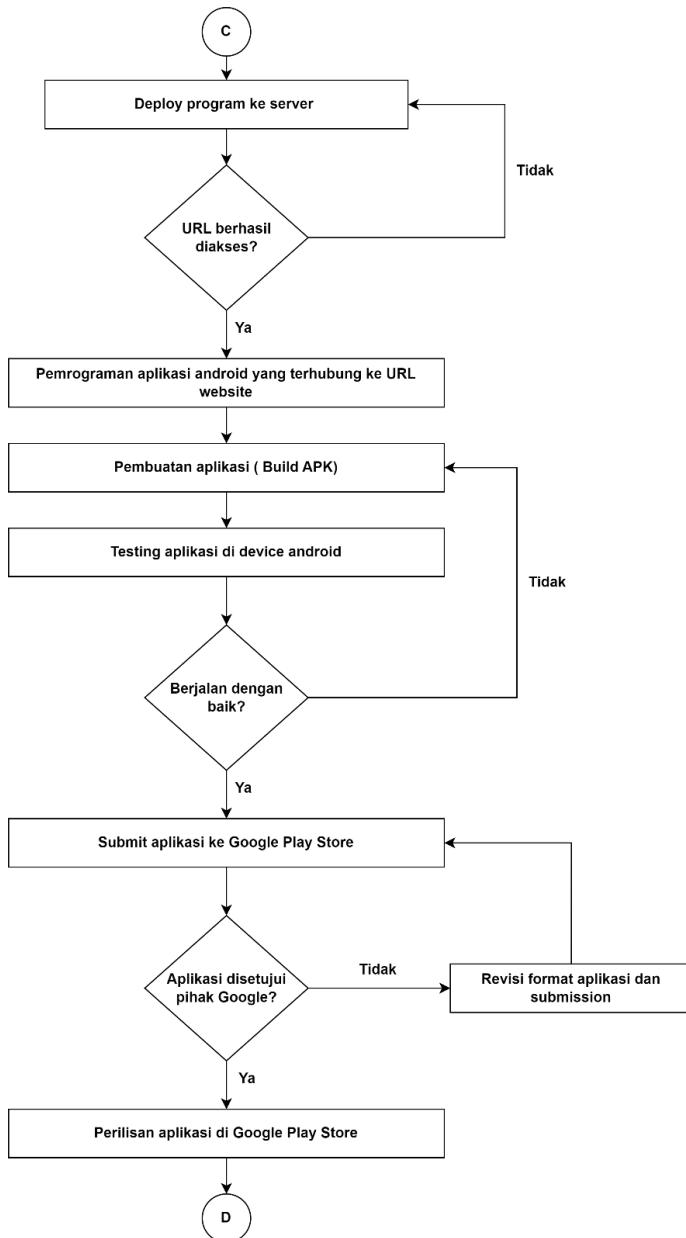
Gambar 3.9 Diagram Alir tahap Pengembangan (1)

- Training* model bahasa adalah proses awal dalam tahap pengembangan yang dilakukan dengan menerapkan *few-shot learning* pada model bahasa *GPT-3*. Tidak seperti *zero-shot learning* yang hanya melengkapi (*auto-complete*) *prompt* yang diberikan, teknik *few-shot* dipilih oleh penulis karena *output* yang diharapkan lebih sesuai konteks. Dalam *training* yang dilakukan, teknik *few-shot learning* diterapkan pada saat melakukan *setup* di *OpenAI GPT-3 Playground*, dengan memberikan

lebih dari satu *prompt* beserta hasil yang diharapkan, kemudian hasil *output*-nya digunakan sebagai *input* untuk *training* selanjutnya, hingga menghasilkan *output* yang lebih koheren dengan *prompt* yang diberikan. Ketika melakukan *training*, Mode yang digunakan adalah mode *Complete*. Model *GPT-3* yang diterapkan adalah *text-davinci-002* yang lebih unggul dalam menyelesaikan berbagai tugas dibanding dengan model yang lainnya. *Temperature* diatur ke nilai 0.5 untuk mengatur tingkat kerandoman kata yang dihasilkan, serta panjang maksimum tokennya adalah 256. Penerapan metode *few-shot learning* pada penelitian ini dilakukan dengan memberikan tiga sampel *input* dan *output* contoh yang dilanjutkan dengan pemberian *prompt* untuk melanjutkan *output* yang diproses oleh *GPT-3*. Pada metode *training* ini, setelah diberikan tiga *sample input output* pada setiap fiturnya, menghasilkan kalimat yang sudah sesuai konteks *input* yang diberikan.

- b) *Output training* akan terus diamati hingga mendapatkan hasil yang baik agar bisa dilanjutkan ke proses selanjutnya.
- c) *Import API Key* dilakukan melalui akun *developer* yang sudah didaftarkan oleh penulis di website *OpenAI*. Kode *API* yang didapatkan akan berbeda pada setiap akun *developer*.
- d) Pemrograman *Backend* meliputi kegiatan penulisan skrip kode untuk mengatur sistem aplikasi agar dapat berjalan dan melakukan *request API* ke layanan *GPT-3* serta menarik kembali *output* yang dihasilkan oleh *GPT-3* agar ditampilkan hasilnya melalui *command prompt*. Bahasa pemrograman yang digunakan untuk pembuatan sistem *backend*-nya yaitu *Python*, dengan mengimport *library OpenAI*.
- e) Pemrograman *Frontend* meliputi kegiatan penulisan kode program untuk tampilan *website*. Setelah *prototype* desain aplikasi dibuat di aplikasi *Figma*, maka setiap komponen grafisnya diunduh dan diprogram menjadi satu tampilan web *HTML* yang dapat dibuka di *browser*.
- f) Evaluasi Pemrograman (*fullstack*) menghubungkan kode pemrograman sistem dalam *backend* dan kode pemrograman tampilan web dan navigasinya dalam *frontend*.

- g) *Testing Program (localhost)* atau dilakukan uji coba program aplikasi yang sudah menjadi satu kesatuan dan dijalankan di *browser* setelah menjalankan kode *python* yang dibuat. Uji coba ini dilakukan di *localhost*, dan hanya bisa diakses secara lokal oleh komputer yang memiliki kode programnya. Uji coba yang dilakukan meliputi keberhasilan sistem dalam melakukan *request API* serta menampilkan *output* model bahasa *GPT-3*, susunan *layout* web, responsivitas tampilan *web*, navigasi ke halaman yang lainnya. Jika hasil *testing* belum baik, maka perlu perbaikan kode di tahap evaluasi pemrograman *fullstack*.

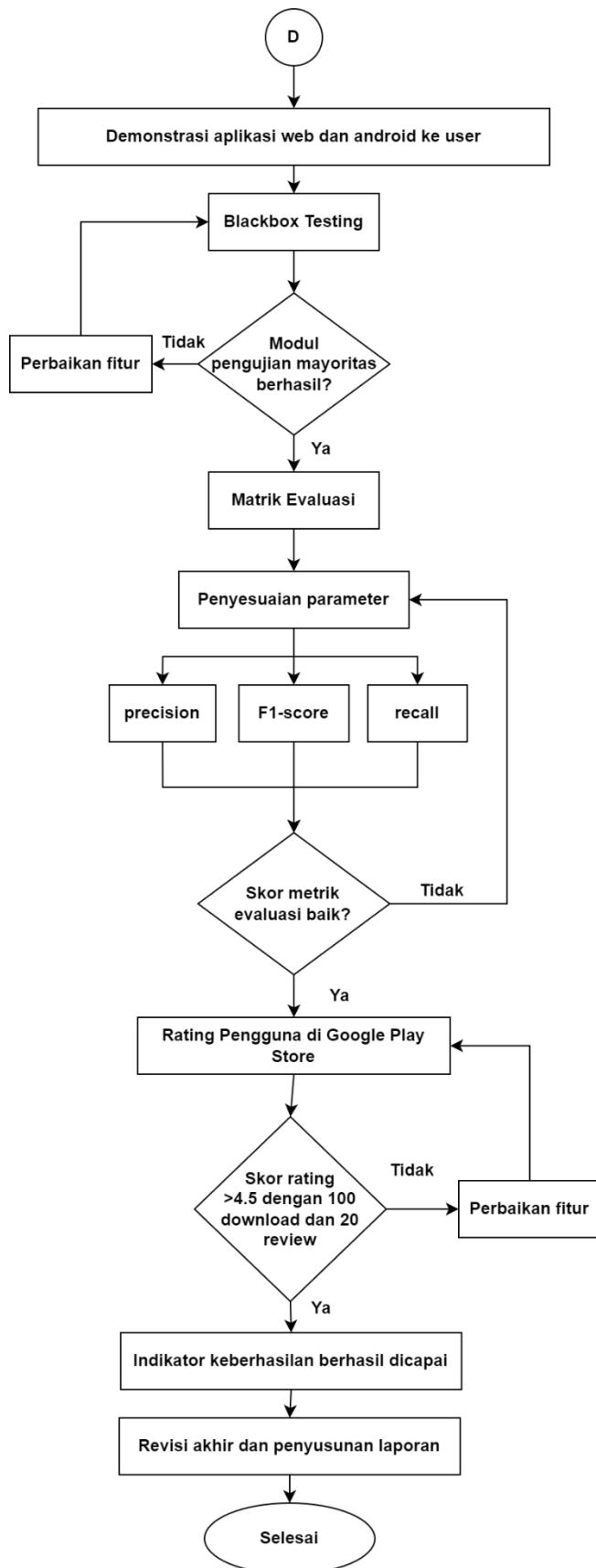


Gambar 3.10 Diagram Alir tahap Pengembangan (2)

- h) *Deploy Program* ke *Server* meliputi proses *deployment* kode program yang sebelumnya sudah berjalan dengan baik di *localhost*, diupload ke *server* agar aplikasi *web* memiliki *open URL* agar dapat diakses oleh semua perangkat. Penulis menggunakan layanan *server PythonAnywhere* untuk melakukan *deployment* aplikasi *web* yang sudah dibuat. Dalam proses *deployment*, jika *URL* yang dihasilkan masih belum dapat diakses, perlu *setting* ulang di *server*. Jika *URL* sudah bisa diakses oleh berbagai *device*, dapat dilanjutkan ke tahap berikutnya.
- i) Pemrograman Aplikasi Android dilakukan dengan menggunakan aplikasi *Android Studio*. Namun pemrograman aplikasi yang dilakukan tidak dari nol, karena hanya melakukan *request* ke *URL website* yang sudah dibuat agar dapat tampil dalam bentuk aplikasi android. Setelah aplikasi berhasil disimulasikan, maka dilakukan *build APK* agar aplikasi dapat dibangun dalam bentuk paket *file* yang bisa diinstal di perangkat android.
- j) *Testing* Aplikasi Android (*local*) dilakukan dengan meng-*install* paket aplikasi yang sudah dibuat ke perangkat android (*smartphone*) untuk menguji aplikasi berjalan dengan baik atau terdapat *error* baik di proses penginstalan atau menjalankan aplikasinya. Jika masih ditemukan *error* pada saat menjalankan aplikasi di perangkat *smartphone*, maka perlu diulangi langkah *build APK* untuk memperbaiki aplikasi di *Android Studio*.
- k) *Submit* Aplikasi ke *Google Play store* merupakan proses lanjutan jika aplikasi berjalan lancar. Proses *submit* aplikasi ke *Google Play Store* membutuhkan akun *developer* untuk mengunggah paket aplikasi, dan *fee service* untuk merilis aplikasi tersebut di *platform Google*. Jika aplikasi diterima oleh pihak *Google*, maka aplikasi dapat dirilis di *platform* tersebut, sehingga dapat diunduh oleh semua *user* yang membutuhkan.

4. Tahap Pengujian

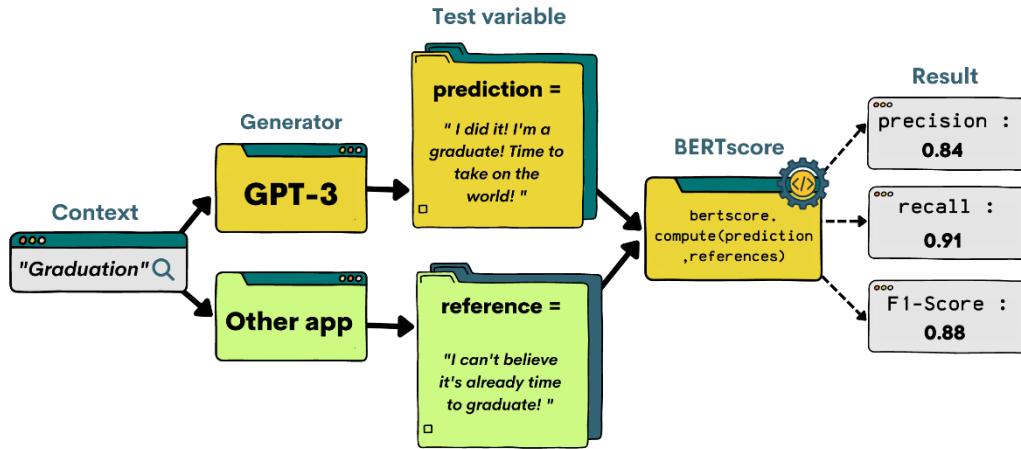
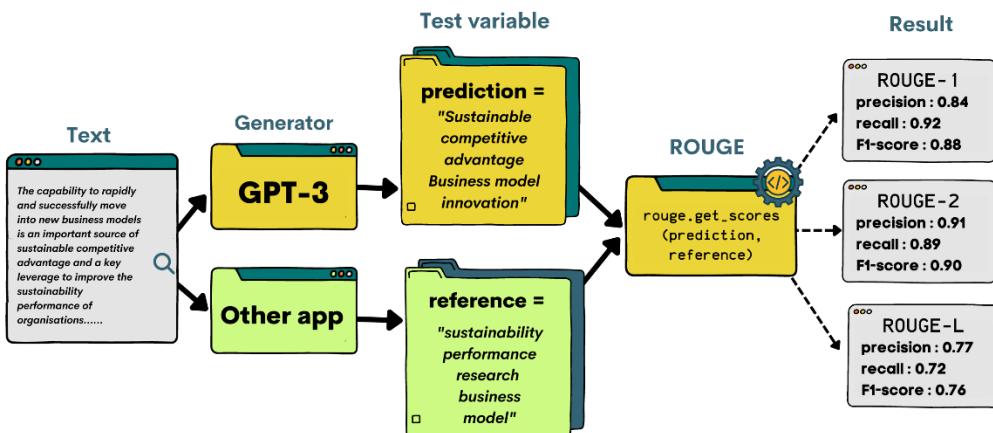
Tahap Pengujian meliputi beberapa proses untuk menguji keberhasilan aplikasi yang telah dibuat dan di uji coba pada berbagai perangkat.



Gambar 3.11 Diagram Alir tahap Pengujian dan Laporan

- a) Demonstrasi Aplikasi *Web* dan *Android* ke *User* dilakukan oleh peneliti dengan menyebarkan informasi berupa aplikasi yang telah tersedia di *platform Google Play store* dan *URL* untuk mengakses versi web.
- b) *Blackbox Testing* atau disebut *behaviorial testing* merupakan salah satu metode pengujian aplikasi yang dilakukan dengan cara pengamatan *input* dan hasil *output* aplikasi tersebut tanpa memperhatikan struktur kode dari aplikasi tersebut dan melaporkan apakah aplikasi dapat berfungsi dengan baik. Teknik *blackbox testing* yang diterapkan oleh peneliti yaitu *all-pair testing* atau *pairwise testing* dengan menguji semua probabilitas kombinasi dari seluruh hubungan antar bagian aplikasi berdasarkan *input* parameternya. Jika hasil pengujian parameter lebih banyak *error*, maka perlu diulangi langkah untuk perbaikan fitur.
- c) Matrik Evaluasi merupakan salah satu metode uji yang berfokus pada model bahasa *GPT-3* yang digunakan. Terdapat dua jenis matrik yang dipilih oleh peneliti untuk menetapkan standar keberhasilan aplikasi yang telah dibuat. Kedua matrik tersebut yaitu:
 1. *ROUGE* atau *Recall-Oriented Understudy for Gisting Evaluation*, matrik ini akan menghasilkan 3 nilai yaitu *ROUGE-1*, *ROUGE-2*, dan *ROUGE-L*.
 2. *BERTscore*, matrik ini akan menghasilkan 3 nilai yaitu *Precision*, *Recall*, dan *F1-score*.

Penghitungan hasil evaluasi dilakukan dengan penentuan dari 12 fitur aplikasi untuk dikumpulkan data pengujian pada masing-masing fiturnya. Setiap fitur dikategorikan untuk diuji dengan menggunakan *ROUGE* atau *BERTscore*. Jika akan diukur skor kemiripan “konteks” atau makna kalimat, maka digunakan *BERTscore*. Jika akan diukur skor “kesamaan” kata dalam kalimat maka digunakan *ROUGE*. Gambar 3.12 dan Gambar 3.13 menjelaskan proses pengujian dengan matrik evaluasi *BERTscore* dan *ROUGE* pada aplikasi yang dibuat.

Gambar 3.12 Diagram blok pengujian *BERTscore*Gambar 3.13 Diagram blok pengujian *ROUGE*

Setelah data untuk pengujian sudah ditentukan, maka diambil data referensi yang didapat melalui teks yang dihasilkan oleh aplikasi penulis *AI* yang lainnya. Sehingga data yang diuji ada dua, yaitu data “*predictions*” yang berasal dari hasil kalimat oleh aplikasi yang dibuat, dan data “*references*” yang berasal dari hasil kalimat oleh aplikasi yang sudah ada sebelumnya seperti *copy.ai*, *rytr.me*, dan *quillbot*. Jika data untuk pengujian sudah didapatkan, dilakukan pengujian sesuai dengan parameter yang sesuai pada setiap fiturnya. Proses pengujian dilakukan dengan memanggil modul atau *library python ROUGE* dan *BERTscore*.

pada kode *python* melalui *Jupyter Notebook*, dilanjutkan dengan memasukkan data teks uji kedalam variabel masing-masing yaitu “*references*” dan “*predictions*” dalam baris program *python*, hingga akhirnya dilakukan running program untuk menampilkan hasil pengujian kualitas kalimat yang dihasilkan oleh aplikasi yang sudah dibuat, apakah kualitasnya sudah hampir setara dengan aplikasi penulis premium lainnya.

- d) *Rating Google Play Store* merupakan salah satu parameter uji yang berkaitan langsung dengan kepuasan target *user* dalam menggunakan aplikasi AI Caption Generator. Dalam hal ini, jika *rating google playstore* diatas nilai 4.5 dari nilai maksimum 5 dan jumlah pengguna yang mengunduh aplikasi lebih dari 100 unduhan serta 20 ulasan pada halaman aplikasi, maka dalam parameter uji ini akan dikatakan berhasil

5. Tahap Laporan

Tahap Laporan merupakan tahap akhir yang dilakukan setelah semua indikator keberhasilan penelitian sudah tercapai. Pembuatan laporan disusun sesuai ketentuan yang dicantumkan dalam pedoman penulisan skripsi Universitas Lampung.

BAB V

KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil pembuatan aplikasi AI Caption Generator pada penelitian “Implementasi Model Bahasa *OpenAI GPT-3* untuk Aplikasi *Text Content Generator* Berbasis Web dan Aplikasi *Mobile*” dapat diambil kesimpulan:

Telah berhasil dibuat aplikasi “AI Caption Generator” yang dapat diakses melalui *browser* web dan perangkat android yang mampu melakukan berbagai tugas pembuatan teks dengan 12 fitur yang tersedia, dengan total *rating* sebesar 5.0 pada platform Google *Play store* dan hasil rata-rata pengujian matrik *BERTscore precision, recall, F1-score* masing-masing sebesar 86%, 88%, 87% dan metrik *ROUGE precision, recall, F1-score* masing-masing sebesar 55%, 60%, 57% dan skor ini lebih tinggi dari penelitian [58, 59, 60].

5.2 Saran

Berdasarkan penelitian yang telah dilakukan mengenai “Implementasi Model Bahasa *OpenAI GPT-3* untuk Aplikasi *Text Content Generator* Berbasis Web dan Aplikasi *Mobile*”, saran yang dapat dilakukan terhadap pengembangan penelitian selanjutnya adalah:

1. Penambahan fitur seiring berkembangnya kategori pembuatan teks.
2. Pengembangan *user interface* web dan aplikasi android yang lebih dinamis dengan fitur yang lebih kompleks.

DAFTAR PUSTAKA

- [1] A. Vaswani et al., “Attention is All you Need, Advances in neural information processing systems,” in *Proc. Advances in Neural Information Processing Systems* vol. 30, 2017.
- [2] Surianto, *Bangkit Dari Pandemi Covid-19, MenkopUKM Ajak Milenial Berwirausaha*, Dinas Koperasi UMKM Provinsi Kepulauan Bangka Belitung, Aug. 2021. [Online]. Available: <https://kukm.babelprov.go.id/content/bangkit-dari-pandemi-covid-19-menkopukm-ajak-milenial-berwirausaha> [Accessed 06 Aug. 2022].
- [3] Accenture, “Accenture Report : Artificial Intelligence Has Potential to Increase Corporate Profitability in 16 Industries by an Average of 38 Percent by 2035,” Accenture, 2017.
- [4] PwC, “*Bot.Me: A revolutionary partnership, How AI is pushing man and machine closer together,*” United Kingdom : PwC Publications, 2018. [E-book]. Available: PwC publication library, https://www.pwc.com/it/it/publications/assets/docs/PwC_botme-booklet.pdf [Accessed: 30 Aug.2022].
- [5] B. Marr, *Artificial Intelligence Can Now Write Amazing Content, What Does That Mean For Humans?*, Forbes, Mar. 2019. [Online]. Available: <https://www.forbes.com/sites/bernardmarr/2019/03/29/artificial-intelligence-can-now-write-amazing-content-what-does-that-mean-for-humans/?sh=7ecb8c3650ab> [Accessed: 10 Aug. 2022].
- [6] A. Radford, K. Narashiman, T. Salimans, and I. Sutskever, *Improving Language Understanding by Generative Pre-Training*, OpenAI, 2018, [Online]. Available: OpenAI Blog, <https://openai.com/blog/language-unsupervised/> [Accessed: 12 Aug. 2022].
- [7] Q. Wang, P. Liu, Z. Zhu, H. Yin, Q. Zhang, and L. Zhang, “A text abstraction summary model based on BERT word embedding and reinforcement learning,” *Applied Science*, vol. 9, no. 21, Nov. 2019. [Online]. Available: Applied Science Digital Library, doi: 10.3390/app9214701 [Accessed 15 Aug. 2022]
- [8] C. Jinyin, Y. Wu, C. Jia, H. Zheng, and H. Guohan, “Customizable Text

- Generation via Conditional Text Generative Adversarial Network,” *Neurocomputing*, vol. 416, Nov. 2020, pp. 125-135. [Online]. Available: Neurocomputing ScienceDirect, doi: 10.1016/j.neucom.2018.12.092 [Accessed 15 Aug. 2022].
- [9] V. Risne, “Text summarization using transfer learning: Extractive and abstractive summarization using BERT and GPT-2 on news and podcast data,” M.Comp.Sc. thesis, Computer Science and Engineering, University of Gothenburg, 2019.
 - [10] V. Kieuvongngam, B. Tan, and Y. Niu, “Automatic Text Summarization of COVID-19 Medical Research Articles using BERT and GPT-2,” *Computing Research Repository (CoRR)*, Jun 2020. [Online]. Available: arXiv, doi: 10.48550/arXiv.2006.01997 [Accessed: 16 Aug. 2022]
 - [11] F. Harrag, M. Debbah, K. Darwish, and A. Abdelali, “BERT Transformer model for Detecting Arabic GPT2 Auto-Generated Tweets,” in *Proc. 5th Arabic Natural Language Processing Workshop*, Barcelona: ACL, Jan. 2021, pp. 207–214.
 - [12] N. Akbar, I. Darmayanti, S. Fati, and A. Muneer, “Deep Learning of a Pre-trained Language Model’s Joke Classifier Using GPT-2,” *Journal of Hunan University Natural Sciences*, vol. 48, no. 8, Aug 2021, pp. 235-241. [Online]. Available: Hunan Daxue Xuebao, ISSN: 1674-2974 [Accessed: 17 Aug. 2022].
 - [13] B. Chintagunta, N. Katariya, X. Amatriain, and A. Kannan, “Medically Aware GPT-3 as a Data Generator for Medical Dialogue Summarization,” in *Proc. 2nd Natural Language Processing for Medical Conversations*, ACL, 2021, pp. 66–76, [Online]. doi: 10.18653/vl/2021.nlpmc-1.9.
 - [14] P. Mishra, C. Diwan, S. Srinivasa, and G. Srinivasaraghavan, “Automatic Title Generation for Text with Pre-trained Transformer Language Model,” in *IEEE 15th International Conference on Semantic Computing (ICSC)*, IEEE, Jan. 2021, pp. 17–24, [Online]. doi: 10.1109/ICSC50631.2021.00009.
 - [15] G. P. Prodan and E. Pelican, “Prompt scoring system for dialogue summarization using GPT-3,” *ACM Transaction on Audio, Speech, and Language Processing*, Sep. 2022, pp. 1–9. [Online]. Available: TechRxiv

- Digital Library, doi: 10.36227/techrxiv.16652392.v1 [Accessed: 17 Aug. 2022].
- [16] A. Olmo, S. Sreedharan, and S. Kambhampati, “GPT3-to-plan: Extracting plans from text using GPT-3,” in *International Conference on Automated Planning and Scheduling (ICAPS)*, United States: AAAI Press, 2021, [Online]. doi: <https://doi.org/10.48550/arXiv.2106.07131> [Accessed 18 Aug.2022].
 - [17] J. Shobana and M. Murali, “Abstractive Review Summarization based on Improved Attention Mechanism with Pointer Generator Network Model,” *Webology*, vol. 22, no. 1, Mar 2021, pp. 77–91. [Online]. Available: Webology Journal, doi: 10.14704/WEB/V18I1/WEB18028 [Accessed: 17 Aug. 2022].
 - [18] R. Sawai, I. Paik, and A. Kuwana, “Sentence augmentation for language translation using gpt-2,” *Electronics*, vol. 10, no. 24, Dec 2021. [Online]. Available: MDPI, doi: 10.3390/electronics10243082 [Accessed: 17 Aug. 2022].
 - [19] K.-L. Chiu, A. Collins, and R. Alexander, “Detecting Hate Speech with GPT-3,” *Computing Research Repository (CoRR)*, March 2021, pp. 1–29. [Online]. Available: arXiv, doi: 10.48550/arXiv.2103.12407
 - [20] Priya and S. Gupta, “Hate Speech Detection using OpenAI and GPT-3,” *International Journal of Emerging Technology and Advanced Engineering*, vol. 12, no. 5, May 2022, pp. 132–138. [Online]. Available: IJETAE, doi: 10.46338/ijetae0522_15 [Accessed: 18 Aug. 2022].
 - [21] M. E. G. Beheit and M. Ben Haj Hmida, “Automatic Arabic Poem Generation with GPT-2,” in *Proc. 14th International Conference on Agents and Artificial Intelligence, (ICAART 2022)*, vol. 2, Unites States: Curran Associates, pp. 366–374, [Online]. doi: 10.5220/0010847100003116.
 - [22] S. Balkus and D. Yan, “Improving Short Text Classification With Augmented Data Using GPT-3,” *arXiv*, May 2022, pp. 1–27 [Online]. Available: arXiv, doi: 10.48550/arXiv.2205.10981.
 - [23] M. kamal, S. Ali, A. Nasir, A. Samad, S. Basser, and A. Irshad, “An Automated Approach for the Prediction of the Severity Level of Bug Reports

- Using GPT-2,” *Security and Communication Networks*, May 2022, pp. 1-11 [Online]. Available: ACM Digital Library, doi: 10.1155/2022/2892401.
- [24] D. L. Yadin, *Creative Marketing Communications: A Practical Guide to Planning, Skills and Techniques*, 3rd ed. Great Britain: Kogan Page Publishers, 2001.
- [25] N. Malik and A. Solanki, “Simulation of Human Brain: Artificial Intelligence-Based Learning,” in *Impact of AI Technologies on Teaching, Learning, and Research in Higher Education*, 1st ed. Shivani Verma and Pradeep Tomar, India: IGIGLOBAL, 2020, pp. 150–160 [Online]. doi: 10.4018/978-1-7998-4763-2.ch009.
- [26] J. Alzubi, A. Nayyar, and A. Kumar, “Machine Learning from Theory to Algorithms: An Overview,” *Journal of Physics: Conference Series*, vol. 1142, no. 1, Dec 2018, pp. 1-15. [Online]. Available: IOP Science, doi: 10.1088/1742-6596/1142/1/012012.
- [27] Y. Lu, “Deep neural networks and fraud detection,” Uppsala University: Sweden, UUDM Report 2017:38, 2017.
- [28] R. Nuzzi, G. Boscia, P. Marolo, and F. Ricardi, “The Impact of Artificial Intelligence and Deep Learning in Eye Diseases: A Review,” *Frontiers in Medicine*, vol. 8, Aug 2021, pp. 1–11, 2021. [Online]. Available: Frontiers, doi: 10.3389/fmed.2021.710329.
- [29] S. Kostadinov, *Understanding Encoder-Decoder Sequence to Sequence Model*, Toward Data Science, Feb. 2019. [Online]. Available: <https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346> [accessed Aug. 12, 2021].
- [30] D. Bahdanau, K. H. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *3rd International Conference on Learning Representations (ICLR 2015)*. San Diego: DBLP, pp. 1–15, 2015.
- [31] N. S. Chauhan, *Attention mechanism in Deep Learning, Explained*, KDnuggets, Jan. 2021. [Online]. Available: <https://www.kdnuggets.com/2021/01/attention-mechanism-deep-learning-explained.html> [accessed Aug. 12, 2021].
- [32] Z. Lin et al., “A structured self-attentive sentence embedding,” in *5th*

- International Conference on Learning Representations (ICLR 2017).* France: DBLP, pp. 1–15, 2017.
- [33] A. Vaswani et al., “Attention is all you need,” in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, United States: ACM, pp. 5999–6009, 2017.
 - [34] L. Weng, *Attention? Attention!*, Github.io, Jun 2018. [Online]. Available: <https://lilianweng.github.io/posts/2018-06-24-attention/> [accessed Aug. 12, 2022].
 - [35] J. Kaplan et al., “Scaling Laws for Neural Language Models,” *OpenAI Publications*, Jan 2020, pp. 1-30. [Online]. Available: arXiv, doi: 10.48550/arXiv.2001.08361.
 - [36] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1. Minnesota: ACL, pp. 4171–4186, 2019.
 - [37] A. Zhang, *Large-Scale Pretraining with Transformers*, Dive Into Deep Learning, 2019. [Online]. Available: https://d2l.ai/chapter_attention-mechanisms-and-transformers/large-pretraining-transformers.html [accessed Aug. 1, 2022].
 - [38] T. B. Brown et al., “Language models are few-shot learners,” in *34th Conference on Neural Information Processing Systems (NeurIPS 2020) - Proceedings of the Conference*. Canada: Curran Associates, pp. 1877-1901, 2020.
 - [39] M. Sanad, *A Comprehensive Guide to Build your own Language Model in Python!*, Medium, Aug. 2019. [Online]. Available: <https://medium.com/Analytics-Vidhya/a-comprehensive-guide-to-build-your-own-language-model-in-python-5141b3917d6d> [accessed Sep. 20, 2022].
 - [40] T. Fatyanosa, *Language Modeling dengan Transformer*, Medium, Dec. 2020. [Online]. Available: <https://fatyanosa.medium.com/language-modeling-dengan-transformer-ae2d1bab647d> [accessed Sep. 15, 2022].
 - [41] J. Alammar, *The Illustrated GPT-2 (Visualizing Transformer Language*

- Models)*. Github.io, 2018. [Online]. Available: <http://jalamar.github.io/illustrated-gpt2/> [accessed Sep. 10, 2022].
- [42] P. J. Liu et al., “Generating wikipedia by summarizing long sequences,” in *6th International Conference on Learning Representations (ICLR 2018)*, Canada: OpenReview, pp. 1–18, 2018.
- [43] Lysandre, *Summary of the tokenizers*, HuggingFace, 2021. [Online]. Available: https://huggingface.co/docs/transformers/tokenizer_summary [accessed Aug. 13, 2022].
- [44] S. Zhao, R. Gupta, Y. Song, and D. Zhou, “Extremely small BERT models from mixed-vocabulary training,” in *6th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2021)*, Online: ACL, pp. 2753–2759, 2021. [Online]. doi: 10.18653/v1/2021.eacl-main.238.
- [45] M. P. Marcus, *Language Modelling on Penn Treebank (Perplexity)*, Papers With Code, 2020. [Online]. Available: <https://paperswithcode.com/sota/language-modelling-on-penn-treebank-word> [accessed Aug. 15, 2022].
- [46] M. P. Marcus, *Language Modelling on Penn Treebank (Parameters)*, Papers With Code, 2020. [Online]. Available: <https://paperswithcode.com/sota/language-modelling-on-penn-treebank-word?metric=Params> [accessed Aug. 15, 2022].
- [47] A. Kothalawala, *What is an API? How does it work?*, Medium, Jun 2018. [Online]. Available: <https://medium.com/@ama.thanu/what-is-an-api-how-does-it-work-f4ea552d741f> [accessed Sep. 10, 2022].
- [48] E. Bruno, *A Scrollable, Selectable HTML Table with JavaScript and CSS*, Sweetcode, 2020. [Online]. Available: <https://sweetcode.io/scrollable-selectable-html-table/> [accessed Sep. 05, 2022].
- [49] StatCounter, *Desktop vs Mobile Market Share Worldwide*, StatCounter, 2022. [Online]. Available: <https://gs.statcounter.com/platform-market-share/desktop-mobile/worldwide/#yearly-2011-2022> [accessed Sep. 05, 2022].
- [50] A. Celikyilmaz, E. Clark, and J. Gao, “Evaluation of Text Generation: A Survey,” *Computing Research Repository (CoRR)*, May 2021, pp. 1–75,

- 2020, [Online]. Available: arXiv, doi: 2006.14799.
- [51] C.-Y. Lin, “ROUGE: A Package for Automatic Evaluation of Summaries,” in *Proc. Workshop on Text Summarization Branches Out*, Spain: ACL, no. 12, pp. 74–81, 2004.
- [52] J. Briggs, *The Ultimate Performance Metric in NLP*, Medium, Mar 2021. [Online]. Available: <https://towardsdatascience.com/the-ultimate-performance-metric-in-nlp-111df6c64460> [accessed Sep. 09, 2022].
- [53] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating Text Generation with BERT,” in *8th International Conference On Learning Representations (ICLR 2020)*, Addis Ababa: OpeReview, pp. 1–43, 2019. [Online]. doi: 1904.09675.
- [54] I. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, “Language Models are Unsupervised Multitask Learners,” OpenAI: United States, Technical Report, 2019.
- [55] J. R. Simón, Y. Ledeneva, and R. A. García-Hernández, “Calculating the significance of automatic extractive text summarization using a genetic algorithm,” *Journal of Intelligent and Fuzzy Systems*, vol. 35, no. 1, July 2018, pp. 293–304. [Online]. Available: IOS Press, doi:10.3233/JIFS-169588.
- [56] W. Chen, K. Ramos, K. N. Mullaguri, and A. S. Wu, “Genetic Algorithms For Extractive Summarization,” *Computing Research Repository (CoRR)*, Jan 2022, pp. 1-4. [Online]. Available: arXiv, doi: abs/2105.02365.
- [57] T. Goyal, J. J. Li, and G. Durrett, “News Summarization and Evaluation in the Era of GPT-3,” *arXiv*, Sep 2022, pp. 1-19. [Online]. Available: arXiv, doi: 10.48550/arxiv.2209.12356