KLASIFIKASI ARTIKEL PUBLIKASI PENELITIAN DENGAN PENDEKATAN *LARGE LANGUAGE MODEL* PADA PENYEMATAN TEKS

SEMINAR BIDANG KAJIAN

BOLDSON HERDIANTO SITUMORANG
99223103

PROGRAM DOKTOR TEKNOLOGI
INFORMASI UNIVERSITAS GUNADARMA
JUNI 2024

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Bagi kalangan peneliti, jurnal ilmiah menjadi salah satu sumber referensi dalam membuat penelitian. Mesin pencari, perpustakaan digital, dan indeks kutipan digunakan secara luas untuk mencari publikasi penelitian. Saat pengguna mengajukan kueri, dari sejumlah besar dokumen, hanya sedikit dokumen yang relevan. Selain itu, adanya kemungkinan sebuah makalah penelitian dapat diklasifikasikan ke dalam satu atau lebih kategori, sehingga diperlukan pemetaan makalah penelitian dengan kategori terkait. Karena pengindeksan yang tidak memadai, sebagian dokumen yang dihasilkan tidak terstruktur. Dengan semakin meningkatnya jumlah penelitian yang diterbitkan dalam jurnal ilmiah di Indonesia, khususnya jurnal ilmiah terakreditasi Sinta, maka diperlukan sebuah sistem yang dapat secara otomatis mengklasifikasikan publikasi penelitian ke dalam kategori yang tepat. Pemetaan makalah penelitian dengan kategori yang spesifik dapat membantu para peneliti dalam hal: (1) membantu peneliti dalam menemukan bahan yang relevan dengan topiknya, (2) menemukan literatur yang sesuai untuk menjelaskan konsep latar belakang penelitian yang diusulkan, dan (3) untuk pertanyaan pengguan, mesin pencari dan perpustakaan digital mengembalikan dokumen yang sesuai.

Klasifikasi dokumen adalah suatu teknik *text mining* yang digunakan untuk membagi dokumen teks ke dalam kelompok-kelompok yang berbeda berdasarkan tema atau kategori yang relevan (Yelmen et al., 2023). Klasifikasi dokumen dapat digunakan untuk membantu mengatur dan mengelompokkan dokumen teks yang besar sehingga lebih mudah dicari dan dianalisis dengan menggunakan algoritma klasifikasi yang telah dilatih dengan menggunakan data teks yang telah diberi label secara manual untuk mengklasifikasikan dokumen baru ke dalam kelompok yang relevan (Hossain et al., 2021). Dengan menggabungkan Pemrosesan Bahasa Alami (NLP) dan pembelajaran mesin (*machine learning*) dapat menyusun dan menganalisis teks dalam jumlah besar dan tidak terstruktur dengan cepat, efisien, dan akurat (Galanis et al., 2021). Pada klasifikasi publikasi penelitian diterapkan metode *supervised learning* dengan memberikan label pada kumpulan data pelatihan untuk memprediksi kategori dokumen baru secara akurat (Roihan et al., 2020). Secara konseptual, metode *supervised learning* berupaya menemukan hubungan antara dokumen dan kategorinya dengan melihat data historis yang diberi label. Teknik pelabelan memungkinkan peneliti untuk membandingkan dan mengevaluasi kinerja berbagai metode klasifikasi dalam konteks pengelompokkan artikel ilmiah (Rivest et al., 2021). Pelabelan biasanya dimulai dengan

meminta manusia untuk membuat penilaian tentang sepotong data tanpa label tertentu. Model *machine learning* menggunakan label yang disediakan manusia untuk mempelajari pola yang mendasarinya dalam proses yang disebut "pelatihan model". Hasilnya adalah model terlatih yang dapat digunakan untuk memprediksi pada data baru (Zhdanovskaya et al., 2023).

Klasifikasi publikasi penelitian terbagi menjadi dua kategori pendekatan: (1) pendekatan berbasis konten (seperti kata dan frasa), dan (2) teknik berbasis metadata (judul, kata kunci, istilah kunci, penulis, dan kategori). Menerapkan metadata ke dokumen dapat membantu mengatur konten dengan lebih baik, memastikan keakuratan informasi, menyederhanakan pencarian dan pengambilan dokumen, dan menghemat waktu saat melakukan pencarian (Agrawal et al., 2023).

Metode klasifikasi K-Nearest Neighbor (KNN) digunakan dalam pengklasifikasian objek berdasarkan data training yang paling dekat dengan objek tersebut. Memiliki fungsi untuk mengklasifikasikan data berdasarkan data pembelajaran (training dataset), yang diambil dari k tetangga terdekatnya (nearest neighbors). Dengan k merupakan banyaknya tetangga terdekat. Proses metode KNN melakukan pencarian data uji untuk kelompok k objek yang paling dekat dengan objek pada data baru atau data uji. KNN merupakan algoritma *supervised learning*, artinya algoritma KNN memanfaatkan data yang ada sebelumnya dan sudah diketahui hasilnya (Uddin et al., 2022). KNN merupakan contoh basis pembelajaran yang menyimpan data training sehingga klasifikasi untuk data yang belum terklasifikasi dapat ditemukan dengan cara membandingkannya dengan data training.

Bahasa memainkan peran mendasar dalam memfasilitasi komunikasi dan ekspresi diri manusia, serta interaksinya dengan mesin. Kebutuhan akan model yang digeneralisasi berasal dari meningkatnya permintaan akan mesin untuk menangani tugas-tugas bahasa yang kompleks, termasuk penerjemahan, peringkasan, pengambilan informasi, interaksi percakapan, dan lain-lain (Chernyavskiy et al., 2021). Dalam beberapa tahun terakhir, *Large Language Model* (LLM) telah muncul sebagai alat yang ampuh untuk beragam tugas NLP. LLM dirancang untuk memahami dan menghasilkan teks manusia dengan tingkat kompleksitas yang tinggi sehingga dapat digunakan dalam berbagai tugas NLP seperti penerjemahan, analisis sentimen, dan generasi teks dengan lebih efisien dan akurat. LLM dapat belajar dan memahami pola bahasa yang kompleks dalam teks manusia, sehingga dapat memberikan interpretasi yang lebih halus terhadap preferensi pengguna dan menghasilkan rekomendasi yang lebih canggih. LLM dapat disesuaikan (*fine-tuned*) untuk tugas-tugas khusus sehingga dapat meningkatkan kinerja model dalam menangani tugas-tugas spesifik (Ji et al., 2024). Pada penelitian (Agrawal et al., 2023), teknik pemodelan bahasa digunakan untuk mempelajari representasi teks dari

setiap film dalam ruang *embedding* teks dengan konteks metadata konten. Jaringan saraf umpan maju (*feedforward neural network*) digunakan untuk memprediksi label *genre* menggunakan embedding teks yang dipelajari (*learned textual embeddings*) sebagai fitur input. Hasil penelitian menunjukkan embedding Genre Spectrum yang dihasilkan menggunakan LLM menunjukkan kinerja yang lebih baik daripada embedding teks yang dihasilkan menggunakan Doc2Vec dan BERT dalam semua kelompok popularitas film.

1.2. Batasan dan Tujuan

Karena keragaman fitur, pendekatan berbasis konten biasanya memberikan hasil yang lebih baik dibandingkan teknik berbasis metadata. Namun, salah satu kelemahan paling signifikan dari teknik berbasis konten adalah tidak tersedianya sebagian besar makalah secara publik. Metadata publikasi penelitian seperti judul, kata kunci, istilah kunci, penulis, dan kategori yang dapat diperoleh gratis secara *online*, menjadi pendekatan yang dipilih beberapa peneliti untuk mengkategorikan makalah penelitian. Fokus dari penelitian ini adalah klasifikasi publikasi penelitian dalam bidang Rekayasa Perangkat Lunak (*Software Engineering*). Model yang diusulkan dengan menggunakan teknik berbasis metadata, yaitu mengklasifikasikan publikasi penelitian berdasarkan judul, abstrak, dan kata kunci.

Penyematan kata (*word embeddings*) merupakan proses konversi kata yang berupa karakter *alphanumeric* ke dalam bentuk *vector*. Dengan *word embedding*, kata-kata yang memiliki *semantic meaning* yang sama, berada tidak jauh satu sama lain pada *space* tersebut. *Large Language Model* (LLM) dapat mengklasifikasikan teks dengan makna yang serupa. LLM merupakan sebuah algoritma pembelajaran mendalam (*deep learning*) yang dapat mengenali, meringkas, menerjemahkan, memprediksi, dan menghasilkan teks dan konten lain berdasarkan pengetahuan yang diperoleh dari kumpulan data besar (Ouyang et al., 2022). Penerapan LLM pada *word embeddings* dilakukan dalam penelitian ini. Selama pelatihan, model secara berulang menyesuaikan nilai parameter hingga model memprediksi dengan benar token berikutnya dari urutan token masukan sebelumnya. Setelah dilatih, LLM dapat dengan mudah diadaptasi untuk melakukan banyak tugas menggunakan kumpulan data diawasi yang relatif kecil, sebuah proses yang dikenal dengan *fine tuning* (Wan et al., 2024).

Kebanyakan peneliti saat ini menerapkan strategi dengan menanyakan nilai ambang kesamaan kepada para ahli di bidang yang relevan, lalu memastikannya pada dataset melalui *trial and error*, sehingga pemrosesannya memakan waktu. Kesenjangan penelitian yang ditemukan adalah sebagian besar penelitian mengandalkan ukuran statistik tradisional untuk mengukur kesamaan. Untuk representasi tekstual, mereka hanya mencatat data berdasarkan

frekuensi, bukan berdasarkan makna dan konteks frasa. Dalam penelitian ini, penerapan strategi dengan teknik klasifikasi multi-label digunakan untuk memberikan beberapa label pada dokumen berdasarkan beberapa nilai ambang kesamaan yang menjadi batas bawah untuk mengkategorikan publikasi penelitian. Rata-rata skor kemiripan suatu dokumen uji tiap kategori dibandingkan dengan nilai ambang batas kemiripan kategori tersebut. Kategori yang mempunyai skor lebih tinggi dari nilai ambang batas dipilih sebagai kategori akhir dokumen uji. Tujuan dari penelitian ini adalah untuk melihat seberapa besar model semantik dapat meningkatkan akurasi klasifikasi serta bagaimana nilai ambang batas ditetapkan pada kategorisasi multi-label.

Menurut Computing Curricula 2020 (CC2020), rumpun ilmu pengetahuan dan teknologi komputasi terbagi ke dalam 7 (tujuh) lapisan fokus keilmuan komputasi, meliputi: (1) Sistem/ Teknik Komputer, (2) Ilmu Komputer, (3) Cyber-Security, (4) Sistem Informasi, (5) Teknologi Informasi, (6) Rekayasa Perangkat Lunak, (7) Sains Data (Clear & Parrish, 2021). Pada akhir tahun 2022, IEEE Computer Society (IEEE-CS) merilis versi beta 4 dari panduan *Software Engineering Body of Knowledge* (SWEBOK) untuk membantu profesional rekayasa sistem dan rekayasa perangkat lunak, pelajar, dan peneliti untuk mengidentifikasi dan berbagi pemahaman bersama tentang "pengetahuan yang diterima secara umum" dalam rekayasa perangkat lunak. Bidang pengetahuan Rekayasa Perangkat Lunak berdasarkan SWEBOK 4.0 meliputi bidang pengetahuan *Software Requrements*, *Software Architecture*, *Software Design*, *Software Construction*, *Software Testing*, *Software Engineering Operations*, *Software Maintenance*, *Software Configuration Management*, *Software Engineering Management*, *Software Engineering Process*, *Software Engineering Models and Methods*, *Software Quality*, *Software Security*, *Software Engineering Professional Practice*, *Software Engineering Economics*, *Computing Foundations*, *Mathematical Foundations*, dan *Engineering Foundations* Penelitian ini berfokus pada klasifikasi publikasi penelitian bidang Rekayasa Perangkat Lunak pada jurnal terakreditasi berbahasa Indonesia.

Tujuan penelitian ini adalah melakukan klasifikasi publikasi penelitian bidang Rekayasa Perangkat Lunak (*Software Engineering*) pada jurnal terakreditasi berbahasa Indonesia dengan melakukan penyematan kata (*word embeddings*) menggunakan algoritma *Large Language Model* (LLM).

1.3. Kontribusi

1. Penelitian berfokus pada klasifikasi publikasi penelitian bidang Rekayasa Perangkat Lunak (*Software Engineering*) sehingga dapat membantu profesional rekayasa sistem dan rekayasa

perangkat lunak, pelajar, dan peneliti untuk mengidentifikasi dan berbagi pemahaman bersama tentang "pengetahuan yang diterima secara umum" dalam rekayasa perangkat lunak.

2. Klasifikasi publikasi penelitian dilakukan berbasis metadata berdasarkan judul, abstrak, dan kata kunci sehingga dapat memberikan kontribusi dalam membantu pelajar dan peneliti untuk memperoleh referensi publikasi penelitian dengan keakuratan informasi, menyederhanakan pencarian dan pengambilan dokumen, dan menghemat waktu saat melakukan pencarian.
3. Klasifikasi teks atau dokumen untuk mengkategorikan publikasi penelitian menggunakan metode klasifikasi k-Nearest Neighbor (kNN).
4. Mengklasifikasikan teks yang memiliki makna serupa (*semantic meaning*) dengan teknik *word embeddings* menggunakan algoritma *Large Language Model* (LLM) sehingga setelah dilatih, LLM dapat dengan mudah diadaptasi untuk melakukan banyak tugas menggunakan kumpulan data diawasi (*supervised dataset*).

BAB II
TINJAUAN PUSTAKA

2.1. Rekayasa Perangkat Lunak (*Software Engineering*)

Menurut Computing Curricula 2020 (CC2020), rumpun ilmu pengetahuan dan teknologi komputasi terbagi ke dalam 7 (tujuh) lapisan fokus keilmuan komputasi, meliputi: (1) Sistem/ Teknik Komputer, (2) Ilmu Komputer, (3) Cyber-Security, (4) Sistem Informasi, (5) Teknologi Informasi, (6) Rekayasa Perangkat Lunak, (7) Sains Data (Clear & Parrish, 2021).

ISO/IEC/IEEE Systems and Software Engineering Vocabulary (SEVOCAB) mendefinisikan rekayasa perangkat lunak sebagai "penerapan pendekatan yang sistematis, disiplin, dan terukur terhadap pengembangan, pengoperasian, dan pemeliharaan perangkat lunak; yaitu penerapan rekayasa pada perangkat lunak" (IEEE Computer Society, 2024). Bidang pengetahuan Rekayasa Perangkat Lunak berdasarkan SWEBOK 4.0 meliputi bidang (IEEE Computer Society, 2024):

1. *Software Requirements*: menjelaskan aktivitas yang terlibat dalam memperoleh, menganalisis, menentukan, memvalidasi, dan mengelola kebutuhan perangkat lunak.
2. *Software Architecture*: merupakan bidang pengetahuan baru yang dihasilkan dari kesadaran bahwa arsitektur adalah disiplin ilmu yang signifikan dan berbeda dibandingkan desain perangkat lunak. Arsitektur perangkat lunak terdiri dari struktur dasar elemen perangkat lunak, hubungan antar elemen, dan properti elemen dan relasi.
3. *Software Design*: berfokus pada proses mengubah kebutuhan menjadi representasi sistem perangkat lunak.
4. *Software Construction*: meliputi aktivitas yang terlibat dalam menerjemahkan desain perangkat lunak ke dalam kode yang dapat dieksekusi. Bidang ini telah diperbaharui untuk merefleksikan teknik dan praktik konstruksi modern: mengelola ketergantungan, pengembangan lintas platform dan migrasi, putaran umpan balik untuk konstruksi, pemrograman visual, dan platform *low-code*/*zero-code* (yaitu, disebut *no code*).
5. *Software Testing*: berhubungan dengan berbagai pendekatan dan teknik yang digunakan untuk menilai kebenaran perangkat lunak dan memverifikasi kesesuaiannya dengan kebutuhan. Bidang ini menyediakan konten baru tentang pengujian perangkat lunak dalam proses pengembangan terkini (seperti Agile, DevOps, dan Test-Driven Development), domain aplikasi (seperti otomotif, perawatan kesehatan, seluler, legal, dan IoT), teknologi baru (seperti AI, blockchain, cloud) dan atribut kualitas (seperti keamanan dan privasi).

6. *Software Engineering Operations*: merupakan bidang pengetahuan baru yang membahas evolusi peran software engineer untuk memasukkan aktivitas DevOps dan infrastruktur sebagai Layanan (IaaS) sambil menghilangkan silo organisasi antara pengembangan, pemeliharaan, dan operasi.
7. *Software Maintenance (SM)*: membahas dasar-dasar, proses, dan teknik untuk memberikan dukungan hemat biaya untuk pengoperasian perangkat lunak.
8. *Software Configuration Management (SCM)*: berfokus pada empat fungsi dasar SCM: (1) identifikasi konfigurasi: mengidentifikasi semua item konfigurasi dari status sistem perangkat lunak, (2) kontrol perubahan: mengoordinasikan dan membatasi akses ke konfigurasi dalam sistem perangkat lunak, (3) audit konfigurasi: serangkaian tinjauan yang mengonfirmasi bahwa perubahan sedang dilakukan agar sesuai dengan kondisi sistem perangkat lunak yang diinginkan, (4) status accounting: secara otomatis mencatat informasi tentang identifikasi konfigurasi, hubungan, baseline, dan perubahan pada setiap identifikasi konfigurasi yang dikembangkan, serta alasan pembuatannya, kapan dibuat, dan siapa yang membuatnya.
9. *Software Engineering Management*: menjelaskan aktivitas perencanaan, pengorganisasian, estimasi, dan pelacakan proyek perangkat lunak.
10. *Software Engineering Process*: menjelaskan aktivitas, metode, dan teknik yang digunakan untuk mendefinisikan, mengimplementasikan, menilai, dan meningkatkan proses pengembangan perangkat lunak.
11. *Software Engineering Models and Methods*: memperkenalkan berbagai model, metode, dan alat yang mendukung proses pengembangan perangkat lunak.
12. *Software Quality*: mengatasi aktivitas dan teknik yang terlibat dalam mencapai dan mengevaluasi kualitas perangkat lunak. Kini mencakup topik-topik baru: (1) tingkat ketergantungan dan integritas perangkat lunak, (2) standar, model, dan sertifikasi, (3) kebijakan, proses, dan prosedur, dan (4) pengendalian mutu dan pengujian.
13. *Software Security*: merupakan bidang pengetahuan baru yang berfokus pada topik keamanan yang lebih luas, khususnya dasar-dasar keamanan, manajemen keamanan, alat keamanan, dan keamanan khusus domain, serta aktivitas rekayasa keamanan utama (seperti pengujian keamanan).
14. *Software Engineering Professional Practice*: meliputi pertimbangan etis, kode etik profesional, dan peran rekayasa perangkat lunak dalam masyarakat, meliputi: (1) praktik profesional mengikuti praktik, standar, dan pedoman yang diterima secara umum yang ditetapkan oleh masyarakat profesional yang berlaku, (2) desain inklusif antarmuka

pengguna/ pengalaman pengguna (UI/UX), (3) pertimbangan tentang keberagaman dan inklusi, dan (4) pertimbangan tentang kelincahan.

15. *Software Engineering Economics*: berfokus pada esensi rekayasa ekonomi, ilmu pengambilan keputusan, dan memperluas lebih tradisional, pandangan keuangan murni tentang rekayasa ekonomi.
16. *Computing Foundations*: meliputi konsep dan teori dasar yang mendasari disiplin rekayasa perangkat lunak.
17. *Mathematical Foundations*: memperkenalkan teknik dan prinsip matematika yang relevan dengan rekayasa perangkat lunak, seperti logika, probabilitas, dan matematika diskrit.
18. *Engineering Foundations*: meliputi prinsip dan praktik teknik yang berlaku untuk rekayasa perangkat lunak, termasuk rekayasa sistem dan manajemen proyek.

2.2. Penyematan Kata (*Word Embedding*)

Word embedding merupakan proses konversi kata yang berupa karakter *alphanumeric* ke dalam vektor. Dengan *word embedding*, kata-kata yang memiliki *semantic meaning* yang sama, menjadi lebih dekat dan memiliki representasi yang serupa. Pilihan untuk memilih teknik *word embedding* terbaik sangat penting sebagai langkah pra-pemrosesan dalam tugas NLP seperti klasifikasi teks (Roman et al., 2021).

Penelitian (Jeangirard, 2021) menggunakan pendekatan gabungan antara klasifikasi berbasis jurnal dan klasifikasi berbasis artikel dalam konteks publikasi biomedis dengan menggunakan metadata dari Pubmed. Teknik embedding dalam NLP digunakan untuk melatih pengklasifikasi Fasttext dengan metadata artikel (judul, abstrak, kata kunci, dan Medical Subject Headings/ MeSH) yang diberi label dengan klasifikasi tingkat jurnal FoR (Fields of Research). Fasttext menggunakan teknik word embeddings untuk merepresentasikan kata-kata dalam bentuk vektor numerik yang memuat semantik tentang kata-kata tersebut. Model Fasttext dibangun dan dilatih menggunakan data teks yang telah diproses tokenisasi. Nilai F1 score dari model-model yang dilatih berbasis judul jurnal 99,7%, judul 40,8%, abstrak 44,6%, kata kunci 43,6%, MeSH 42,8%.

(Mustafa et al., 2021) menggunakan pendekatan Word2Vec untuk menghasilkan representasi vektor kata-kata berdasarkan teks dari artikel penelitian yang berasal dari Journal of Universal Computer Sience (J.UCS) sebanyak 1.460 publikasi penelitian, dan dari Association of Computing Machinery (ACM) sebanyak 86.116 publikasi penelitian. Eksperimen menghasilkan akurasi rata-rata 0,86 dan 0,84 untuk JUCS dan ACM dengan *single*

label classification, serta 0,81 dan 0,80 untuk JUCS dan ACM dengan *multi lable classification*. Dibandingkan dengan Bag of Words, TF, dan TF-IDF, model Word2Vec mempunyai kinerja yang baik dalam pemrosesan bahasa alami (NLP).

Teknik *word embedding* menggunakan Global Vectors (GloVe), Infersent, dan BiDirectional Encoder Representation from Transformers (BERT) digunakan (Roman et al., 2021) pada klasifikasi maksud kutipan (*citation intent*) untuk memudahkan peneliti menavigasi dan menemukan penelitian saat menjelajahi kutipan (*citation*). Hasil penelitian menunjukkan teknik *word embedding* BERT berkinerja jauh lebih baik, dengan skor presisi 89%.

Penggunaan metode TF-ID (*Term Frequency-Inverse Document*) untuk mengekstrak kata-kata fitur yang penting dari abstrak artikel jurnal wisata dilakukan pada penelitian (Chang et al., 2022). TF-IDF digunakan untuk menyoroti kata-kata yang paling penting dan mewakili topik-topik yang dibahas dalam abstrak artikel jurnal pariwisata. Kata-kata umum yang muncul di banyak dokumen akan memiliki bobot IDF rendah, sehingga kata-kata ini tidak akan memberikan kontribusi besar dalam menentukan topik dari suatu dokumen. Dengan mengalikan nilai TF (frekuensi kata dalam dokumen) dengan nilai IDF (kebalikan frekuensi kata di seluruh dokumen), TF-IDF memberikan bobot yang lebih akurat pada kata-kata yang penting dan spesifik untuk setiap dokumen.

Berdasarkan beberapa penelitian di atas, hasil penerapan beberapa teknik *word embedding* menunjukkan kekuatan dan kelemahan yang hampir serupa, yang dirangkum pada Tabel 1.

Tabel 1. Teknik *Word Embedding*

No	Algoritma	Keunggulan	Kelemahan
1	Fasttext (Jeangirard, 2021)	a. Efisien dalam penggunaan sumber daya komputasi, sehingga cocok untuk melatih model pada dataset besar seperti metadata publikasi ilmiah. b. Penelitian dapat dilakukan dengan waktu yang lebih singkat dikarenakan cepat dalam melakukan pelatihan dan inferensi. c. Menghasilkan representasi vektor kata yang kaya dan bermakna.	a. Memiliki keterbatasan dalam memahami konteks yang sangat kompleks dalam metadata publikasi ilmiah, terutama jika terdapat hubungan yang rumit antar kata. b. Karena sulitnya mengevaluasi relevansi klasifikasi secara otomatis, penelitian (Jeangirard, 2021) memerlukan pengecekan manual pada sampel representatif publikasi untuk mengukur validitas pendekatan yang digunakan. c. Fasttext bergantung pada informasi yang terdapat dalam metadata publikasi

			seperti judul, abstrak, kata kunci, MeSH, dan judul jurnal. Jika metadata tidak lengkap atau tidak representatif, klasifikasi dapat terpengaruh.
2	Word2Vec (Mustafa et al., 2021)	a. Word2Vec mampu menghasilkan representasi vektor kata-kata yang menangkap informasi semantik dan kontekstual dari teks dokumen karena model dapat memahami hubungan antara kata-kata dalam dokumen dengan lebih baik. b. Dengan menggunakan representasi vektor kata-kata dari Word2Vec, klasifikasi dokumen dapat ditingkatkan karena informasi semantik yang terkandung dalam representasi vektor tersebut. c. Word2Vec dapat mengatasi masalah dimensi dalam representasi teks dengan mengubah kata-kata menjadi vektor numerik dengan dimensi yang lebih rendah.	a. Untuk melatih model Word2Vec yang baik, seringkali diperlukan dataset teks yang besar. Proses pelatihan model Word2Vec dapat memakan waktu dan sumber daya komputasi yang signifikan. b. Word2Vec cenderung memiliki keterbatasan dalam menangani kata-kata yang jarang muncul dalam dataset. Kata-kata yang jarang muncul mungkin tidak memiliki representasi vektor yang baik. c. Word2Vec memperhitungkan konteks lokal dari kata-kata dalam teks, namun memiliki keterbatasan dalam menangkap konteks global atau hubungan antar kalimat.
3	GloVe, Infersent, BERT (Roman et al., 2021)	a. BERT telah terbukti memberikan hasil yang lebih baik dalam analisis maksud kutipan (*citation intent*) dibandingkan GloVe dan Infersent. b. Infersent, GloVe, dan BERT dapat mengelompokkan kalimat kutipan dengan representasi internal yang mirip.	a. BERT memerlukan sumber daya komputasi yang lebih besar untuk pelatihan dan penggunaan dibandingkan dengan teknik *word embedding* lainnya. b. Meskipun GloVe dan Inversent dapat memberikan representasi kata yang baik untuk kata-kata yang sering muncul bersama dalam korpus teks, GloVe dan Infersent tidak dapat menangkap hubungan kontekstual antara kata-kata dalam teks.
4	TF-IDF (Chang et al., 2022)	a. TF-IDF membantu dalam mengidentifikasi kata-kata kunci yang paling penting dalam setiap dokumen, sehingga memudahkan dalam pemahaman topik yang dibahas.	a. TF-IDF dapat menjadi sensitif terhadap panjang dokumen, di mana dokumen yang lebih panjang cenderung memiliki bobot yang lebih tinggi untuk kata-kata yang sering muncul. b. TF-IDF tidak memperhitungkan urutan

		b. TF-IDF memberikan bobot yang lebih akurat pada kata-kata yang spesifik dan penting. c. TF-IDF membantu dalam pemilihan fitur yang relevan untuk analisis klasterisasi dan klasifikasi topik, sehingga meningkatkan interpretabilitas hasil.	kata dalam dokumen, sehingga informasi tentang konteks atau struktur kalimat tidak dipertimbangkan. c. Meskipun TF-IDF efektif dalam menyoroti kata-kata penting, namun tidak sepenuhnya memahami makna sebenarnya dari kata-kata tersebut.

Dalam beberapa tahun terakhir, *Large Language Model* (LLM) telah muncul sebagai alat yang ampuh untuk beragam tugas NLP. LLM dirancang untuk memahami dan menghasilkan teks manusia dengan tingkat kompleksitas yang tinggi sehingga dapat digunakan dalam berbagai tugas NLP (Ji et al., 2024). Penerapan LLM pada *word embeddings* akan dilakukan dalam penelitian ini.

2.3. *Large Language Model* (LLM)

LLM adalah model AI tingkat lanjut yang dirancang untuk memahami, menghasilkan, dan berinteraksi dengan bahasa manusia. LLM dan GenAI (*Generative AI*) adalah bagian dari pembelajaran mesin yang berfokus pada pembuatan konten yang meniru kemampuan manusia. LLM disebut "besar" karena dilatih dengan data teks dalam jumlah besar dan berisi miliaran atau bahkan triliunan parameter (Ji et al., 2024).

LLM biasanya dibangun menggunakan arstektur transformator. Transformator adalah jenis jaringan saraf yang cocok untuk tugas pemrosesan bahasa alami (NLP). Transformator mampu mempelajari ketergantungan jangka panjang antar kata, yang penting untuk memahami nuansa bahasa manusia (Fields, J., Chovanec & Madiraju, 2024). LLM memanfaatkan data pelatihan yang sangat besar, arsitektur *deep neural network* dan transformator untuk memahami dan menghasilkan teks mirip manusia, menjadikannya alat yang ampuh untuk berbagai aplikasi dalam pemrosesan bahasa alami (NLP), seperti membuat kesimpulan untuk menghasilkan teks, menjawab pertanyaan, peringkasan, menerjemahkan bahasa, pembuatan kode (*code generation*), dan chatbots. Beberapa tahun terakhir telah terjadi peningkatan fenomenal dalam kinerja dan aplikasi jaringan saraf transformator. Ragam jaringan transformator, termasuk *Bidirectional Encoder Representations from Transformer* (BERT), *Generative Pre-trained Transformer* (GPT) dan *Vision Transformer* (ViT), telah menunjukkan keefektifannya di seluruh domain *Natural Language Processing* (NLP) dan *Computer Vision* (Chitty-Venkata et al., 2023). Bidang NLP telah mengalami kemajuan yang signifikan karena

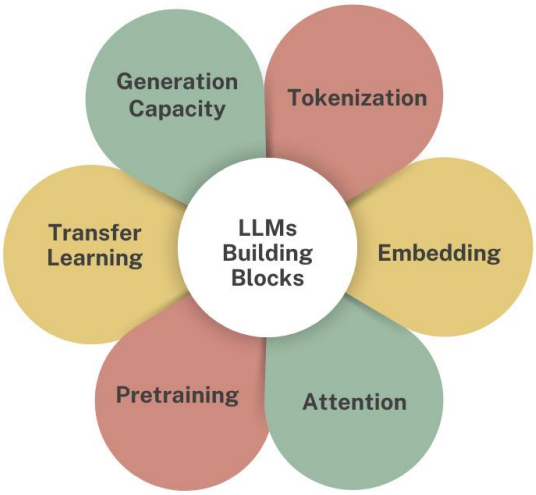
munculnya *Pre-trained Language Models* dalam skala besar, yang mencakup BERT dan GPT. Model ini telah meningkatkan efisiensi tugas NLP dan mengaktifkan aplikasi baru, termasuk ChatGPT, BART dan pembuatan konten (Paaß & Giesselbach, 2023).

Faktor kunci dalam cara kerja LLM adalah cara mereka merepresentasikan kata-kata. Bentuk *machine learning* sebelumnya menggunakan tabel numerik untuk merepresentasikan setiap kata. Namun, bentuk representasi ini tidak dapat mengenali hubungan antara kata-kata, seperti kata-kata dengan makna yang serupa. Keterbatasan ini diatasi dengan menggunakan vektor multidimensi, yang biasa disebut penyematan kata (*word embedding*), untuk merepresentasikan kata-kata sehingga kata-kata dengan makna kontekstual yang sama atau hubungan lainnya saling berdekatan dalam ruang vektor (Chernyavskiy et al., 2021).

LLM dilatih menggunakan teknik yang disebut pembelajaran tanpa pengawasan (*unsupervised learning*). Pelatihan ini melibatkan pemberian contoh teks kepada model dan mengajarkannya untuk memprediksi kata berikutnya secara berurutan berdasarkan kata-kata sebelumnya. LLM sangat besar sehingga tidak dapat dijalankan di satu komputer. Mereka biasanya dilatih pada kelompok komputer bahkan pada platform *cloud computing* (Usman Hadi et al., 2023). LLM terdiri dari beberapa komponen utama yang bekerjasama untuk memungkinkan LLM memahami, menghasilkan bahasa manusia dengan kelancaran dan akurasi yang luar biasa.

2.3.1. Komponen LLM

Terdapat enam komponen pada LLM yang ditunjukkan pada Gambar 1 (Naveed et al., 2023):



Gambar 1. Komponen-Komponen pada LLM

1. ***Tokenization*** **(Tokenisasi)**

   Tokenisasi adalah langkah pra-pemrosesan yang penting pada pelatihan LLM yang mengurai teks menjadi unit-unit yang tidak terurai disebut token. Token dapat berupa karakter, subkata, simbol, atau kata-kata, bergantung pada proses tokenisasi. Beberapa skema tokenisasi yang umum digunakan di LLM termasuk *wordpiece*, *byte pair coding* (BPE), dan *unigramLM*.

2. ***Embedding*** **(Penyematan)**

   Penyematan merupakan bagian integral dari operasi skala besar LLM. Penyematan adalah representasi vektor berkelanjutan dari token yang menangkap informasi semantik. Karena ukuran model LLM ini yang sangat besar, penyematan dipelajari melalui pelatihan ekstensif. Vektor berdimensi tinggi mengkodekan hubungan rumit antar token, sehingga memungkinkan model memahami nuansa kontekstual yang halus.

3. ***Attention*** **(Perhatian)**

   Mekanisme perhatian, terutama perhatian diri, memainkan peran penting dalam kemampuan LLM untuk menangani ukurannya yang besar. Mekanisme perhatian diri menganalisis hubungan antara semua token secara berurutan, memfasilitasi penangkapan ketergantung jangka panjang. Dalam model LLM, mekanisme perhatian ini sangat dapat diparalelkan, memungkinkan pemrosesan rangkaian ekstensif secara efisien.

5. ***Pretraining*** **(Pra-Pelatihan)**

   LLM yang berukuran besar dimanfaatkan melalui pra-pelatihan pada kumpulan data yang sangat besar. Selama pra-pelatihan, model mempelajari pola linguistik umum, worl knowledge, dan pemahaman kontekstual. Model terlatih ini menjadi repositori keahlian bahasa, yang kemudian dapat disesuaikan untuk tugas tertentu menggunakan kumpulan data yang lebih kecil.

6. ***Transfer Learning*** **(Pembelajaran Transfer)**

   LLM terlatih berukuran besar memfasilitasi kemampuan pembelajaran transfet yang luar biasa. Menyempurnakan model yang telah menyerap sejumlah besar pengetahuan linguistik memungkinkannya unggul dalam berbagai tugas. Pendekatan pembelajaran transfer ini memanfaatkan model terlatih dalam skala besar untuk beradaptasi dengan tugas baru tanpa perlu berlatih ulang dari awal.

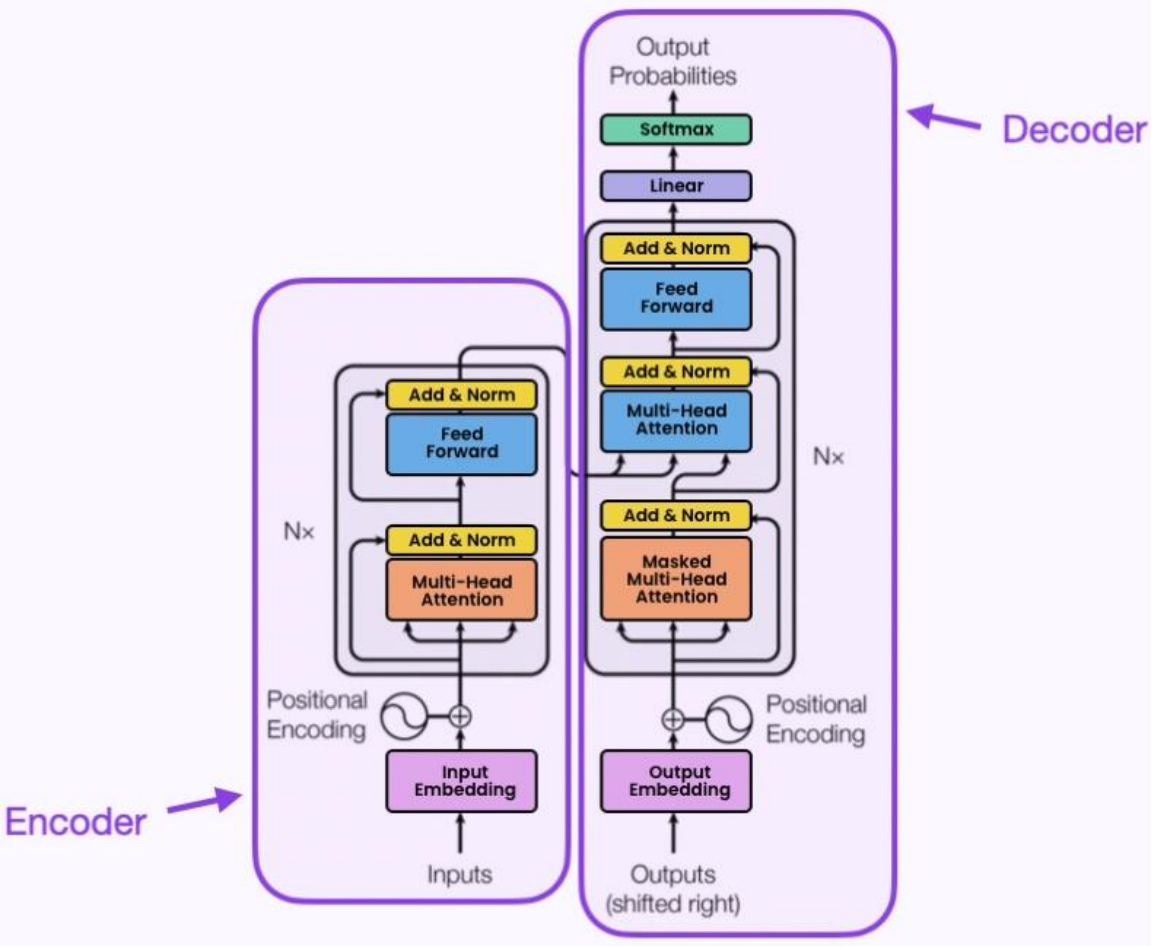
7. ***Generation Capacity*** **(Kapasitas Pembangkitan)**

   Luasnya LLM dalam hal parameter dan pengetahuan yang dipelajari memberdayakan LLM

dengan kapasitas pembuatan teks yang sangat besar. LLM dapat menghasilkan teks yang koheren dan relevan secara kontekstual di berbagai domain. Pemaparan ekstensif selama pelatihan memungkinkan LLM meniru penggunaan bahasa seperti manusia, menjadikannya alat serbaguna untuk tugas-tugas seperti pembuatan teks, terjemahan, ringkasan, dan banyak lagi.

2.3.2. Arsitektur LLM

Arsitektur LLM berakar pada kerangka Transformer, yang dikembangkan pada tahun 2017 oleh para peneliti di Google. Kerangka kerja ini secara mendasar telah mengubah lanskap pemrosesan dan pemahaman bahasa alami. Transformator terdiri dari dua komponen utama: *encoder* dan *decoder*. Model canggih ini beroperasi dengan memecah data masukan menjadi token, yang kemudian dilakukan operasi matematika simultan yang bertujuan untuk mengumngkap hubungan rumit antara token tersebut. Proses ini memberdayakan sistem untuk mengekstrak dan mengenali pola dengan cara yang analog dengan pemahaman manusia ketika dihadapkan pada penyelidikan serupa (Chitty-Venkata et al., 2023).



Gambar 2. Model Arsitektur Transformator (Chitty-Venkata et al., 2023)

1. ***Input Embedding*** **(Penyematan Masukan)**: kata-kata diubah menjadi vektor berdimensi tinggi yang disebut *embedding* (penyematan), yang diumpankan ke mekanisme perhatian.
2. ***Positional Encoding*** **(Pengkodean Posisi)**: karena transformator tidak memiliki operasi pengulangan atau konvolusi, maka transformator memerlukan mekanisme untuk mengingat informasi posisi relatif dari kata-kata dalam urutan masukan. Informasi posisi diinduksi menggunakan fungsi sin dan kosinus pada posisi genap dan ganjil, masing-masing, dalam urutan masukan (*input*).
3. ***Self-Attention***: Arsitektur transformator mengandalkan mekanisme self-attention, yang menunjukkan paralelisme model yang lebih baik dibandingkan dengan lapisan berulang dan memerlukan bias induktif minimal dibandingkan dengan jaringan konvolusi dari urutan masukan secara dinamis, menetapkan korelasi berpasangan dan memodelkan ketergantungan jangka panjang antara elemen urutan data masukan. Dalam self-attention, model menghitung bobot perhatian untuk setiap posisi dalam urutan, yang mencerminkan pentingnya setiap posisi dengan yang lain. Hal ini memungkinkan model untuk menangani bagian-bagian berbeda dari urutan tergantung pada masukannya. Masukan dari modul perhatian diumpankan ke tiga lapisan yang terhubung sepenuhnya (*fully-connected*/ FC), yang dipelajari selama pelatihan, untuk menghasilkan tensor *Query* (Q), *Key* (K), dan *Value* (V). *attention dot-product* berskala (A), yang diberikan dalam Persamaan 1, mewakili pengaruh setiap kata dalam *Query* terhadap kata lain dalam matriks *Key*.

$$A = softmax\left(\frac{QK^T}{\sqrt{D_k}}\right) \tag{1}$$

Query dan *Key* dikalikan elemen demi elemen untuk menghasilkan matriks skor, yang dibagi dengan $\sqrt{\boldsymbol{D_k}}$, akar kuadrat dari dimensi keluaran matriks *Key* untuk mengatasi hilangnya gradien. Fungsi *softmax* meningkatkan skor tinggi nilai dan meredam nilai skor yang lebih rendah. Skor attention akhirnya diperoleh dengan mengalikan matriks *attention* dan *value*, seperti yang diberikan pada Persamaan 2.

$$Attention\,(Q, K, V) \;=\; AV \tag{2}$$

4. ***Multi-Head Self-Attention*** **(MHA)**: modul MHA mencakup beberapa "head", yang masing-masing secara bersamaan menghitung operasi *attention*. Masukan ke modul MHA direplikasi di semua head. Masukan (X) ke head diproses melalui tiga lapisan FC (Wqi, Wki, Wvi) untuk mendapatkan satu set vektor Query ($\boldsymbol{Q_i}$), Key ($\boldsymbol{K_i}$), dan Value ($\boldsymbol{V_i}$) pada setiap head, seperti pada Persamaan 3.

$$Q_i = XW^{Q_i}, K_i = XW^{K_i}, V_i = XW^{V_i} \quad (3)$$

Output ($\boldsymbol{Z_i}$) dari setiap head dihitung menggunakan vektor $\boldsymbol{Q_i}$, $\boldsymbol{K_i}$, dan $\boldsymbol{V_i}$ melalui mekanisme self-attention, sesuai Persamaan 4.

$$head_i = Self - attention(Q_i, K_i, V_i), i = 1, 2, \ldots, h \quad (4)$$

Keluaran (output) independen dari semua head $\{\boldsymbol{head_1}, \ldots, \boldsymbol{head_h}\}$ digabungkan secara mendalam dan ditransformasikan secara linier menggunakan lapisan FC, sesuai Persamaan 5 untuk menghasilkan keluaran modul MHA.

$$\text{MHA(Q, K, V)} = [head_1, \ldots, head_h] * W^O \quad (5)$$

5. ***Pointwise Feed Forward Network*** **(FFN)**: unit FFN atau multi-layer perceptron (MLP) adalah serangkaian dua lapisan yang terhubung penuh (FC) dengan fungsi aktivasi ReLU atau GELU. FFN mempelajari informasi posisi-spesifik sehubungan dengan rangkaian urutan masukan yang berbeda. Keluaran MHA diumpankan ke FFN searah, yang selanjutnya diproses menggunakan operasi normalisasi (Norm).

2.3.2.1. *Encoder*

Modul *encoder* dan *decoder* dibuat dengan menumpuk lapisan yang identik, masing-masing dengan dua sub-lapisan: MHA dan FFN. Masukan ke modul MHA pertama adalah penyematan posisi dan masukan ke modul berikutnya adalah keluaran dari lapisan sebelumnya. Tensor keluaran MHA selanjutnya melalui lapisan Normalisasi dan diumpankan ke blok FFN untuk meningkatkan ekspresi urutan masukan menggunakan koneksi sisa dan dinormalisasi untuk menghasilkan *output encoder*.

2.3.2.2. *Decoder*

Decoder mengikuti struktur yang mirip dengan encoder dan dibangun menggunakan tumpukan tiga sub-lapisan yang identik. Sub-lapisan pertama adalah unit *masked* MHA. Pengoperasiannya setara dengan MHA, kecuali posisi akan datang di urutannya ditutupi karena belum diprediksi oleh jaringan. Sub-lapisan kedua adalah unit *multi-headed cross-attention*, dengan keluaran encoder dicampur dengan keluaran sub-lapisan pertama (*masked* MHA). Skema cross-attention ini menggunakan urutan yang dihasilkan sebelumnya encoder dan berfokus pada informasi penting dalam urutan tersebut. Sub-lapisan ketiga adalah FFN, yang mempelajari informasi posisi-spesifik dari urutan yang diproses, diikuti oleh lapisan FC.

2.3.3. Keluarga Arsitektur Transformator

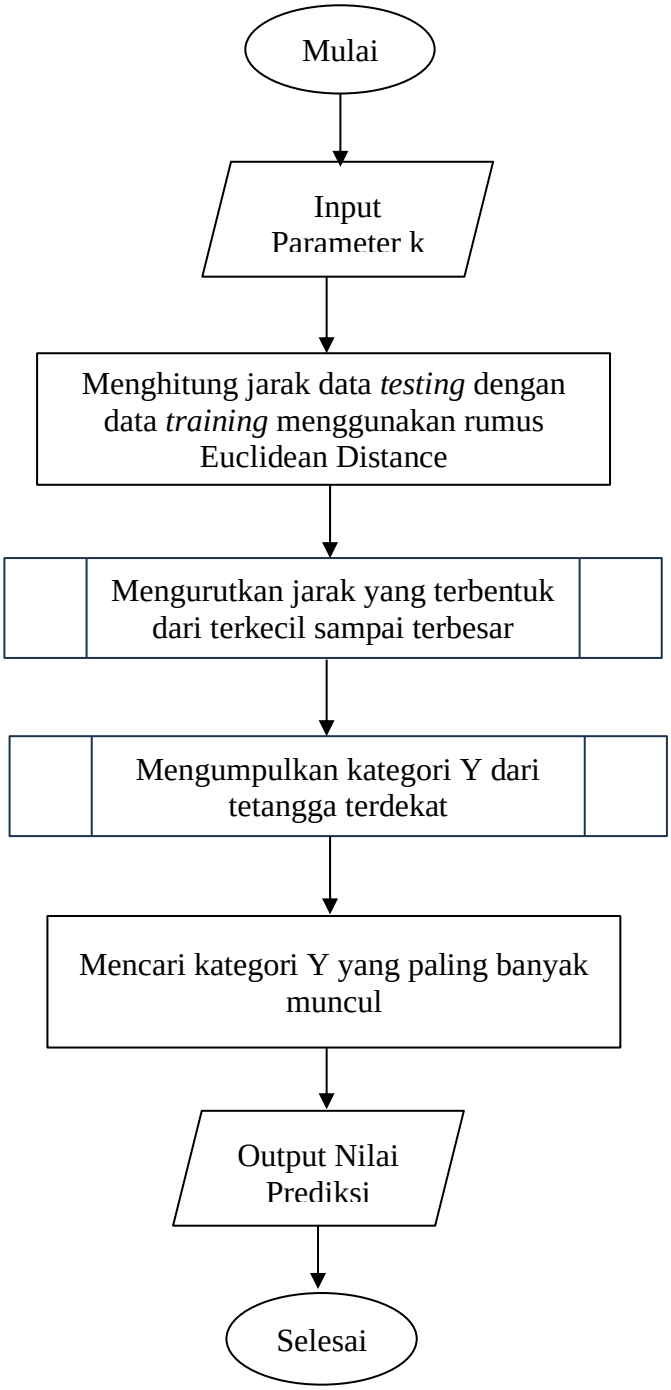
Beberapa model besar berbasis transformator yang paling menonjol adalah model BERT dan GPT. Model ini mempelajari representasi bahasa universal dari dataset besar yang tidak berlabel dan menyaring pengetahuan ke aplikasi *downstream* pada data berlabel. Pra-pelatihan BERT atau GPT adalah *fine-tuned* pada aplikasi *downstream* spesifik. Tugas NLP terbagi menjadi dua kategori: (1) tugas diskriminatif merangkum urutan masukan atau mengklasifikasikan kalimat. Model BERT banyak digunakan untuk tugas semacam ini; (2) tugas generatif menggunakan model GPT untuk merangkum urutan masukan dan menghasilkan token baru.

1. BERT: model bahasa awal dirancang untuk memproses data teks secara berurutan secara searah: dari kanan ke kiri atau dari kiri ke kanan. Sebaliknya, BERT memprediksi data yang hilang berdasarkan kata sebelumnya dan kata berikutnya dalam urutan masukan; oleh karena itu, dinamakan dua arah. Model BERT hanya terdiri dari model encoder dari transformator asli. Model ini menutupi 15% kata dalam data urutan masukan.
2. GPT: adalah LLM yang dilatih sebelumnya tanpa pengawasan pada beragam data teks untuk melakukan tugas prediktif. GPT hanya mempertahankan decoder yang berisi pengkodean posisi (*positional encoding*), masked MHA, FFN, dan operasi normalisasi. Varian GPT termasuk GPT-1, GPT-2, GPT-3, dan lain-lain. Model GPT digunakan dalam berbagai aplikasi *real-world*, seperti ChatGPT.

2.4. *K-Nearest Neighbor* (kNN)

Algoritma k-nearest neighbor (kNN) adalah algoritma pembelajaran mesin terawasi (*supervised maching learning*) yang umumnya digunakan untuk tujuan klasifikasi teks, pengenalan pola, penambangan data, dan machine learning. Pada klasifikasi teks atau dokumen, kNN memprediksi klasifikasi data tidak berlabel dengan mempertimbangkan fitur dan label data pelatihan. Ketika diberikan sebuah *instance* pengujian, algoritma klasifikasi kNN terlebih dahulu menghitung kemiripan antara *instance* yang diberikan dan semua *instance* dalam himpunan pelatihan, kemudian hasil *query instance* yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam kNN (Song et al., 2022). Di antara semua algoritma pembelajaran mesin, algoritma kNN adalah salah satu bentuk yang paling sederhana dan banyak digunakan dalam tugas klasifikasi karena sederhana dalam cara kerja dan penghitungannya, efektif jika data besar, performa cukup baik, tahan terhadap data latih yang *noisy* (Uddin et al., 2022). Algoritma kNN memiliki kelemahan di antaranya adalah perlu menentukan jumlah tetangga terdekat dari target data yang disimbolkan dengan nilai

parameter k, data training yang didasarkan pada hasil perhitungan jarak kurang akurat karena harus memilih, mencoba, dan menentukan jenis jarak yang digunakan dan atribut mana yang dipakai untuk mendapatkan hasil dengan perhitungan jarak yang terbaik, waktu komputasi tinggi jika data latih besar disebabkan karena semua data diukur jaraknya untuk setiap data uji, sangat sensitive dengan ciri yang redundan atau tidak relevan sehingga perlu ditanggulangi dengan seleksi ciri atau pembobotan ciri (Uddin et al., 2022). Tahapan algoritma kNN ditunjukkan dalam *flowchart* pada Gambar 3 (Cholil et al., 2021).



Gambar 3. Tahapan Algoritma kNN

2.4.1. Mengukur Kinerja Algoritma kNN dengan *Confusion Matrix*

Pengukuran terhadap kinerja suatu sistem klasifikasi merupakan hal yang penting. Kinerja sistem klasifikasi menggambarkan seberapa baik sistem dalam mengklasifikasikan data. *Confusion matrix* merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi (Uddin et al., 2022). Pada penelitian ini *confusion matrix* digunakan untuk mengukur kinerja algoritma kNN pada klasifikasi publikasi penelitian, meliputi pengukuran *accuracy*, *precision*, dan *recall*.

Terdapat empat istilah pengukuran kinerja menggunakan *confusion matrix*, yaitu: (1) *True Positive*, (2) *False Positive*, (3) *False Negative*, (4) *True Negative*, yang disajikan pada Gambar 4.

		Actual Class Values	
		1	0
Predicted Class Values	1	True Positive (TP)	False Positive (FP)
	0	False Negative (FN)	True Negative (TN)

Gambar 4. *Confusion Matrix* (Uddin et al., 2022)

Jika klasifikasi diprediksi 1 dan nilai sebenarnya adalah 1, maka hasilnya diklasifikasikan sebagai *True Positive* (TP). Prinsip yang sama berkisar pada nilai 0 dan diklasifikasikan sebagai *True Negative* (TN). Jika prediksinya 1 dan nilai sebenarnya 0, hasilnya diklasifikasikan sebagai *False Positive* (FP), dan kebalikannya disebut *False Negative* (FN). Ukuran akurasi dihitung dengan mengambil semua prediksi yang benar dan membaginya dengan nilai prediksi, termasuk prediksi sebenarnya.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

Pengukuran presisi dihitung dengan mengambil nilai true positive dan membaginya dengan nilai *true* dan *false positive*.

$$Pecision = \frac{TP}{TP + FP} \tag{7}$$

Pengukuran recall dihitung serupa dengan pengukuran presisi dengan mengambil nilai *true positive* dan membaginya dengan nilai *true positive* dan *false negative*.

$$Recall = \frac{TP}{TP + FN} \quad (8)$$

2.5. Perbandingan Tinjauan

Klasifikasi artikel penelitian ke dalam kategori dengan pemberian label dapat meningkatkan efisiensi berbagai perpustakaan digital. Penelitian (Mustafa et al., 2021) menyajikan model klasifikasi yang melakukan klasifikasi artikel penelitian dari dua dataset bidang ilmu computer (JUCS dan ACM) dengan bantuan metadata dan kombinasinya. Penelitian ini menggunakan model Word2Vec untuk representasi teks yang menangkap konteks semantik teks. Untuk mengatasi masalah dalam menemukan ambang batas, penelitian telah mengusulkan metode untuk menentukan nilai ambang batas untuk setiap kategori berdasarkan pemeriksaan kumpulan data secara menyeluruh. Eksperimen menghasilkan akurasi rata-rata 0,86 dan 0,84 dengan *Single-Label Classification* (SLC) untuk JUCS dan ACM, serta 0,81 dan 0,80 dengan *Multi-Label Classification* (MLC). Pada kedua dataset, model SLC yang diusulkan meningkatkan akurasi hingga 4% sedangkan model MLC yang diusulkan meningkatkan akurasi sebesar 3%. Kekurangan penelitian tersebut adalah metode pembelajaran yang memakan waktu karena harus menghitung rata-rata kemiripan setiap artikel uji dengan artikel kategori setiap saat.

Penelitian (Rivest et al., 2021) membandingkan teknik klasifikasi *deep learning* pada lebih dari 40 juta artikel ilmiah dan puluhan ribu jurnal ilmiah. Penelitian ini mengkaji penggunaan *deep learning* dengan penggabungan bibliografi, pada dasarnya pada tingkat korpus database Scopus versi Science-Metrix. Model berkinerja terbaik ketika diberikan fitur-fitur: afiliasi penulis, nama jurnal yang dirujuk dalam bibliografi, judul referensi, abstrak publikasi, kata kunci publikasi, judul publikasi, dan klasifikasi referensi publikasi. Hasilnya menunjukkan bahwa *deep learning* cukup menjanjikan karena kinerjanya sama baiknya dengan pendekatan lainnya namun lebih banyak fleksibilitas untuk ditingkatkan lebih lanjut. Misalnya, *deep neural network* yang menggabungkan informasi dari jaringan citation kemungkinan besar akan menjadi kunci bagi algoritma klasifikasi yang lebih baik. Klasifikasi Science-Metrix merupakan klasifikasi tingkat jurnal yang tidak sempurna (53% akurat) yang digunakan sebagai rangkaian pelatihan karena tidak adanya standar emas yang berfungsi sebagai kebenaran dasar.

(Chang et al., 2022) mengumpulkan abstrak makalah jurnal pariwisata SSCI antara tahun

2010 dan 2019 dari database WoS (*Web of Science*) dan menggunakan metode klasifikasi topik baru untuk mengeksplorasi karakteristik kosakata artikel yang diklasifikasi. Korpora abstrak diberi bobot kuantitatif TF-IDF. Analisis klaster K-means hierarki kemudian dilakukan untuk mengklasifikasikan artikel secara otomatis. Berdasarkan hasil 5783 abstrak, analisis klaster mengklasifikasikan jumlah klaster K-means menjadi enam kategori: perjalanan, budaya, keberlanjutan, model, perilaku, dan hotel. Metode *cross-check* diterapkan untuk menilai konsistensi klasifikasi topik, daftar judul dan kata kunci dokumen dengan tiga jarak terkecil di setiap kategori dan menerapkan diagram strategis untuk menyajikan fitur-fitur dan kategori yang berbeda. Analisis *cross-check* terhadap judul, abstrak, dan kategori topik dari delapan belas dokumen di enam topik menegaskan bahwa empat artikel memiliki relevansi rendah dengan kata-kata utama dari topik spesifik Perjalanan, Perilaku, dan Hotel. Penggunaan metode cross-check akan menghasilkan konsistensi judul, abstrak, dan kata kunci untuk Sebagian besar topik dan artikel. Bantuan dan pendapat pakar domain dapat membantu memastikan klasifikasi topik dan isi dokumen.

Terdapat dua jenis pendekatan yang digunakan untuk menganalisis publikasi ilmiah: klasifikasi pada Tingkat jurnal dan klasifikasi pada tingkat artikel. (Jeangirard, 2021) mengusulkan pendekatan campuran, memanfaatkan teknik embedding di NLP untuk melatih pengklasifikasi metadata artikel (judul, abstrak, kata kunci) diberi label dengan klasifikasi Tingkat jurnal FoR (*Fields of Research*) dan kemudian menerapkan pengklasifikasi ini di level artikel. Klasifikasi menggunakan pendekatan dalam konteks publiksi biomedis menggunakan metadata dari pengklasifikasi Fasttext Pubmed dilatih dengan kode FoR dan digunakan untuk mengklasifikasikan publikasi berdasarkan metadata yang tersedia. Hasil menunjukkan bahwa penggunaan strategi pengambilan sampel stratifikasi untuk pelatihan membantu mengurangi bias karena distribusi bidang yang tidak seimbang.

(Roman et al., 2021) menyelidiki secara kritis kumpulan data yang tersedia untuk *citation intent* dan mengusulkan teknik *citation intent* otomatis untuk memberi label konteks *citation* dengan *citation intent*. Peneliti menganotasi sepuluh juta konteks *citation* (kutipan) dengan *citation intent* dari kumpulan data *Citation Context Dataset* (C2D) dengan bantuan metode yang diusulkan. Peneliti menerapkan motode word embedding Global Vectors (GloVe), Infersent, dan BERT serta membandingkan pengukuran Precision, Recall, dan F1. Ditemukan bahwa embedding BERT berkinerja jauh lebih baik, dengan skor Presisi 89%. Kumpulan data berlabel yang tersedia secara gratis untuk tujuan penelitian, akan meningkatkan studi analisis konteks *citation*.

Tabel 2. Perbandingan Tinjauan

Penelitian	**Dataset**	**Tipe Klasifikasi**	**Metode Klasifikasi**	***Word Embedding***	**Disiplin Ilmu**
(Mustafa et al., 2021)	ACM & JUCS	Klasifikasi berbasis metadata (judul, abstrak, kata kunci)	Single dan Multi-label classification algorithm	Word2Vec	Ilmu Komputer
(Rivest et al., 2021)	Multi Jurnal	Klasifikasi berbasis bibliografi	*Convolutional Neural Network* (CNN)	Character Embedding	Multi disiplin ilmu
(Chang et al., 2022)	Jurnal Pariwisata yang terindeks SSCI	Klasifikasi berbasis konten (abstrak)	Hierarchical K-Means	TF-IDF	Pariwisata
(Jeangirard, 2021)	ERA 2018 Journal List	Klasifikasi berbasis metadata (judul, abstrak, kata kunci, MeSH)	Content-based deep learning	Fasttext	Biomedis
(Roman et al., 2021)	Citation Context Dataset (C2D)	Klasifikasi berbasis metadata (latar belakang, metode, hasil)	K-Means, HDBSCAN	Glove, Infersent, BERT	Multi disiplin ilmu
Penelitian yang Diusulkan	Jurnal Terakreditasi Sinta	Klasifikasi berbasis konten (kesimpulan)	K-Nearest Neighbor (KNN)	Large Language Model (LLM)	Rekayasa Perangkat Lunak (*Software Engineering*)

BAB III
METODOLOGI

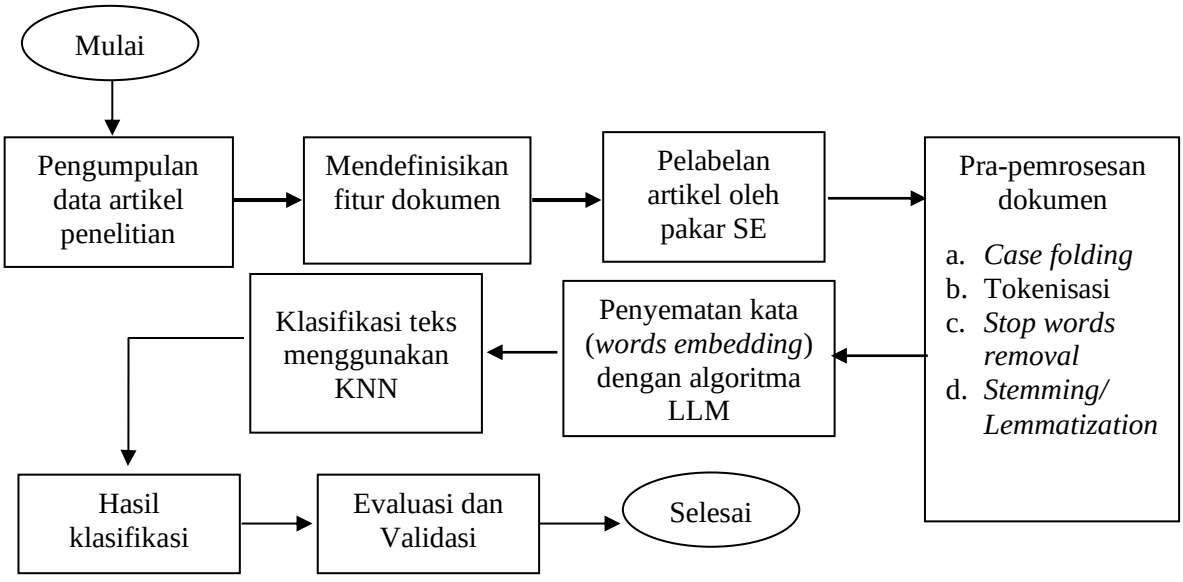
3.1. Motivasi

Klasifikasi dokumen adalah suatu teknik *text mining* yang digunakan untuk membagi dokumen teks ke dalam kelompok-kelompok yang berbeda berdasarkan tema atau kategori yang relevan (Yelmen et al., 2023). *Text mining* menggunakan berbagai macam teknik dan algoritma, seperti pemrosesan bahasa alami (NLP), analisis korpus, atau pembelajaran mesin, untuk mengolah dan menganalisis teks secara otomatis (Hossain et al., 2021).

Proses *text mining* melibatkan beberapa tahapan, seperti *pre-processing* data teks, ekstraksi fitur, dan aplikasi algoritma pembelajaran mesin. Pada tahap *pre-processing*, data teks diperoleh dan dibersihkan dari *noise* atau karakter yang tidak relevan. Setelah itu, fitur-fitur yang dianggap penting dari data teks tersebut diekstrak, seperti kata-kata yang sering muncul atau entitas yang disebutkan. Kemudian, algoritma pembelajaran mesin seperti klasifikasi diterapkan untuk memproses data teks dan menemukan pola atau hubungan di dalamnya. Hasil dari proses *text mining* kemudian dapat digunakan untuk mengambil keputusan atau menyusun laporan (Magadum, 2023).

3.2. Framework Riset

Klasifikasi artikel publikasi penelitian dilakukan mengikuti Langkah-langkah dalam alur penelitian pada Gambar 5.



Gambar 5. Alur Penelitian

3.2.1. Pengumpulan Data Artikel Penelitian

Dokumen artikel penelitian bidang Rekayasa Perangkat Lunak (*Software Engineering*) dikumpulkan dari database jurnal terakreditasi Sinta pada sistem informasi penelitian berbasis web yang dibuat oleh Kemdikbud untuk kategori jurnal Sinta 1 sampai dengan Sinta 6 periode tahun 2020 sampai dengan 2024.

3.2.2. Mendefinisikan Fitur Dokumen

Fitur dokumen merujuk pada atribut atau karakteristik spesifik yang diekstrak dari dokumen untuk digunakan dalam analisis atau pemrosesan lebih lanjut. Fitur yang digunakan dalam penelitian ini meliputi judul, abstrak, dan kata kunci yang terdapat pada setiap artikel penelitian. Kumpulan besar teks yang digunakan sebagai basis data untuk analisis linguistik atau pelatihan model NLP disebut sebagai korpus.

3.2.3. Pelabelan Artikel oleh Pakar Rekayasa Perangkat Lunak

Pakar memiliki pengetahuan mendalam dan pengalaman dalam bidangnya, sehingga mampu memberikan label yang akurat pada artikel penelitian. Pelabelan oleh pakar membantu memastikan bahwa data pelatihan yang digunakan dalam model klasifikasi memiliki kualitas tinggi dan dapat diandalkan. Dengan adanya label dari pakar Rekayasa Perangkat Lunak, hasil klasifikasi yang dihasilkan oleh model dapat divalidasi dan dibandingkan dengan pelabelan yang telah dilakukan sebelumnya.

3.2.4. Pra-Pemrosesan Dokumen

Dalam linguistik komputasi, sekumpulan besar dokumen buatan manusia yang terorganisir disebut sebagai korpus teks (Kononova et al., 2021). Pra-pemrosesan (*pre-processing*) bertujuan memproses korpus mencakup *case folding*, tokenisasi, *stop words removal*, *stemming*/ *lemmatization*. Pra-pemrosesan teks dilakukan dengan paket *Natural Language ToolKit* (NLTK) pada Python.

a. *Case folding*: merupakan tahap untuk merubah semua huruf kapital yang ada dalam dokumen menjadi huruf kecil.
b. Tokenisasi: pemotongan kata dalam kalimat, paragraf, maupun halaman menjadi potongan kata dasar atau kata tunggal dan menghilangkan delimiter seperti tanda titik (.), koma (,), at (@), dan (&), seru (!), tanya (?), titik dua (:), tab, enter, dan karakter angka yang ada pada dokumen tersebut.

c. *Stop words removal*: kata hasil tokenisasi akan dibandingkan dengan kata yang ada di dalam kamus stop words. Jika kata tersebut ada dalam *stop words* maka kata tersebut akan dihapus, jika tidak maka disimpan.
d. *Stemming*: merupakan proses untuk memetakan berbagai variasi morfologikal dari kata menjadi bentuk dasar yang sama, dengan menghilangkan semua imbuhan baik yang terdiri dari awalan, sisipan, akhiran, dan kombinasi dari awalan dan akhiran pada kata berimbuhan. Untuk *stemming* dalam Bahasa Indonesia, terdapat dua jenis metode *stemming* yang sudah ada, yaitu algoritma *stemming* yang berbasis kamus (*dictionary based*) dan algoritma *stemming* yang berbasis non-kamus (*purely rule based*). Penelitian ini menggunakan algoritma stemming berbasis kamus, yaitu dengan algoritma Nazief dan Andriani, karena memiliki akurasi paling tinggi dan overstemming paling sedikit dibandingkan dengan algoritma berbasis non-kamus seperti algoritma Vega dan Tala (Simarangkir, 2017).

3.2.5. Penyematan Kata (*Word Embedding*) dengan Algoritma *Large Language Model* (LLM)

Penyematan kata (*word embeddings*) merupakan proses konversi kata yang berupa karakter *alphanumeric* ke dalam vektor. Dengan *word embedding*, kata-kata yang memiliki *semantic meaning* yang sama, berada tidak jauh satu sama lain pada *space* tersebut. *Large Language Model* (LLM) dapat mengklasifikasikan teks dengan makna yang serupa. LLM merupakan sebuah algoritma pembelajaran mendalam (*deep learning*) yang dapat mengenali, meringkas, menerjemahkan, memprediksi, dan menghasilkan teks dan konten lain berdasarkan pengetahuan yang diperoleh dari kumpulan data besar (Ouyang et al., 2022).

Pada tahun 2020, OpenAI merilis model bahasa terbesar hingga saat ini, *Generative Pre-trained Transformer* (GPT-3), yang dilatih pada data teks dalam jumlah besar dan mampu menghasilkan teks yang sangat koheren dan terdengar alami (Brown et al., 2020). GPT-3 menunjukkan potensi LLM untuk berbagai tugas NLP (Floridi & Chiriatti, 2020). GPT-3 mengikuti arsitektur berbasis transformator, yang memungkinkannya menangkap pola dan ketergantungan linguistik yang kompleks dalam teks (Chavez et al., 2023). Model terdiri dari tumpukan lapisan transformator, memungkinkannya memproses dan menghasilkan teks dalam berbagai tingkat abstraksi. Proses GPT-3 melibatkan pembelajaran tanpa pengawasan (*unsupervised learning*) pada kumpulan besar data teks yang tersedia untuk umum dari internet. Dengan memanfaatkan ukurannya yang sangan besar dan data pelatihan yang luas, GPT-3 telah memperoleh pemahaman bahasa yang luas dan dapat menghasilkan teks mirip manusia di berbagai topik (Hassani & Silva, 2023).

Setelah GPT-3 dipilih untuk *embedding* teks yang dibutuhkan pada penelitian ini, selanjutnya akan diinstal pustaka 'transformers' dari Hugging Face untuk mendukung berbagai alur kerja *machine learning*, sehingga dapat mempercepat proses pengembangan, meningkatkan performa model, dan mengintegrasikan solusi AI yang lebih efisien dan efektif.

3.2.6. Klasifikasi Teks Menggunakan k-Nearest Neighbor (k-NN)

Setelah *embedding* teks dilakukan, langkah selanjutnya adalah melatih model klasifikasi dengan metode k-NN menggunakan *embedding* yang telah dihasilkan dari GPT-3. Langkah-langkah klasifikasi teks dilakukan menggunakan algoritma k-NN yang telah ditunjukkan pada Gambar 3.

3.2.7. Evaluasi dan Validasi

Confusion matrix digunakan untuk mengukur kinerja algoritma kNN pada klasifikasi publikasi penelitian, meliputi pengukuran: (1) *accuracy*: didefinisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai aktual; (2) *precision*: adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem; dan (3) *recall*: adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi (Uddin et al., 2022).

Cross-validation (validasi silang) merupakan sebuah teknik validasi model untuk menilai bagaimana hasil statistik analisis akan men-generalisasi kumpulan data independen. Teknik ini utamanya digunakan untuk melakukan prediksi model dan memperkirakan seberapa akurat sebuah model prediktif ketika dijalankan dalam praktiknya. Salah satu teknik dari validasi silang adalah *k-fold cross-validation*, yang mana memecah data menjadi K bagian set data dengan ukuran yang sama (Wong & Yeh, 2020).

3.3. Pendekatan

Pendekatan *Large Language Model* (LLM) pada *word embedding* dilakukan dalam penelitian ini. Model transformer LLM, GPT-3, menunjukkan potensi LLM untuk berbagai tugas NLP (Floridi & Chiriatti, 2020). GPT-3 mengikuti arsitektur berbasis transformator, yang memungkinkannya menangkap pola dan ketergantungan linguistik yang kompleks dalam teks (Chavez et al., 2023). Pustaka 'transformers' dari Hugging Face diinstal untuk mendukung berbagai alur kerja *machine learning*, sehingga dapat mempercepat proses pengembangan, meningkatkan performa model, dan mengintegrasikan solusi AI yang lebih efisien dan efektif.

DAFTAR PUSTAKA

Agrawal, S., Trenkle, J., & Kawale, J. (2023). Beyond Labels: Leveraging Deep Learning and LLMs for Content Metadata. *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023*, 74–77. https://doi.org/10.1145/3604915.3608883

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language models are few-shot learners -- special version. *Conference on Neural Information Processing Systems (NeurIPS 2020)*, *NeurIPS*, 1–25.

Chang, I. C., Horng, J. S., Liu, C. H., Chou, S. F., & Yu, T. Y. (2022). Exploration of Topic Classification in the Tourism Field with Text Mining Technology—A Case Study of the Academic Journal Papers. *Sustainability (Switzerland)*, *14*(7). https://doi.org/10.3390/su14074053

Chavez, Butler, Rekawek, Heo, K. (2023). Chat generative pre-trained transformer: why we should embrace this technology. *American Journal of Obstetrics and Gynecology*, *228*(6), 706–711. https://doi.org/10.1016/j.ajog.2023.03.010

Chernyavskiy, A., Ilvovsky, D., & Nakov, P. (2021). *Transformers: "The End of History" for NLP?* http://arxiv.org/abs/2105.00813

Chitty-Venkata, K. T., Mittal, S., Emani, M., Vishwanath, V., & Somani, A. K. (2023). A survey of techniques for optimizing transformer inference. *Journal of Systems Architecture*, *144*(9), 1–43. https://doi.org/10.1016/j.sysarc.2023.102990

Cholil, S. R., Handayani, T., Prathivi, R., & Ardianita, T. (2021). Implementasi Algoritma Klasifikasi K-Nearest Neighbor (KNN) Untuk Klasifikasi Seleksi Penerima Beasiswa. *IJCIT (Indonesian Journal on Computer and Information Technology)*, *6*(2), 118–127. https://doi.org/10.31294/ijcit.v6i2.10438

Clear, A., & Parrish, A. (2021). Computing Curricula 2020. Paradigms for global computing education. In *Computing Curricula 2020*. https://dl.acm.org/doi/book/10.1145/3467967

Fields, J., Chovanec, K., & Madiraju, P. (2024). A Survey of Text Classification With Transformers: How Wide? How Large? How Long? How Accurate? How Expensive? How Safe? *IEEE Access*, *12*, 6518–6531. https://doi.org/10.1109/ACCESS.2024.3349952

Floridi, L., & Chiriatti, M. (2020). GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds and Machines*, *30*(4), 681–694. https://doi.org/10.1007/s11023-020-09548-1

Galanis, N. I., Vafiadis, P., Mirzaev, K. G., & Papakostas, G. A. (2021). Machine Learning Meets Natural Language Processing - The Story so Far. *IFIP Advances in Information and Communication Technology*, *627*, 673–686. https://doi.org/10.1007/978-3-030-79150-6_53

Hassani, H., & Silva, E. S. (2023). The Role of ChatGPT in Data Science: How AI-Assisted Conversational Interfaces Are Revolutionizing the Field. *Big Data and Cognitive Computing*, *7*(2). https://doi.org/10.3390/bdcc7020062

Hossain, A., Karimuzzaman, M., Hossain, M. M., & Rahman, A. (2021). Text mining and sentiment analysis of newspaper headlines. *Information (Switzerland)*, *12*(10). https://doi.org/10.3390/info12100414

IEEE Computer Society. (2024). *Guide to the software engineering body of knowledge v.40* (IEEE Computer Society (ed.)). https://www.computer.org/volunteering/boards-and-committees/professional-educational-activities/software-engineering-committee/swebok-evolution

Jeangirard, E. (2021). *Content-based subject classification at article level in biomedical context*. 0–13. http://arxiv.org/abs/2104.14800

Ji, J., Li, Z., Xu, S., Hua, W., Ge, Y., Tan, J., & Zhang, Y. (2024). GenRec: Large Language Model for Generative Recommendation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *14610 LNCS*, 494–502. https://doi.org/10.1007/978-3-031-56063-7_42

Kononova, O., He, T., Huo, H., Trewartha, A., Olivetti, E. A., & Ceder, G. (2021). Opportunities and challenges of text mining in aterials research. *IScience*, *24*(3), 102155. https://doi.org/10.1016/j.isci.2021.102155

Magadum, T. (2023). Smart Document Classification. *Interantional Journal of Scientific Research in Engineering and Management*, *07*(08), 3–7. https://doi.org/10.55041/ijsrem22489

Mustafa, G., Usman, M., Yu, L., afzal, M. T., Sulaiman, M., & Shahid, A. (2021). Multi-label classification of research articles using Word2Vec and identification of similarity threshold. *Scientific Reports*, *11*(1), 1–20. https://doi.org/10.1038/s41598-021-01460-7

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2023). *A Comprehensive Overview of Large Language Models*. http://arxiv.org/abs/2307.06435

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens,

M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, *35*.

Paaß, G., & Giesselbach, S. (2023). *Pre-trained Language Models*. https://doi.org/10.1007/978-3-031-23190-2_2

Rivest, M., Vignola-Gagné, E., & Archambault, É. (2021). Article-level classification of scientific publications: A comparison of deep learning, direct citation and bibliographic coupling. *PLoS ONE*, *16*(5 May), 1–18. https://doi.org/10.1371/journal.pone.0251493

Roihan, A., Sunarya, P. A., & Rafika, A. S. (2020). Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper. *IJCIT (Indonesian Journal on Computer and Information Technology)*, *5*(1), 75–82. https://doi.org/10.31294/ijcit.v5i1.7951

Roman, M., Shahid, A., Khan, S., Koubaa, A., & Yu, L. (2021). Citation Intent Classification Using Word Embedding. *IEEE Access*, *9*, 9982–9995. https://doi.org/10.1109/ACCESS.2021.3050547

Simarangkir, M. S. H. (2017). Studi perbandingan algoritma - algoritma. *Jurnal Inkofar*, *1*(1), 40–46.

Song, Y., Kong, X., & Zhang, C. (2022). A Large-Scale k -Nearest Neighbor Classification Algorithm Based on Neighbor Relationship Preservation. *Wireless Communications and Mobile Computing*, *2022*. https://doi.org/10.1155/2022/7409171

Uddin, S., Haque, I., Lu, H., Moni, M. A., & Gide, E. (2022). Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction. *Scientific Reports*, *12*(1), 1–11. https://doi.org/10.1038/s41598-022-10358-x

Usman Hadi, M., al tashi, qasem, Qureshi, R., Shah, A., muneer, amgad, Irfan, M., Zafar, A., Bilal Shaikh, M., Akhtar, N., Wu, J., Mirjalili, S., Al-Tashi, Q., & Muneer, A. (2023). A Survey on Large Language Models: Applications, Challenges, Limitations, and Practical Usage. *Authorea Preprints*. https://www.authorea.com/doi/full/10.36227/techrxiv.23589741.v1?commit=b1cb46f5b0f749cf5f2f33806f7c124904c14967

Wan, M., Safavi, T., Jauhar, S. K., Kim, Y., Counts, S., Neville, J., Suri, S., Shah, C., White, R. W., Yang, L., Andersen, R., Buscher, G., Joshi, D., & Rangan, N. (2024). TnT-LLM: Text Mining at Scale with Large Language Models. In *Proceedings of ACM Conference (Conference'17)* (Vol. 1, Issue 1). Association for Computing Machinery. http://arxiv.org/abs/2403.12173

Wong, T., Yeh, P. (2020). Reliable Accuracy Estimates from k-Fold Cross Validation. *IEEE*

Transactions on Knowledge and Data Engineering, *32*(8), 1586–1594. https://doi.org/10.1109/TKDE.2019.2912815

Yelmen, I., Gunes, A., & Zontul, M. (2023). Multi-Class Document Classification Using Lexical Ontology-Based Deep Learning †. *Applied Sciences (Switzerland)*, *13*(10). https://doi.org/10.3390/app13106139

Zhdanovskaya, A., Baidakova, D., & Ustalov, D. (2023). Data Labeling for Machine Learning Engineers: Project-Based Curriculum and Data-Centric Competitions. *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, *37*, 15886–15893. https://doi.org/10.1609/aaai.v37i13.26886