**Decomposed Prompting (DecomP)**

1. **Penjelasan Dataset yang Digunakan**
2. **Penjelasan Library yang Digunakan**

Berikut adalah daftar library yang digunakan dalam kode program, beserta penjelasan fungsi/kegunaannya masing-masing:

1. **os** :merupakanlibrary bawaan Python untuk berinteraksi dengan sistem operasi. Kegunaan secara lebih spesifik pada kode program yaitu, digunakan untuk mengambil nilai dari environment variable dengan os.getenv("GEMINI\_API\_KEY"). Membantu menjaga keamanan API Key dengan tidak menuliskannya secara langsung dalam kode.
2. **google.generativeai** :merupakanlibrary resmi dari Google untuk menggunakan model Generative AI (termasuk Gemini). Kegunaan dari library tersebut pada kode program yaitu, untuk mengatur API Key untuk autentikasi dengan menggunakan genai.configure(api\_key=API\_KEY). Selain itu juga digunakan untuk menginisialisasi model Gemini versi 2.0 Flash dan melakukan pengiriman prompt ke model dan menerima hasil teks yang dihasilkan dengan menggunakan model.generate\_content(prompt, generation\_config=...).
3. **python-dotenv** :merupakan library untuk membaca file .env dan memuat isinya ke dalam environment variable Python. Kegunaan pada kode program yaitu, untuk memuat file .env sehingga os.getenv("GEMINI\_API\_KEY") bisa mengakses API key yang disimpan secara aman.
4. **json** :merupakanlibrary bawaan Python untuk bekerja dengan data JSON (JavaScript Object Notation). Kegunaan dari library tersebut pada kode program yaitu, untuk melakukan pembacaan(load) dan penyimpanan (dump) file dalam format JSON. Perubahan menjadi format JSON sangat membantu, karena JSON adalah format populer untuk menyimpan data terstruktur, sangat umum digunakan dalam data exchange dan konfigurasi. Terutama ketika ingin melakukan penyimpanan data hasil generate dan juga pemrosesan dataset.
5. **argparse** :adalah library bawaan Python yang digunakan untuk menangani argument command-line. Library ini memudahkan pengguna untuk menjalankan program Python dengan parameter tertentu dari terminal atau script. Library tersebut digunakan pada file main.py pada function parse\_args(), untuk mendefinisikan dan membaca argumen yang bisa diberikan saat menjalankan main.py lewat terminal, dengan memberikan tambahan command --<ARG\_NAME> <ARG\_VAL>, dan juga bisa setup default value untuk argument.
6. **re** :library ini digunakan untuk memproses teks dengan pola pencarian (regex). Library ini digunakan untuk memproses hasil output dari prompt yang telah diberikan kepada LLM melalui API Gemini, misalnya untuk mendapatkan jawaban akhir atau komponen tertentu dari hasil prompt yang dihasilkan oleh LLM. Selain itu juga bisa digunakan untuk membersihkan sebuah string dari character ataupun string tertentu.
7. **Time** :library ini digunakan untuk mengatur waktu, delay, dan pengukuran waktu dalam detik. Kegunaan dari library ini pada kode program adalah untuk memberikan delay proses, misalnya ketika diberikan time.sleep(60) maka proses akan di delay selama 1 menit. Hal ini berguna untuk menghindari rate limit dari API Gemini.
8. **ast** : library Abstract Syntax Trees ini digunakan untuk mengevaluasi ekspresi Python dengan aman. Kegunaan dari library tersebut pada kode program adalah untuk mengubah string representasi list (hasil dari model) menjadi list Python asli secara aman, hanya mendukung literal seperti list, dict, str, int, dan lainnya. Contoh yaitu ast.literal\_eval(output[-1]).
9. **Penjelasan Kode Program**

Penjelasan berikut membahas setiap bagian dari program yang telah dibuat, mulai dari function, cara kerja, hingga peran masing-masing komponen yang mereferensi ke dokumentasi kode yang telah dilakukan :

1. **utils/gemini\_client.py** : File ini berfungsi sebagai klien untuk berkomunikasi dengan model Gemini 2.0 Flash dari Google melalui API. Di dalamnya, terdapat konfigurasi API key, inisialisasi model, dan satu fungsi utama (generate\_answer) yang digunakan untuk mengirim prompt dan menerima jawaban dari model.
2. **Line 3 – 5:** Melakukan import library yang akan digunakan pada file tersebut, misalnya os dan juga dotenv dari python-dotenv digunakan untuk mendapatkan value yang terdapat pada file .env, seperti API KEY dari GeminiAPI. Selain itu juga terdapat genai library yang berasal dari google.generativeai, library ini digunakan untuk mengakses layanan GenAI seperti Gemini.
3. **Line 7 - 13:** Memanggil fungsi untuk memuat variabel dari file .env ke dalam environment Python, termasuk GEMINI\_API\_KEY. Jika tidak ditemukan pada program akan berhenti dengan pesan error yang jelas.
4. **Line 14 – 16:** Mengatur API key untuk mengautentikasi permintaan ke layanan Gemini, dan membuat objek model gemini-2.0-flash yang siap digunakan untuk menghasilkan konten.
5. **Function generate\_answer (Line 18 - 26) :** Function tersebut akan menerima teks prompt sebagai input, setelah itu akan mengirim prompt ke model Gemini dan mengatur **temperature=0.7** (untuk hasil yang seimbang antara kreativitas dan konsistensi). Hasil output dari model LLM tersebut juga akan diberikan sebagai return value. Tetapi jika terdapat error maka fungsi akan mengembalikan string error dengan awalan [ERROR].
6. **utils/io\_utils.py** : File ini berisi function utilitas untuk membaca dan menulis file JSON. Format JSON sering digunakan untuk menyimpan data konfigurasi atau hasil proses dalam bentuk yang terstruktur. Modul ini membantu menyederhanakan proses input-output file JSON dalam program.
7. **Line 3 :** Melakukan import untuk library yang diperlukan yaitu JSON, library tersebut akan digunakan untuk pemrosesan JSON pada masing masing fungsi.
8. **Function load\_json (Line 5 - 7) :** Tujuannya adalah untuk membaca file JSON dari path tertentu dan mengembalikan isinya dalam bentuk Python object (dict atau list). with.open akan digunakan untuk membuka file dengan aman dan otomatis tertutup setelah selesai. “r” berguna untuk mode baca (read). encoding="utf-8" akan Memastikan karakter non-ASCII (misal huruf beraksen) bisa dibaca dengan benar. json.load(f) akan mengubah isi file JSON menjadi object python. Fungsi ini biasanya digunakan untuk load dataset yang akan digunakan.
9. **Function save\_json (Line 9 - 11) :** Tujuannya adalah untuk menyimpan data Python (biasanya dict atau list) ke dalam file JSON. "w" digunakan untuk mode tulis (write), dan akan menimpa file jika sudah ada. indent=2 berguna untuk memberikan indentasi 2 spasi agar file mudah dibaca. ensure\_ascii=False digunakan memastikan karakter non-ASCII tidak dikonversi menjadi \uXXXX (misal, tetap menulis "é" bukan "\\u00e9"). Function ini biasanya digunakan untuk menyimpan hasil output dari pemrosesan dataset ke dalam sebuah file json.
10. **main.py** : File main.py berfungsi sebagai entry point utama untuk menjalankan program pemrosesan pertanyaan menggunakan API Gemini. Program ini menerima argumen dari command line, memilih jenis task (seperti commaqa, reverse, atau lettercat), lalu menjalankan controller yang sesuai dengan dataset tertentu untuk menghasilkan dan memproses jawaban dari model Gemini. Output hasil pemrosesan kemudian disimpan ke file JSON.
11. **Line 3 – 9:** import file ataupun library yang akan digunakan pada file tersebut, file yang di import seperti function yang terdapat pada folder utils seperti untuk load dan save json, dan juga gemini client untuk generate answer menggunakan GeminiAPI dengan model yang telah ditentukan. Library lainnya seperti re, os, time akan digunakan untuk pengolahan teks, sistem, dan penundaan waktu. Selain itu argparse akan digunakan untuk membaca argumen dari command line.
12. **Function parse\_args (Line 11 - 15) :** digunakan untuk mengatur dua argumen yang bisa diberikan saat menjalankan program. Misalnya –output sebagai lokasi file untuk menyimpan hasil (default generate\_res/dummy.json), dan –task untuk memilih jenis task yang ingin dijalankan (commaqa, reverse, atau lettercat).
13. **Function get\_controller (Line 17-28) :** function ini akan menerima task yang berasal dari argumen –task pada awal menjalankan program menggunakan command lain. setelah itu akan memberikan controller yang sesuai dengan task yang dipilih. Jika task tidak terdaftar maka akan memberikan error dengan pesan sesuai.
14. **Function main (Line 30 - 40) :** Sebagai function utama yang akan dipanggil ketika program main.py dieksekusi. args = parse\_args() digunakan untuk membaca argumen dari command line. Setelah itu controller = get\_controller(args.task) akan digunakan untuk mendapatkan controller sesuai dengan task. Setelah itu controller.solve() akan digunakan untuk memproses dataset dan menghasil output yang akan disimpan pada file output.
15. **Line 42 – 43 :** Baris ini memastikan bahwa main() hanya dijalankan ketika file main.py dieksekusi langsung, bukan saat diimpor sebagai modul.
16. **reverse\_controller.py** : File reverse\_controller.py berisi class ReverseController yang digunakan untuk memproses dataset reverse sequence dataset dengan pendekatan DecomP. Diawali dengan pengambilan dataset, sampai pada pemrosesan dataset yang dilakukan dengan bantuan template prompt dan model Gemini API. Terdapat mekanisme untuk menyimpan hasil dari setiap langkah dan digunakan untuk langkah berikutnya dalam mendapatkan output keseluruhan dari sebuah data / record pada dataset.
17. **Line 3 – 8** : import file ataupun library yang akan digunakan pada file tersebut, seperti utility untuk json, template prompting yang akan digunakan, dan juga gemini client untuk generate hasil output dari prompt pada LLM. Selain itu juga terdapat re, os, dan time untuk memproses pengolahan teks, dan penundaan waktu agar terhindar dari rate limit API.
18. **Line (9 - 107) :** merupakan isi dari class ReverseController. Pada point selanjutnya akan dijelaskan mengenai kegunaan dari masing masing segmen kode dari class ini.
19. **Function \_\_init\_\_ (Line 10 – 11 ) :** digunakan untuk memuat dataset pada dataset/reverse.json.
20. **Function print\_questions (Line 13 – 16) :** digunakan untuk menampilkan semua pertanyaan yang ada dalam dataset (untuk bantu/cek data).
21. **Function replace\_references (Line 18 - 27) :** digunakan untuk mengganti referensi ke jawaban sebelumnya dalam string seperti #1 dengan nilai sebenarnya dari hasil sebelumnya (result\_array) yang sesuai dengan index yang diminta.
22. **Function extract\_qs (Line 29 - 31) :** Mengambil string di dalam tanda kutip dari kalimat tertentu seperti: Sequence is "1. Ab 2. Bc" → akan mengambil "1. Ab 2. Bc".
23. **Function get\_generated\_answer (Line 33 – 38) :** Membersihkan hasil output dari GeminiAPI, dan mengambil bagian setelah “Answer: ”, jika tidak ditemukan maka langsung di kembalikan.
24. **Function chain\_processing (Line 40 – 78) :** digunakan untuk memproses sebuah data atau pertanyaan berdasarkan langkah-langkah yang diberikan kepada fungsi tersebut. Setiap langkah yang telah ditentukan akan dijalankan satu per satu, setiap langkah yang dijalankan pasti akan mereferensi ke sebuah subtask. Pada kasus dataset reverse sequence, terdapat **4 subtask** yang dapat digunakan yaitu,
25. **extract,** untuk mendapatkan kata yang ingin digunakan untuk reverse sequence.
26. **remove\_numbers,** untuk menghapus nomer urut dari setiap kata.
27. **reverse,** digunakan untuk membalikan kata tertentu
28. **join,** untuk menggabungkan beberapa hasil sebelumnya.

Dari keempat subtask tersebut hanya subtask untuk extract yang menggunakan function yang ditulis pada kode program, sedangkan 3 subtask lainnya memanfaatkan template prompt yang telah disediakan untuk melakukan generate hasil pemrosesan, akhir dari langkah penyelesaian pasti akan diakhiri dengan “[EOQ]” yang berarti seluruh proses telah selesai. Hasil dari setiap langkah akan disimpan dan dapat dirujuk pada langkah berikutnya menggunakan pattern “#n” yang berarti mendapatkan jawaban ke-n untuk langkah tertentu. Hasil dari seluruh langkah akan diberikan menjadi return value dari fungsi tersebut.

1. **Function generate\_chain (Line 80 – 87) :** function ini digunakan untuk mendapatkan langkah-langkah hasil pemecahan masalah menggunakan prompt template decomp\_chain yang telah dibuat pada file template untuk reverse dataset.
2. **Function solve (Line 89 – 107) :** function ini digunakan untuk memproses setiap record yang terdapat pada dataset, setelah itu akan melakukan perbandingan / pengecekan apakah jawaban yang diberikan tepat atau tidak. Data seperti pertanyaan yang diberikan, hasil pemecahan masalah, jawaban dan juga hasil validasi jawaban akan dicatat pada file output yang telah ditentukan. Terdapat juga sleep 1 menit untuk menghindari rate limit API. Untuk memproses sebuah data / record akan diawali dengan generate chain atau generate langkah penyelesaian, setelah itu barulah dilakukan pemrosesan secara terurut sampai menghasilkan sebuah output.
3. **lettercat\_controller.py** : File lettercat\_controller.py berisi class LatterCatController yang digunakan untuk memproses dataset letter concat dataset dengan pendekatan DecomP. Diawali dengan pengambilan dataset, sampai pada pemrosesan dataset yang dilakukan dengan bantuan template prompt dan model Gemini API. Terdapat mekanisme untuk menyimpan hasil dari setiap langkah dan digunakan untuk langkah berikutnya dalam mendapatkan output keseluruhan dari sebuah data / record pada dataset.
4. **Line 3 – 8** : import file ataupun library yang akan digunakan pada file tersebut, seperti utility untuk json, template prompting yang akan digunakan, dan juga gemini client untuk generate hasil output dari prompt pada LLM. Selain itu juga terdapat re, os, dan time untuk memproses pengolahan teks, dan penundaan waktu agar terhindar dari rate limit API**.**
5. **Line (9 - 126) :** merupakan isi dari class LatterCatController. Pada point selanjutnya akan dijelaskan mengenai kegunaan dari masing masing segmen kode dari class ini.
6. **Function print\_questions (Line 13 – 16) :** digunakan untuk menampilkan semua pertanyaan yang ada dalam dataset (untuk bantu/cek data).
7. **Function replace\_references (Line 18 - 27) :** digunakan untuk mengganti referensi ke jawaban sebelumnya dalam string seperti #1 dengan nilai sebenarnya dari hasil sebelumnya (result\_array) yang sesuai dengan index yang diminta.
8. **Function get\_references (Line 29 – 31) :** digunakan untuk mendapatkan seluruh referensi index jawaban sebelumnya yang dibutuhkan pada sebuah langkah pemrosesan, format index yang dicari seperti “#n” (index ke-n adalah yang dicari).
9. **Function get\_generated\_answer (Line 33 – 38) :** Membersihkan hasil output dari GeminiAPI, dan mengambil bagian setelah “Answer: ”, jika tidak ditemukan maka langsung di kembalikan.
10. **Function chain\_processing (Line 40 – 97) :** digunakan untuk memproses sebuah data atau pertanyaan berdasarkan langkah-langkah yang diberikan kepada fungsi tersebut. Setiap langkah yang telah ditentukan akan dijalankan satu per satu, setiap langkah yang dijalankan pasti akan mereferensi ke sebuah subtask. Pada kasus dataset letter concat, terdapat **4 subtask** yang dapat digunakan yaitu,
11. **split,** yangdapat digunakan untuk memisahkan kata pada sebuah kalimat, dan juga dapat untuk memisahkan karakter dari sebuah kata. Selain itu juga bisa memberikan informasi index dari kata atau karakter yang dipisahkan tergantung dari permintaan.
12. **str\_position,** merupakan sebuah subtask yang dapat dipecah menjadi beberapa subtask lagi dengan tujuan akhir untuk mendapatkan posisi kata pada kalimat ataupun karakter pada kata tertentu. Subtask yang digunakan pada str\_position adalah split, dan arr\_position. Biasanya subtask str\_position ini akan dilakukan untuk setiap kata yang dihasilkan oleh split.
13. **arr\_position,** untuk mendapatkan kata pada index tertentu atau karakter pada index tertentu yang telah diproses oleh subtask split. Subtask arr\_position ini akan digunakan pada langkah akhir untuk menyelesaiakn subtask str\_position.
14. **merge,** akan digunakan untuk menggabungkan jawaban yang telah diberikan pada langkah sebelumnya, misalnya hasil sebelumnya "['n', 'm', 't', 'n', 'n']", maka hasil merge akan menjadi "n;m;t;n;n" atau dapat juga menjadi "n m t n n" tergantung dari permintaan soal pada dataset.

Dari 4 subtask tersebut semuanya menggunakan bantuan LLM dan template prompt tertentu untuk mendapatkan hasil, hanya saja pada str\_position terdapat proses lebih kompleks, karena terdapat proses docomposition lagi untuk bisa mendapatkan hasil akhir dari subtask tersebut. Akhir dari langkah penyelesaian pasti akan diakhiri dengan “[EOQ]” yang berarti seluruh proses telah selesai. Hasil dari setiap langkah akan disimpan dan dapat dirujuk pada langkah berikutnya menggunakan pattern “#n” yang berarti mendapatkan jawaban ke-n untuk langkah tertentu. Hasil dari seluruh langkah akan diberikan menjadi return value dari fungsi tersebut.

1. **Function generate\_chain (Line 99 – 104) :** function ini digunakan untuk mendapatkan langkah-langkah hasil pemecahan masalah menggunakan prompt template decomp\_chain yang telah dibuat pada file template untuk lettercat dataset.
2. **Function solve (Line 106 – 126) :** function ini digunakan untuk memproses setiap record yang terdapat pada dataset, setelah itu akan melakukan perbandingan / pengecekan apakah jawaban yang diberikan tepat atau tidak. Data seperti pertanyaan yang diberikan, hasil pemecahan masalah, jawaban dan juga hasil validasi jawaban akan dicatat pada file output yang telah ditentukan. Terdapat juga sleep 1 menit untuk menghindari rate limit API. Untuk memproses sebuah data / record akan diawali dengan generate chain atau generate langkah penyelesaian, setelah itu barulah dilakukan pemrosesan secara terurut sampai menghasilkan sebuah output.
3. **commaqa\_controller.py** :
4. **prompts/reverse/template.py** : File template.py ini menyimpan template prompt khusus untuk task reverse sequence (digunakan oleh ReverseController), template yang terdapat pada file ini mencakup template untuk pembuatan urutan langkah-langkah yang akan dilakuan, selain itu template prompt yang akan digunakan untuk menyelesaikan subtask tertentu.
5. **Line 2 - 31** : merupakan decomp\_chain template untuk menghasilkan QS (Question Sequence) yang merupakan langkah-langkah penyelesaian, prompt ini digunakan di awal sebelum function solve pada controller dipanggil. Template ini menggunakan fewshot prompting.
6. **Line** **32 – 52** : merupakan join\_template yang menggunakan konsep fewshot prompting untuk melakukan penggabungan beberapa 2 kalimat (berisi daftar kata), menjadi satu kesatuan daftar kata.
7. **Line 54 – 68 :** merupakan remove\_numbers template yang menggunakan fewshot prompting untuk menghilangkan nomor urut pada beberapa kata. Contohnya, jika awalnya "4. bottle, 3. laptop, 2. glasses, 1. newspaper" maka hasil output LLM yang diharapkan jika menggunakan template prompt tersebut dan telah melewati pemrosesan adalah “bottle, laptop, glasses, newspaper”.
8. **Line 70 – 88 :** merupakan sebuah template yang menggunakan konsep CoT untuk melakukan reverse terhadap sebuah urutan kata. Contohnya, jika awalnya “laptop, photo, clip” maka hasil output LLM yang diharapkan jika menggunakan template prompt tersebut dan telah melewati pemrosesan adalah “clip, photo, laptop".
9. **prompts/lettercat/template.py** : File template.py ini menyimpan template prompt khusus untuk task letter concat (digunakan oleh LatterCatController), template yang terdapat pada file ini mencakup template untuk pembuatan urutan langkah-langkah yang akan dilakuan, selain itu template prompt yang akan digunakan untuk menyelesaikan subtask tertentu.
10. **Line 3 - 29** : merupakan template prompting untuk menghasilkan QS (Question Sequence) yang merupakan langkah-langkah penyelesaian, prompt ini digunakan di awal sebelum function solve pada controller dipanggil. Template ini menggunakan fewshot prompting.
11. **Line 31 – 51 :** merupakan template untuk melakukan split sebuah kata menjadi daftar karakter, atau kalimat menjadi daftar kata. Contohnya, jika terdapat “Alan” maka hasil yang diharapkan adalah “["A", "l", "a", "n"]” atau “[(A, 1), (l, 2), (a, 3), (n, 4)]” yang tergantung pada kebutuhan, jika pada kalimat juga akan sama, hanya saja pemisahan bukan dilakukan per karakter tetapi per kata.
12. **Line 53 – 73 :** merupakan template untuk melakukan merge antar kata atau karakter yang diberikan. Contohnya, jika terdapat “["Alan", "Mathison", "Turing"]” maka diharapkan menjadi “Alan Mathison Turing”, penggabungan tersebut dapat juga dilakukan pada karakter, dan juga penggabungan tersebut dapat dilakukan dengan menggabungkan antar kata atau karakter menggunakan spasi (“ ”) atau menggunakan character lain tergantung permintaan.
13. **Line 75 – 98 :** merupakan template yang digunakan untuk membuat langkah-langkah decomposition untuk menyelesaikan subtask str\_position. Sehingga hasil yang diharapkan dari penggunaan template prompt tersebut adalah daftar QS yang akan dijalankan menggunakan subtask lainnya (seperti split, dan arr\_position) untuk mendapatkan hasil final dari str\_position.
14. **Line 100 – 122 :** merupakan template untuk arr\_position, untuk mendapatkan karakter atau kata pada index tertentu. Contohnya, jika diberikan "[(M, 1), (a, 2), (t, 3), (h, 4), (i, 5), (s, 6), (o, 7), (n, 8)]" dan diminta nilai dari index ke 1, maka hasi yang diharapkan dengan menggunakan template prompt tersebut adalah “M”.
15. **prompts/commaQA/template.py** : File template.py ini menyimpan template prompt khusus untuk task long context question answering (digunakan oleh CommAQAController), template yang terdapat pada file ini mencakup template untuk pembuatan urutan langkah-langkah yang akan dilakuan, selain itu template prompt yang akan digunakan untuk menyelesaikan subtask tertentu.
16. **Line 2 – 45 :** merupakan template prompting untuk menghasilkan QS (Question Sequence) yang merupakan langkah-langkah penyelesaian, prompt ini digunakan di awal sebelum function solve pada controller dipanggil. Template ini menggunakan fewshot prompting.
17. **Line 47 – 82 :** merupakan template prompting untuk pos\_qa (Position QA), sehingga akan digunakan untuk mendapatkan informasi yang berkaitan dengan posisi, peran, atau keterlibatan seseorang dalam film. Contohnya, jika terdapat pertanyaan mengenai “For which movies was Teeplemole the producer?”, maka hasil yang diharapkan akan seperti ["<entitas\_1>", “<entitas\_n>”]. Template prompt tersebut juga disediakan supaya bisa memberikan context yang akan digunakan untuk menjawab pertanyaan tersebut.
18. **Line 84 – 119 :** merupakan template prompting untuk aw\_qa (Award QA), sehingga akan digunakan untuk mendapatkan informasi yang berkaitan dengan penghargaan. Contohnya, jika terdapat pertanyaan mengenai “Which awards were given to Zalate?”, maka hasil yang diharapkan akan seperti ["<entitas\_1>", .., “<entitas\_n>”]. Template prompt tersebut juga disediakan supaya bisa memberikan context yang akan digunakan untuk menjawab pertanyaan tersebut.
19. **Line 121 – 155 :** merupakan template prompting untuk simp\_qa (Simple QA) yang digunakan untuk menjawab pertanyaan sederhana, seperti “Siapa yang lahir pada tahun X”, atau “Siapa yang berasal dari negara X”, dan jawaban yang diberikan akan memiliki format [“<nama\_entitas\_1>”, .. , “<nama\_entitas\_N>”]. Template prompt tersebut juga disediakan supaya bisa memberikan context yang akan digunakan untuk menjawab pertanyaan tersebut.
20. **Mekanisme Perhitungan Performansi**

Perhitungan performansi dilakukan pada setiap dataset yang digunakan. Perhitungan tersebut dilakukan dengan menghitung persentase jawaban benar yang dihasilkan melalui proses yang telah dirancang dengan teknik Decomposed Prompting. Jawaban yang dihasilkan pada akhir pemrosesan sebuah record / data pada dataset akan secara langsung dibandingkan dengan jawaban yang benar (terdapat pada dataset) sebelum lanjut pada data / record selanjtunya. Berikut adalah hasil performansi dari penggunaan Decomposed Prompting dengan LLM Gemini 2.0 Flash :

1. **Pada Dataset Word Reversal** : Perhitungan score dilakukan dengan jumlah jawaban benar dibandingkan dengan jumlah total soal yang ada pada dataset.
2. Benar : 14
3. Salah : 1
4. Score : 93.33%
5. **Pada Dataset kth Letter Concatenation** : Perhitungan score dilakukan dengan jumlah jawaban benar dibandingkan dengan jumlah total soal yang ada pada dataset.
6. Benar : 18
7. Salah : 0
8. Score : 100%
9. **Pada Dataset Long Context Question Answering (CommaQA)** : Perhitungan score pada dataset ini berbeda dengan lainnya, karena jawaban yang benar lebih dari 1, sehingga perhitungan dilakukan dengan menghitung jumlah jumlah jawaban benar dibandinkan dengan jawaban yang tidak ditemukan dan jawaban yang berlebihan. Dengan demikian maka masing masing data / record pada dataset akan menggunakan perhitungan tersebut untuk mendapatkan score. Setelah itu akan dijumlahkan semua score tersebut.
10. Score 13.99
11. Max Score : 15 (Terdapat 15 soal, jika 1 soal benar secara total maka mendapatkan score 1.00)