https://jurnal.pustakagalerimandiri.co.id/index.php/pustakaai
DOI : https://doi.org/10.55382/jurnalpustakaai.v5i2.1069

JURNAL PUSTAKA AI
JURNAL PUSAT AKSES KAJIAN TEKNOLOGI ARTIFICIAL INTELLIGENCE
Pustaka Galeri Mandiri

Vol. 5 No. 2 (2025) 418 – 427 E ISSN: 2809-4069

Pengaruh Few-shot Learning pada Kinerja LLM untuk Ekstraksi Entitas Iklan Lowongan Kerja

Alvalen Shafelbilyunazra[1], Didik Dwi Prasetya[2]
[12]Departemen Teknik Elektro dan Informatika, Fakultas Teknik, Universitas Negeri Malang
[1]alvalen.shafelshafelbilyunazra.2205356@students.um.ac.id , [2]didikdwi@um.ac.id

Abstract

Extracting information from unstructured text, such as job advertisements, poses a significant challenge. Traditional fine-tuning approaches demand massive labelled datasets and substantial computational resources. Alternatively, modern Large Language Models (LLMs) offer a more efficient In-Context Learning (ICL) paradigm. This study systematically investigates the impact of few-shot learning techniques, specifically varying the number of examples (*k*), on LLM accuracy in entity extraction from Indonesian job advertisements. Using the generative Gemini model, experiments were conducted from zero-shot (*k=0*) to few-shot scenarios (*k=1, 3, 5, 10, 20*). Each scenario was evaluated five times using Monte Carlo Cross-Validation, measuring performance with Precision, Recall, and F1-Score. Results indicate a strong positive correlation between the number of examples and accuracy, but with a point of diminishing returns. The most drastic performance improvement occurred with 1 to 5 shots, and performance saturated after 10 shots, where further examples provided only marginal gains. Additionally, the model consistently exhibited higher Precision than Recall, suggesting a conservative tendency to prioritize the correctness of extracted data. This study concludes that an optimal prompting strategy balances accuracy and efficiency, recommending 5 to 10 examples for most applications. These findings offer data-driven practical guidance for developers to effectively and efficiently optimize LLM usage in information extraction tasks.

Keywords: Entity Extraction; Few-shot Learning; In-Context Learning; Large Language Model (LLM); Prompt Engineering.

Abstrak

Ekstraksi informasi dari teks tidak terstruktur, seperti iklan lowongan kerja, merupakan tantangan besar. Pendekatan tradisional berbasis fine-tuning membutuhkan dataset berlabel masif dan sumber daya komputasi tinggi. Sebagai alternatif, Large Language Model (LLM) dengan In-Context Learning (ICL) menawarkan efisiensi. Penelitian ini menginvestigasi pengaruh few-shot learning, khususnya variasi jumlah contoh (k), terhadap akurasi LLM dalam ekstraksi entitas dari iklan lowongan kerja berbahasa Indonesia. Menggunakan model Gemini, eksperimen dilakukan dengan skenario zero-shot (k=0) hingga few-shot (k=1, 3, 5, 10, 20). Setiap skenario dievaluasi lima kali menggunakan Monte Carlo Cross-Validation, dengan metrik Presisi, Recall, dan F1-Score. Hasil menunjukkan korelasi positif antara jumlah contoh dan akurasi, namun dengan point of diminishing returns. Peningkatan kinerja drastis terjadi pada 1-5 shot, dan performa mencapai kejenuhan setelah 10 shot. Model cenderung memiliki Presisi lebih tinggi daripada Recall, memprioritaskan kebenaran ekstrak. Studi ini menyimpulkan bahwa strategi prompting optimal memerlukan keseimbangan akurasi dan efisiensi, merekomendasikan 5-10 contoh untuk sebagian besar aplikasi. Temuan ini memberikan panduan praktis untuk optimalisasi penggunaan LLM dalam ekstraksi informasi.

Kata kunci: Ekstraksi Entitas; Few-shot Learning; In-Context Learning; Large Language Model (LLM); Prompt Engineering.

Submitted : 05-08-2025 | Reviewed : 16-08-2025 | Accepted : 31-08-2025

1. Pendahuluan

Revolusi digital telah mengubah lanskap ekonomi global, mendorong perusahaan untuk beradaptasi dengan cepat demi mempertahankan keunggulan kompetitif. Di jantung transformasi ini, manajemen sumber daya manusia (SDM) memegang peranan krusial dalam mengakuisisi talenta terbaik [1]. Portal lowongan kerja daring seperti Jobstreet, Kalibrr, dan LinkedIn telah menjadi ekosistem utama yang menghubungkan perusahaan dengan pencari kerja, menghasilkan volume data yang sangat besar setiap harinya [2]. Data ini, terutama dalam bentuk iklan lowongan kerja, mengandung informasi vital mengenai tren kebutuhan industri, kualifikasi yang dicari, serta penawaran kompensasi [3]. Namun, informasi berharga ini sering kali terperangkap dalam format teks yang tidak terstruktur, menjadikannya sulit untuk dianalisis secara sistematis dan dalam skala besar.

Tantangan utama dalam mengolah data iklan lowongan kerja terletak pada sifatnya yang tidak terstruktur. Setiap iklan memuat beragam informasi seperti judul posisi, nama perusahaan, lokasi penempatan, rentang gaji, kualifikasi pendidikan dan syarat teknis, hingga persyaratan pengalaman kerja yang disajikan dalam bentuk kalimat naratif atau poin-poin yang bervariasi antar platform [4]. Proses mengekstrak informasi-informasi inti ini secara manual membutuhkan tenaga ahli yang memahami terminologi HR, waktu yang cukup panjang untuk membaca dan menandai setiap iklan, serta sistem pengecekan ulang untuk meminimalkan kesalahan manusia [5]. Akibatnya, skala operasi terbatas, *throughput* rendah, dan konsistensi data sulit dijaga, apalagi ketika volume iklan mencapai ribuan per hari.

Sebagai respons, dunia akademik dan industri telah lama mengandalkan pendekatan *supervised learning* dalam *Named Entity Recognition* (NER). Model-model berbasis arsitektur Transformer seperti BERT, RoBERTa, dan lainnya dilatih melalui proses *fine-tuning* pada korpus iklan beranotasi ribuan hingga puluhan ribu contoh [6]. Meskipun menunjukkan akurasi tinggi pada dataset pengujian, strategi ini menuntut investasi besar dalam pembuatan dataset: ahli anotator harus menandai setiap kata atau frasa sesuai kategori entitas, lalu hasilnya harus diuji ulang untuk menjamin kualitas label [7]. Selain itu, proses pelatihan memerlukan infrastruktur komputasi berat dengan GPU/TPU dan waktu pelatihan berjam-jam hingga berhari-hari [8]. Beban biaya dan waktu inilah yang menjadi hambatan utama bagi organisasi, terutama skala menengah ke bawah, untuk mengadopsi solusi NER tradisional pada domain lowongan kerja yang terus berkembang dinamis.

Sebagai alternatif yang menjanjikan terhadap pendekatan tradisional berbasis *fine-tuning*, kemunculan *Large Language Model* (LLM) modern seperti seri *Generative Pre-trained Transformer* (GPT) dan Gemini telah memperkenalkan sebuah paradigma baru yang dikenal sebagai *In-Context Learning* [9]. Berbeda dari pendekatan konvensional yang membutuhkan pelatihan ulang (*fine-tuning*) dengan dataset beranotasi secara khusus, LLM *pre-trained seperti* GPT dan Gemini dapat menyelesaikan beragam tugas pemrosesan bahasa alami (NLP) tanpa perlu modifikasi parameter model [10]. Dalam kerangka *In-Context Learning*, model hanya perlu diberikan instruksi dalam bahasa alami yang jelas dan beberapa contoh yang relevan di dalam *prompt* [11]. Sebuah teknik yang dikenal sebagai *prompt engineering* ini dapat membantu model LLM dalam melakukan tugas-tugas kompleks seperti ekstraksi entitas, klasifikasi teks, hingga inferensi logis. Pendekatan ini secara signifikan mereduksi hambatan teknis dan kebutuhan sumber daya komputasi, sekaligus mempercepat pengembangan dan implementasi solusi berbasis LLM di berbagai domain aplikasi.

Meskipun demikian, keefektifan LLM tidak bersifat mutlak dan sangat sensitif terhadap kualitas dan struktur *prompt* yang diberikan. Salah satu teknik *prompt* engineering yang paling kuat adalah *Few-shot* Learning, di mana kita menyertakan beberapa contoh (*shots*) input dan output yang benar di dalam *prompt* untuk memandu model menghasilkan jawaban yang akurat [12]. Studi-studi sebelumnya telah menunjukkan bahwa *few-shot* learning secara umum mampu meningkatkan kinerja LLM secara signifikan dibandingkan tanpa contoh (*zero-shot*) atau dengan satu contoh saja (*one-shot*) [13]. Namun demikian, Sebagian besar studi atau penelitian yang ada seringkali berfokus pada benchmark berbahasa Inggris atau tidak secara spesifik menganalisis dampak kuantitatif dari variasi jumlah contoh itu sendiri. Masih terdapat celah pemahaman mengenai seberapa besar peningkatan akurasi yang didapat dari penambahan setiap contoh, dan apakah ada titik di mana penambahan contoh tidak lagi memberikan manfaat yang signifikan (*point of diminishing returns*), khususnya untuk tugas ekstraksi entitas dalam Bahasa Indonesia.

Sebagai respons terhadap keterbatasan tersebut, diperlukan sebuah pendekatan sistematis yang mampu mengukur secara empiris pengaruh jumlah contoh pada skenario *few-shot* terhadap efektivitas model dalam tugas ekstraksi entitas dari teks lowongan kerja berbahasa Indonesia. Penelitian ini mengusulkan pengujian bertingkat mulai dari skenario *zero-shot*, *one-shot*, hingga *few-shot*. Evaluasi dilakukan menggunakan metrik seperti Presisi, Recall, dan F1-score untuk setiap konfigurasi, guna mengidentifikasi pola peningkatan kinerja serta mendeteksi titik kejenuhan (*point of diminishing returns*), yaitu kondisi di mana penambahan jumlah contoh tidak lagi memberikan dampak signifikan terhadap performa model. Pendekatan ini diharapkan dapat memberikan insight kuantitatif yang

Submitted : 05-08-2025 | Reviewed : 16-08-2025 | Accepted : 31-08-2025

mendalam mengenai efektivitas teknik *few-shot* learning pada LLM berbahasa Indonesia, serta menyusun panduan praktis berbasis data empiris bagi praktisi dan pengembang NLP dalam menyusun *prompt* yang efisien dan optimal untuk penerapan model dalam domain spesifik seperti analisis teks lowongan kerja.

1. Metode Penelitian

2.1. Ekstraksi Entitas

Ekstraksi Entitas, yang juga dikenal luas sebagai *Named Entity Recognition* (NER), adalah sebuah tugas fundamental dalam bidang *Natural Language Processing* (NLP). Tujuan utamanya adalah untuk mengidentifikasi dan mengklasifikasikan fragmen teks ke dalam kategori-kategori yang telah ditentukan sebelumnya [14]. Tugas ekstraksi entitas difokuskan pada pengenalan informasi inti dari teks iklan lowongan kerja. Entitas yang menjadi target ekstraksi meliputi Posisi Pekerjaan, Nama Perusahaan, Lokasi, Tipe Pekerjaan (misalnya, Full-time, Hybrid), Rentang Gaji, dan Kualifikasi Teknis.

Proses ini secara esensial berfungsi untuk mengubah data tekstual yang tidak terstruktur menjadi data terstruktur dalam format yang dapat dibaca oleh mesin, seperti JSON. Transformasi data ini sangat krusial karena memungkinkan dilakukannya analisis data lanjutan, seperti pemantauan tren kebutuhan industri, perbandingan kompensasi antar posisi, atau pembangunan sistem pencocokan kandidat otomatis [15]. Dengan demikian, akurasi dari proses ekstraksi entitas menjadi fondasi penting bagi kualitas wawasan yang dapat dihasilkan dari data lowongan kerja.

2.2. *Large Language Model* (LLM)

Judul *Large Language Model* (LLM) adalah model *deep learning* yang dibangun di atas arsitektur Transformer dan dilatih pada volume data teks yang masif, sering kali mencapai miliaran hingga triliunan parameter [16]. Proses pra-pelatihan (*pre-training*) yang ekstensif ini membekali LLM dengan pemahaman yang mendalam mengenai pola, sintaksis, semantik, dan pengetahuan umum dari bahasa manusia [17]. Kapabilitas utamanya yang adalah kemampuannya untuk melakukan berbagai tugas NLP yang kompleks, termasuk ekstraksi entitas, hanya dengan melalui instruksi bahasa alami tanpa memerlukan proses pelatihan khusus atau *fine-tuning* pada dataset spesifik.

Dalam penelitian ini, LLM dimanfaatkan sebagai sebuah layanan yang diakses melalui *Application Programming Interface* (API) dengan model Gemini 2.0 flash yang dikembangkan oleh Google. Pendekatan ini secara signifikan mereduksi hambatan komputasi, karena semua proses inferensi yang berat dijalankan di server milik penyedia layanan. Fondasi arsitektur Gemini adalah Transformer, yang pertama kali diperkenalkan oleh Vaswani et al. dalam karya seminal mereka "Attention Is All You Need" [18]. Secara spesifik, model-model Gemini menggunakan arsitektur *decoder-only*, sebuah pilihan desain yang efektif untuk tugas-tugas generatif [19], [20]. Struktur ini memproses sekuens input secara autoregresif, di mana model memprediksi token berikutnya dalam sebuah urutan dengan mempertimbangkan token-token yang telah dihasilkannya sebelumnya. Mekanisme inti di dalamnya adalah *masked self-attention*, yang memastikan bahwa saat memprediksi sebuah token, model hanya dapat "melihat" token-token sebelumnya dalam urutan, bukan token di masa depan. Untuk meningkatkan efisiensi inferensi, arsitektur ini juga mengimplementasikan optimasi seperti *multi-query attention*.

2.3. *Prompt Engineering*

Prompt Engineering adalah sebuah disiplin ilmu dan seni dalam merancang dan menyusun input (dikenal sebagai *prompt*) secara cermat untuk memandu *Large Language Model* (LLM) agar menghasilkan output yang spesifik, akurat, dan sesuai dengan keinginan [21]. Kinerja LLM pre-trained sangat sensitif terhadap cara instruksi diberikan, sehingga *prompt engineering* menjadi kunci utama untuk memaksimalkan potensi model tanpa perlu memodifikasi parameter internalnya. Sebuah *prompt* yang efektif biasanya terdiri dari beberapa komponen utama [22].

Komponen pertama adalah instruksi, yaitu perintah eksplisit yang memberitahu model tugas apa yang harus dilakukan. Komponen kedua adalah konteks atau contoh (*context/examples*), yang berfungsi sebagai panduan atau rujukan untuk menunjukkan format output yang diharapkan atau cara menangani tugas tersebut. Komponen terakhir adalah input atau pertanyaan (*input/query*), yaitu data mentah yang akan diproses oleh model. Penelitian ini berfokus pada manipulasi sistematis dari komponen kedua, yaitu jumlah contoh, untuk mengukur dampaknya terhadap akurasi ekstraksi.

2.4. *In-Context Learning* dan *Few-shot Learning*

In-Context Learning (ICL) adalah sebuah kemampuan yang dimiliki oleh LLM, di mana model dapat "belajar" untuk melakukan tugas baru pada saat inferensi (inference time) hanya dengan mengondisikan pada *prompt* yang

berisi beberapa contoh demonstrasi [23]. Proses pembelajaran ini terjadi tanpa pembaruan gradien atau perubahan pada bobot (*weights*) model. ICL adalah mekanisme yang mendasari efektivitas *prompt engineering* dan memungkinkan LLM beradaptasi dengan cepat pada berbagai tugas. Dalam penelitian ini, ICL diimplementasikan dalam beberapa skenario untuk dianalisis dan dibandingkan kinerjanya.

Skenario-skenario tersebut adalah sebagai berikut:

- *Zero-shot Learning* (k=0): Pada skenario ini, model hanya diberikan instruksi dan data input tanpa disertai contoh sama sekali. Kinerja model sepenuhnya bergantung pada pengetahuan yang telah ia pelajari selama fase *pre-training*.
- *One-shot Learning* (k=1): Model diberikan satu contoh input-output yang berkualitas tinggi selain instruksi. Contoh tunggal ini berfungsi sebagai jangkar (anchor) untuk memandu format dan gaya penalaran model.
- *Few-shot Learning* (k>1): Model diberikan lebih dari satu contoh. Hipotesisnya adalah bahwa dengan lebih banyak contoh, model mendapatkan konteks yang lebih kaya mengenai nuansa tugas, variasi data, dan struktur output yang diinginkan, yang pada akhirnya diharapkan dapat meningkatkan akurasi dan konsistensi hasil ekstraksi secara signifikan.

2.5. Dataset

Dataset yang digunakan dalam penelitian ini merupakan dataset privat yang dibangun secara khusus untuk menjawab rumusan masalah. Data dikumpulkan melalui proses web scraping dari beberapa portal lowongan kerja daring terkemuka di Indonesia, seperti Kalibrr, Jobstreet, dan Glints pada periode waktu tertentu. Proses ini bertujuan untuk mendapatkan data yang beragam dan representatif dari iklan lowongan kerja di dunia nyata.

Proses pengumpulan menghasilkan total 121 iklan lowongan kerja mentah. Setiap data mentah kemudian melalui tahap pre-processing dan seleksi kualitas untuk memastikan relevansi dan kelengkapan informasi. Dari jumlah tersebut, sebanyak 100 data dipilih untuk digunakan dalam eksperimen. Proses anotasi atau pelabelan dilakukan untuk membuat *ground truth* (kunci jawaban) bagi setiap data. Untuk mempercepat proses, draf anotasi awal dihasilkan menggunakan LLM dengan teknik *one-shot*, kemudian setiap entitas diverifikasi dan dikoreksi secara manual menjamin akurasi. Entitas yang diekstrak disimpan dalam format JSON (*JavaScript Object Notation*) yang terstruktur. Contoh data dan label dapat dilihat pada Tabel 1.

Tabel 1. Contoh Data Mentah dan Anotasi Ground truth

Tipe	Konten
Teks Mentah	Software Engineer - Backend di PT Formulatrix Indonesia PT Formulatrix Indonesia mengundang talenta terbaik untuk bergabung sebagai Software Engineer - Backend di Salatiga. Kami mencari individu yang bersemangat untuk mengembangkan sistem robotik inovatif dari awal. Posisi ini menawarkan gaji kompetitif antara Rp10.000.000 - Rp15.000.000 per bulan. Jika Anda seorang lulusan S1 dengan 1-3 tahun pengalaman dalam pengembangan perangkat lunak, terutama yang menguasai bahasa pemrograman berorientasi objek seperti C#, C++, atau Java, serta memiliki pemahaman kuat tentang Linux, *embedded systems*, dan C++, kami sangat menantikan lamaran Anda. Bergabunglah dengan tim kami untuk menciptakan solusi yang berdampak dan memajukan teknologi di industri bioteknologi

Ground truth (JSON)	'posisi': 'Software Engineer - Backend', 'perusahaan': 'PT Formulatrix Indonesia', 'lokasi': 'Salatiga, Tingkir, Jawa Tengah', 'tipe': 'Full-time', 'gaji': 'Rp 10.000.000 - 15.000.000', 'kualifikasi_teknis': 'menguasai bahasa pemrograman berorientasi objek (C#, C++, Java, Python, dll.), dan memiliki pemahaman tentang Linux, *embedded systems*, serta C++'}

Selanjutnya, dataset final sebesar 100 data ini dibagi menjadi dua himpunan yang tidak saling tumpang tindih:

- *Test Set*: Sebanyak 80 data dialokasikan sebagai *Test Set*. Himpunan ini digunakan secara eksklusif untuk mengevaluasi kinerja model pada setiap skenario pengujian.
- *Shot Pool*: Sebanyak 20 data dialokasikan sebagai *Shot Pool*. Himpunan ini berfungsi sebagai "bank contoh" yang akan dimasukkan ke dalam *prompt* untuk skenario *one-shot* dan *few-shot*.

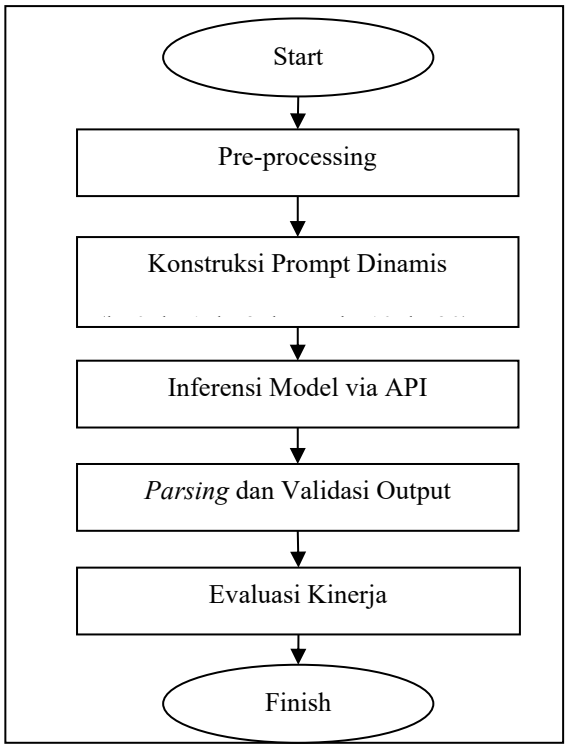
2.6. Desain Eksperimen

Dalam pelaksanaan eksperimen ini, serangkaian perangkat lunak dan keras digunakan untuk mendukung keseluruhan proses penelitian. Visual Studio Code dengan jupyter notebook dimanfaatkan sebagai editor kode utama untuk pengembangan kode dalam bahasa Python (versi 3.11), didukung oleh berbagai ekstensi yang mempermudah manajemen kode. Library inti yang digunakan meliputi pandas untuk manipulasi dan pengelolaan dataset, google-generativeai untuk berinteraksi secara terprogram dengan Application Programming Interface (API) Gemini model 2.0 flash, serta scikit-learn untuk melakukan perhitungan metrik evaluasi secara akurat.

Pada sisi perangkat keras, eksperimen ini dijalankan pada sebuah komputer personal dengan spesifikasi yang mencakup prosesor Intel Core i7-10750H, RAM 16 GB, dan penyimpanan SSD 1 TB. Spesifikasi perangkat keras ini utamanya berfungsi untuk menjalankan kode eksperimen, mengelola aliran data, dan melakukan orkestrasi pemanggilan API. Beban komputasi utama yang terkait dengan proses inferensi model sepenuhnya ditangani oleh infrastruktur server yang disediakan oleh penyedia layanan LLM, sehingga keterbatasan perangkat keras lokal tidak menjadi faktor penghambat.

Untuk memastikan variabilitas yang mungkin timbul dari pemilihan contoh dalam *prompt*, penelitian ini mengadopsi metode validasi yang dikenal sebagai Monte Carlo Cross-Validation (MCCV) [24]. Pendekatan ini penting mengingat penelitian ini bekerja dengan dataset yang berukuran relatif kecil (100 data). Pada dataset yang terbatas, kinerja model dalam skenario *few-shot learning* berisiko bias terhadap pemilihan beberapa contoh spesifik. Dengan mengulang keseluruhan proses pengujian sebanyak lima kali (5x runs) di mana pada setiap iterasinya set contoh dipilih secara acak dari *Shot Pool* metode MCCV secara efektif memitigasi risiko ini.

Alur kerja eksperimen dirancang secara sistematis untuk memproses setiap data uji dan mengevaluasi kinerja model secara objektif. Proses untuk satu data uji dapat diuraikan ke dalam beberapa tahapan utama, yang diilustrasikan pada Gambar 1.



Gambar 1. Metode Penelitian

Tahapan-tahapan dalam alur kerja tersebut adalah sebagai berikut:

Tahapan dimulai dari pre-processing, yaitu dengan memuat dataset yang telah dianotasi sebelumnya dari sebuah file CSV ke dalam struktur Pandas DataFrame. Pada tahap ini, dilakukan pembagian dataset secara deterministik sesuai dengan alokasi yang telah ditentukan, yaitu 80 data untuk *Test Set* dan 20 data untuk *Shot Pool*. Pemisahan yang jelas ini memastikan bahwa data yang digunakan untuk evaluasi tidak pernah tercampur dengan data yang digunakan sebagai contoh.

Berdasarkan data input, sebuah *prompt* dibangun secara dinamis sesuai dengan skenario pengujian yang sedang berjalan. Gambaran *prompt* yang digunakan terlihat pada Tabel 2.

Tabel 2. Prompt Model

Base Prompt	*Shots Prompt*
Anda adalah asisten AI yang ahli dalam ekstraksi informasi. Tugas Anda adalah mengekstrak entitas-entitas berikut dari teks iklan lowongan kerja dan mengembalikannya dalam format JSON yang valid. Entitas yang harus diekstrak adalah: 'posisi', 'perusahaan', 'lokasi', 'tipe' (misal: WFH, WFO, remote, full-time), 'gaji' dan 'kualifikasi_teknis'. Jika sebuah entitas tidak ditemukan dalam teks, gunakan nilai `null` dalam JSON. HANYA kembalikan blok kode JSON, tanpa teks atau penjelasan tambahan.	--- CONTOH --- Teks: “contoh teks pada data *Shot Pool* ” JSON: “contoh *ground truth* pada data example *Shot Pool*” ---SELESAI CONTOH --- --- TEKS YANG HARUS DIPROSES --- Teks: “teks input pada *Test Set*s” JSON:

- Untuk skenario *Zero-shot* (k=0), *prompt* hanya akan berisi instruksi tugas atau base *prompt* dan data input.
- Untuk skenario *One-shot* (k=1) dan *Few-shot* (k=3, k=5, k=10, k=20), *prompt* akan diperkaya dengan menyisipkan sejumlah contoh (sesuai nilai *k*) yang diambil secara acak dari *Shot Pool*.

Prompt yang telah lengkap dikirimkan sebagai permintaan (*request*) ke *endpoint* API Gemini. Proses ini diulang untuk setiap data dalam *Test Set* dan untuk setiap dari lima run eksperimen. Mekanisme ini memastikan bahwa setiap data uji dievaluasi sebanyak lima kali dengan potensi set contoh *few-shot* yang berbeda-beda, sehingga memberikan gambaran kinerja yang lebih general dan andal.

Respons teks dari Gemini diterima dan melalui tahap *parsing*. Pada tahap ini, sebuah fungsi dirancang untuk mengekstrak konten JSON dari respons teks mentah. Dilakukan juga validasi untuk memastikan bahwa output yang diekstrak memiliki struktur JSON yang valid dan sesuai dengan skema yang diharapkan.

JSON yang berhasil *diparsing* (hasil prediksi model) selanjutnya dibandingkan secara mendetail dengan data *ground truth* (kunci jawaban) yang bersesuaian untuk data uji tersebut.

2.7. Metrik Evaluasi

Untuk mengevaluasi kinerja model secara kuantitatif pada tugas ekstraksi entitas, digunakan tiga metrik standar yang diadopsi dari bidang *Information Retrieval* dan NLP. Evaluasi dilakukan pada level entitas, di mana setiap prediksi entitas dibandingkan dengan *ground truth* untuk diklasifikasikan sebagai salah satu dari tiga kategori berikut:

- True Positive (TP): Model berhasil mengekstrak sebuah entitas dengan benar (baik kategori maupun nilainya cocok dengan *ground truth*).
- False Positive (FP): Model mengekstrak sebuah entitas yang sebenarnya tidak ada dalam *ground truth* atau kategorinya salah.
- False Negative (FN): Model gagal mengekstrak sebuah entitas yang seharusnya ada menurut *ground truth*.

Berdasarkan klasifikasi tersebut, metrik-metrik berikut dihitung untuk setiap skenario:

1. Precision: Metrik ini secara spesifik mengukur tingkat keandalan atau kepercayaan dari entitas yang berhasil diekstrak oleh model. Presisi yang tinggi mengindikasikan bahwa model jarang membuat kesalahan dengan "menciptakan" data yang tidak ada atau salah mengklasifikasikan entitas, sehingga memiliki tingkat *False Positive* (FP) yang rendah.
$$Precision = \frac{TP}{TP + FP} \quad (1)$$
2. Recall: Metrik ini mengukur tingkat kelengkapan atau kesensitifan model dalam menemukan semua entitas yang relevan dari teks. *Recall* yang tinggi menunjukkan bahwa model mampu mengidentifikasi sebagian besar entitas yang seharusnya ada dan tidak banyak yang terlewat, sehingga memiliki tingkat *False Negative* (FN) yang rendah.
$$Recall = \frac{TP}{TP + FN} (2)$$
3. F1-Score: Merupakan rata-rata harmonik dari Presisi dan Recall, memberikan sebuah skor tunggal yang menyeimbangkan kedua metrik tersebut. F1-Score sangat berguna terutama jika terdapat ketidakseimbangan antara jumlah FP dan FN.
$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} (3)$$

Secara kolektif, penggunaan ketiga metrik ini memberikan pandangan diagnostik yang komprehensif terhadap perilaku model. Presisi mengukur kualitas dari apa yang ditemukan, Recall mengukur kuantitas dari apa yang seharusnya ditemukan, dan F1-Score merangkum efektivitas keseluruhan dalam satu angka.

3. Hasil dan Pembahasan

Eksperimen dilakukan dengan menjalankan pengujian pada enam skenario *k-shot learning* (k=0, 1, 3, 5, 10, dan 20). Setiap skenario diulang sebanyak lima kali (runs) dengan menggunakan metode Monte Carlo Cross-Validation untuk memastikan stabilitas dan keandalan hasil. Skor metrik yang disajikan pada Tabel 3 merupakan nilai rata-rata dari kelima run yang telah dijalankan.

Tabel 3. Hasil Metrik Evaluasi per Skenario K-Shot (5 *runs*)

Submitted : 05-08-2025 | Reviewed : 16-08-2025 | Accepted : 31-08-2025

Shots (k)	Precision	Recall	F1-Score
0	0.5716	0.5727	0.5722
1	0.7390	0.7346	0.7368
3	0.7881	0.7798	0.7839
5	0.7940	0.7850	0.7895
10	0.8025	0.7990	0.8007
20	0.8047	0.7998	0.8023

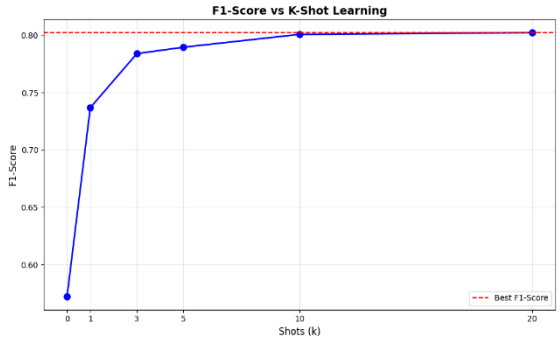
Berdasarkan Tabel 3, hasil eksperimen menunjukkan dinamika peningkatan kinerja yang jelas seiring dengan penambahan jumlah contoh. Pada skenario *zero-shot* (k=0), di mana model hanya mengandalkan pengetahuan *pre-training* tanpa panduan contoh dari *Shot Pool*, kinerja yang dihasilkan menjadi dasar perbandingan dengan F1-Score sebesar 0.5722.

Lompatan kinerja paling signifikan terjadi pada skenario *one-shot* (k=1), di mana penambahan satu contoh saja berhasil meningkatkan F1-Score secara drastis menjadi 0.7368, sebuah peningkatan sebesar 28.8%. Hal ini mengindikasikan bahwa satu contoh berkualitas tinggi sudah cukup bagi model untuk memahami format output JSON yang diharapkan dan konteks dasar dari tugas ekstraksi.

Peningkatan performa berlanjut pada skenario 3-shot, yang mencapai F1-Score 0.7839, dan terus menanjak pada 5-shot dengan skor 0.7895. Pada tahap ini, penambahan contoh menunjukkan bahwa model mulai mempelajari nuansa yang lebih halus dan menangani kasus-kasus yang lebih bervariasi. Skenario 10-shot berhasil menembus ambang batas 0.80 dengan F1-Score 0.8007, namun laju peningkatannya mulai melambat. Akhirnya, kinerja puncak dalam eksperimen ini tercapai pada skenario 20-shot, yang menghasilkan F1-Score tertinggi sebesar 0.8023, meskipun dengan keuntungan marginal yang paling kecil dibandingkan skenario sebelumnya.

Selain peningkatan F1-Score, sebuah pola konsisten yang terungkap dari Tabel 3 adalah nilai Presisi yang selalu lebih tinggi daripada nilai Recall di seluruh skenario *few-shot* (k≥1). Hal ini mengindikasikan bahwa model memiliki kecenderungan untuk bersikap 'hati-hati' atau konservatif. Ketika model memprediksi sebuah entitas, prediksinya memiliki probabilitas tinggi untuk benar, sehingga jumlah *False Positives* (entitas yang salah diekstrak) relatif rendah. Di sisi lain, nilai Recall yang sedikit lebih rendah menunjukkan bahwa sikap 'hati-hati' ini berakibat pada beberapa entitas valid yang terlewatkan atau gagal dideteksi, yang berarti jumlah *False Negatives* lebih tinggi. Secara praktis, dapat diinterpretasikan bahwa model ini lebih memilih untuk tidak memberikan jawaban jika tidak yakin, daripada memberikan jawaban yang berisiko salah.

Secara keseluruhan, data pada Tabel 3 secara kuantitatif membuktikan bahwa terdapat korelasi positif yang kuat antara jumlah contoh yang diberikan dalam *prompt* dan akurasi ekstraksi entitas. Dimulai dari F1-Score 0.5722 pada kondisi tanpa contoh, performa model meningkat secara progresif hingga mencapai puncaknya di 0.8023 pada 20-shot, merepresentasikan peningkatan kinerja total yang substansial sebesar 39.4%. Pola peningkatan yang tidak linear ini, dengan keuntungan terbesar di awal dan keuntungan marginal di akhir, mengindikasikan adanya titik jenuh.



Gambar 2. Kurva Kinerja F1-Score terhadap Shots (k)

Grafik pada Gambar 2 secara visual mengonfirmasi dan memperjelas temuan dari data tabel. Kurva ini dengan jelas mengilustrasikan fenomena *point of diminishing returns* (titik jenuh keuntungan) dalam penerapan *few-shot* learning pada tugas ini. Perilaku kurva dapat dianalisis dalam tiga fase utama:

1. Fase Pembelajaran Cepat (k=0 hingga k=3): Kurva menunjukkan tanjakan yang sangat curam pada fase ini. Peningkatan F1-Score dari 0-shot ke 1-shot, dan dari 1-shot ke 3-shot adalah yang paling dramatis. Ini mengindikasikan bahwa model dengan cepat "mempelajari" tugas yang diberikan seperti format output JSON

yang diinginkan dan pola-pola dasar ekstraksi hanya dengan beberapa contoh awal. Setiap contoh pada fase ini memberikan nilai informasi yang sangat tinggi bagi model.

2. Fase Pematangan (k=3 hingga k=10): Pada fase ini, kemiringan kurva mulai melandai. Peningkatan kinerja dari 3-shot ke 5-shot, dan dari 5-shot ke 10-shot masih terjadi secara konsisten, namun dengan laju yang lebih lambat. Hal ini menunjukkan bahwa model telah menguasai sebagian besar kasus umum. Contoh-contoh tambahan pada tahap ini lebih berfungsi untuk menyempurnakan pemahaman pada kasus-kasus yang lebih ambigu atau bervariasi (*edge cases*), sehingga kontribusi peningkatannya tidak lagi sebesar di awal.
3. Fase Titik Jenuh / *Plateau* (k=10 hingga k=20): Garis kurva menjadi hampir datar antara 10-shot dan 20-shot. Peningkatan F1-Score pada rentang ini sangat marginal (hanya sebesar 0.0016). Ini adalah bukti kuat bahwa model telah mencapai titik jenuh performa dengan metode *prompt*ing ini. Penambahan 10 contoh lagi hampir tidak memberikan keuntungan akurasi yang berarti, namun secara signifikan meningkatkan panjang *prompt*, biaya token, dan potensi latensi.

Kurva kinerja ini membawa implikasi praktis yang dapat digambarkan sebagai sebuah peta panduan untuk implementasi di dunia nyata. Temuan ini menggarisbawahi adanya trade-off yang fundamental antara performa yang ingin dicapai dengan efisiensi sumber daya. Efisiensi di sini tidak hanya terbatas pada biaya pemanggilan API, tetapi juga mencakup latensi respons di mana *prompt* yang lebih panjang membutuhkan waktu inferensi lebih lama dan usaha pengembangan yang diperlukan untuk mengkurasi contoh atau *Shot Pool*. Berdasarkan data, dapat ditarik beberapa rekomendasi strategis. Untuk aplikasi yang memprioritaskan "*return on investment*" yang cepat, di mana kinerja yang "cukup baik" sudah memadai, skenario 5-shot dapat dianggap sebagai titik optimal. Pada titik ini, model telah mencapai sebagian besar dari potensi peningkatannya dengan jumlah contoh yang relatif sedikit.

Sementara itu, untuk aplikasi yang menuntut akurasi setinggi mungkin misalnya dalam sistem analisis data kritis 10-shot merupakan titik optimal yang paling rasional. Hasil eksperimen dengan jelas menunjukkan bahwa peningkatan performa setelah titik ini menjadi sangat marginal dan kemungkinan besar tidak sepadan dengan kenaikan biaya, latensi, serta kompleksitas *prompt* yang berlipat ganda. Skenario 20-shot, meskipun secara teknis menghasilkan skor tertinggi, hanya akan relevan pada kasus di mana bahkan peningkatan akurasi sekecil 0.2% dianggap krusial dan segala bentuk biaya sumber daya dapat diabaikan.

Pemahaman mengenai *trade-off* ini juga harus mempertimbangkan karakteristik perilaku model. Temuan konsisten bahwa nilai Presisi lebih tinggi daripada Recall memberikan wawasan bahwa model lebih dapat diandalkan untuk tidak menghasilkan data yang salah (meminimalisir False Positives), meskipun dengan risiko melewatkan beberapa informasi yang valid (False Negatives). Dalam banyak aplikasi bisnis, seperti populasi basis data otomatis atau analisis pasar, integritas data memastikan bahwa data yang masuk itu benar seringkali lebih diutamakan. Oleh karena itu, perilaku "konservatif" dari model ini dapat dianggap sebagai sebuah keuntungan, karena secara inheren mengurangi risiko kontaminasi data oleh informasi yang diekstrak secara keliru.

Dengan demikian, hasil ini secara komprehensif memetakan "seberapa baik" model dapat bekerja, dan juga "bagaimana" ia bekerja dalam tugas ekstraksi entitas ini. Implikasi praktis dari temuan ini mengarah pada perumusan strategi *prompting* bertingkat yang dapat disesuaikan dengan tujuan spesifik aplikasi. Untuk implementasi yang memprioritaskan efisiensi sumber daya, skenario 5-shot dapat diidentifikasi sebagai titik keseimbangan optimal antara usaha dan hasil. Sementara itu, untuk aplikasi yang menuntut akurasi maksimal, skenario 10-shot terbukti menjadi batas atas yang rasional, karena peningkatan kinerja setelah titik tersebut bersifat marginal dan kemungkinan besar tidak sepadan dengan kenaikan biaya komputasi dan kompleksitas.

Di samping itu, terlihat favoritisme model terhadap presisi atas recall (*precision over recall*) memberikan dimensi pertimbangan strategis. Karakteristik ini mengindikasikan bahwa arsitektur model secara inheren lebih andal dalam memitigasi kesalahan positif (*False Positives*), sebuah sifat yang dapat menjadi keuntungan desain dalam alur kerja otomatis yang menuntut integritas data tinggi. Oleh karena itu, sintesis antara analisis efisiensi kuantitatif dan pemahaman perilaku kualitatif ini menjadi krusial dalam mendasari perumusan strategi implementasi teknologi LLM yang optimal dan berbasis data untuk tugas-tugas serupa di lingkungan nyata.

4. Kesimpulan

Penelitian ini berhasil menunjukkan bahwa penerapan teknik *few-shot learning*, khususnya variasi jumlah contoh (k) dalam *prompt*, memiliki pengaruh yang signifikan terhadap akurasi *Large Language Model* (LLM) pada tugas ekstraksi entitas dari teks tidak terstruktur. Melalui serangkaian skenario pengujian yang sistematis (k=0 hingga k=20), ditemukan bahwa terdapat korelasi positif antara jumlah contoh yang diberikan dan kinerja model, yang diukur dengan F1-Score. Penambahan contoh terbukti secara drastis meningkatkan akurasi dari level dasar *zero-shot*, namun peningkatan ini menunjukkan adanya titik jenuh keuntungan (*point of diminishing returns*). Skenario 5-shot teridentifikasi sebagai titik optimal yang menawarkan keseimbangan terbaik antara pencapaian akurasi

maksimal dan efisiensi sumber daya, karena penambahan contoh setelah titik tersebut hanya menghasilkan keuntungan yang sangat marginal.

Selain itu, nilai Presisi secara konsisten lebih tinggi daripada nilai Recall mengindikasikan bahwa model memiliki kecenderungan bawaan untuk bersikap "konservatif", yaitu lebih memilih untuk tidak mengekstrak entitas jika tidak yakin, daripada menghasilkan data yang berisiko salah. Temuan ganda ini, baik mengenai efisiensi kuantitatif maupun perilaku kualitatif menegaskan bahwa pendekatan "lebih banyak contoh selalu lebih baik" tidak berlaku secara absolut. Sebaliknya, diperlukan strategi *prompt engineering* yang cermat dan berbasis data untuk mencapai hasil yang optimal. Oleh karena itu, penelitian ini tidak hanya memberikan wawasan teoritis, tetapi juga rekomendasi praktis bagi para pengembang yang memanfaatkan teknologi LLM untuk tugas-tugas serupa di lingkungan nyata.

Meskipun demikian, penelitian ini memiliki beberapa keterbatasan yang membuka peluang untuk eksplorasi di masa depan. Validasi temuan ini perlu dilakukan pada domain yang lebih beragam (misalnya, teks hukum atau medis) dan dengan dataset yang lebih besar untuk menguji generalisasi pola kinerja. Selain itu, eksplorasi menggunakan model LLM yang berbeda terutama yang memiliki jumlah parameter dan *context length* yang bervariasi dapat menguji apakah karakteristik kinerja ini bersifat universal. Terakhir, penerapan teknik *prompt engineering* lainnya, seperti *Chain-of-Thought* (CoT), dapat diinvestigasi untuk melihat potensi peningkatan akurasi lebih lanjut di luar titik jenuh yang teridentifikasi dalam penelitian ini.

Daftar Rujukan

[1] Fareiz Aulia Firman, Dr. Vip Paramarta Drs., MM, Rocky Fransiskus Budiman, Yuliani Salewe, and Karlis Karlis. 2023. Fungsi SDM Sebagai Pemain Strategik Manajemen Modal Insani dan Manajemen Talenta. *Journal of Creative Student Research*. 1(3): 289–303.
[2] T. Ghorpade. 2024. Online Job Portal. *Gurukul International Multidisciplinary Research Journal*.
[3] M. Pejic-Bach, T. Bertoncel, M. Meško, and Ž. Krstić. 2020. Text mining of industry 4.0 job advertisements. *Int J Inf Manage*. 50: 416–431.
[4] K. Fabian, E. Taylor-Smith, S. Smith, and A. Bratton. 2023. Signalling new opportunities? An analysis of UK job adverts for degree apprenticeships. *Higher Education, Skills and Work-Based Learning*. 13(2): 299–314.
[5] N. Kamaleson, D. Chu, and F. E. B. Otero. 2021. Automatic Information Extraction from Electronic Documents Using Machine Learning. dalam: M. Bramer and R. Ellis (Eds). *Artificial Intelligence XXXVIII*. Cham: Springer International Publishing: 183–194.
[6] H. Bao, L. Dong, W. Wang, N. Yang, S. Piao, and F. Wei. 2024. Fine-tuning pretrained transformer encoders for sequence-to-sequence learning. *International Journal of Machine Learning and Cybernetics*. 15(5): 1711–1728.
[7] J.-C. Klie, R. E. de Castilho, and I. Gurevych. 2023. Analyzing Dataset Annotation Quality Management in the Wild. *Computational Linguistics*. 50: 817–866. Diunduh di https://api.semanticscholar.org/CorpusID:259937704 tanggal 25 Juni 2025.
[8] J. Lin, X. Li, and G. Pekhimenko. 2020. Multi-node Bert-pretraining: Cost-efficient Approach. *CoRR*. abs/2008.00177. Diunduh di https://arxiv.org/abs/2008.00177 tanggal 25 Juni 2025.
[9] A. Rula and J. D'Souza. 2023. *Procedural Text Mining with Large Language Models*. Prosiding The 12th Knowledge Capture Conference 2023. New York, NY, USA.
[10] J. Cabessa, H. Hernault, and U. Mushtaq. 2024. *In-Context Learning and Fine-Tuning GPT for Argument Mining*. Diunduh di http://arxiv.org/abs/2406.06699 tanggal 25 Juni 2025.
[11] Q. Dong et al. 2024. A Survey on In-context Learning. dalam: Y. Al-Onaizan, M. Bansal, and Y.-N. Chen (Eds). *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Miami, Florida, USA: Association for Computational Linguistics: 1107–1128.
[12] J. Seo et al. 2022. Plain Template Insertion: Korean-Prompt-Based Engineering for Few-Shot Learners. *IEEE Access*. 10: 107587–107597.
[13] C. Pornprasit and C. Tantithamthavorn. 2024. Fine-tuning and prompt engineering for large language models-based code review automation. *Inf Softw Technol*. 175: 107523.
[14] J. Li, A. Sun, J. Han, and C. Li. 2023. *A Survey on Deep Learning for Named Entity Recognition : Extended Abstract*. Prosiding 2023 IEEE 39th International Conference on Data Engineering (ICDE).
[15] A. V. Patil, H. Dand, and S. Kadam. 2024. Identifying specific details from text to populate databases and generate summaries using Named Entity Recognition. *INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)*. 8(5): 1–6.
[16] M. Shao, A. Basit, R. Karri, and M. Shafique. 2024. Survey of Different Large Language Model Architectures: Trends, Benchmarks, and Challenges. *IEEE Access*. 12: 188664–188706.
[17] C. Kauf et al. 2023. Event Knowledge in Large Language Models: The Gap Between the Impossible and the Unlikely. *Cogn Sci*. 47.
[18] A. Vaswani et al. 2017. *Attention Is All You Need*. Diunduh di http://arxiv.org/abs/1706.03762 tanggal 25 Juni 2025.
[19] Google T. 2025. *Gemini: A Family of Highly Capable Multimodal Models*. Diunduh di https://arxiv.org/abs/2312.11805 tanggal 25 Juni 2025.
[20] Google T. 2024. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. Diunduh di https://arxiv.org/pdf/2403.05530 tanggal 25 Juni 2025.
[21] L. Wang et al. 2024. Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs. *NPJ Digit Med*. 7(1): 41.
[22] J. D. Rathod. 2024. Systematic Study of Prompt Engineering. *Int J Res Appl Sci Eng Technol*. 12(6): 597–613.
[23] S. Sivarajkumar, M. Kelley, A. Samolyk-Mazzanti, S. Visweswaran, and Y. Wang. 2024. An Empirical Evaluation of Prompting Strategies for Large Language Models in Zero-Shot Clinical Natural Language Processing: Algorithm Development and Validation Study. *JMIR Med Inform*. 12.
[24] G. Shan. 2022. Monte Carlo cross-validation for a study with binary outcome and limited sample size. *BMC Med Inform Decis Mak*. 22(1).

Submitted : 05-08-2025 | Reviewed : 16-08-2025 | Accepted : 31-08-2025