Pencarian Suatu Effect dan Sinergi Pada Kartu-Kartu Yu-Gi-Oh! Menggunakan Regex dan Fuzzy Match

Nadhif Al Rozin - 13523076
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: nadhifalrozin@gmail.com , 13523076@std.stei.itb.ac.id

***Abstract*—Aplikasi ini digunakan untuk mencari suatu kata kunci dari kartu pada Yu-Gi-Oh. Fitur utama yang dikembangkan adalah penggunaan fuzzy matching, sistem dapat mencari berdasarkan nama, efek, jenis, tipe, atribut, deskripsi dari suatu kartu. Basis data yang digunakan berasal dari wiki web Yu-Gi-Oh, Penggunaan Regex digunakan untuk menampilkan dengan *highlight* untuk kata kata kunci yang ditemukan pada kartu.**

Keywords—Search, Fuzzy, Regex, Yu-Gi-Oh, Kartu

I. Pendahuluan

Para duelist yang baru memulai bermain terkadang mengalami kesulitan untuk memahami kartu kartu yang digunakan. Hal ini dikarenakan kartu pada Yu-Gi-Oh memiliki Tingkat kompleksitas tersendiri yang memiliki banyak sinergi antar kartu dan menyebabkan kebingungan dimana harus memulai. Oleh karena itu kemampuan untuk mencari di antara banyak kartu dengan mencari suatu kata kunci tertentu untuk mencari tahu efek suatu kartu berhubungan dengan kartu yang lain. Oleh karena itu pada makalah ini akan dibuat sebuah sistem pencarian yang menggunakan library fuzzy match di python untuk mencari kemunculan kata kunci pada kartu kartu dan penggunaan library regex dari python untuk memberi highlight ke kata kata tertentu pada kartu tersebut.

Dari hal ini dapat terlihat masalah masalah yang perlu diselesaikan pada sistem ini. Dimulai dengan bagaimana efek kartu dapat dicari secara efisien ? Lalu bagaimana cara kita mengenali hubungan antar 2 kartu berdasarkan suatu kata kunci ? Dan bagaimana kita menampilkan hasil tersebut kepada user. Dari masalah masalah ini diperkirakan dibutuhkan sistem dengan spesifikasi berikut: Sistem pencarian berbasis teks yang berasal dari nama kartu + attribute-attribute kartu lainnya + deskripsi dan menggunakan fuzzy match untuk mencarinya. Agar user dapat mengenali hubungan antar kartu, diperlukan pilihan untuk mencari multiple card yang memiliki kemunculan keyword tertentu. Selain itu diperlukan adanya highlight agar user dapat langsung mengetahui di bagian mana suatu kartu itu mempunyai kata kunci yang kita cari. Untuk menampilkan pada user, selain dari antarmuka yang sederhana bagi pengguna untuk memudahkan penggunaan. Kartu yang ditampilkan akan di urutkan dari nilai kecocokan tertinggi terlebih dahulu dan diberikan batas tertentu sebagai threshold untuk cut off.

II. Landasan Teori

A. Kartu Yu-Gi-Oh

Pada permainan Yu-Gi-Oh, pemain yakni kita para duelist akan bergantian memainkan kartu. Kartu ini terbagi dalam tiga jenis utama, yaitu monster, spell dan trap. Kartu monster berfungsi sebagai komponen serangan dan pertahanan dalam permainan, yang memiliki atribut seperti ATK (damage yang dihasilkan), DEF (damage yang dapat ditahan), Level, Jenis seperti Dragon, Warrior, Zombie, Spell Caster, Machine dan banyak lagi. Untuk Kartu Spell digunakan untuk memberikan efek strategis tertentu yang dapat digunakan untuk mendukung strategi, seperti meningkatkan ATK suatu monster, memanggil kartu dari deck, menghancurkan kartu lawan, membanish kartu lawan (mengirim kartu lawan ke graveyard). Untuk kartu bertipe Trap umumnya bersifat reaktif, kartu ini diaktifkan sebagai respon terhadap aksi lawan dan memberikan efek kejutan seperti membatalkan serangan atau menghapus kartu dari field lawan. Kartu trap biasa digunakan untuk mengantisipasi gerakan lawan pada turn lawan.

Selain klasifikasi 3 kelas utama (Monster, Spell dan Trap) kartu monster memiliki beberapa subtype, seperti Fusion, Synchro, Xyz, Link, Ritual dan Pendulum. Masing-masing subtype memiliki cara pemanggilan yang khusus dan membutuhkan kondisi dan kombinasi kartu kartu tertentu untuk bisa di summon. Misalnya, Monster fusion dipanggil dengan penggabungan dua atau lebih monster menggunakan kartu Spell Polymerization, Sedangkan utnuk monster subtype Synchro monster perlu kombinasi monster “Tuner” dan “Non-Tuner” dengan total level tertentu. Keberadaan subtype ini memperkaya strategi dan keragaman untuk memainkan berbagai macam deck dengan keunikan, nilai kuat dan lemahnya masing masing. Hal ini menuntut player untuk memahami sinergi antar kartu yang kita mainkan dan kartu yang lawan gunakan dengan lebih mendalam.

Untuk bagian efek kartu, struktur teks umumnya sangat beragam dan kompleks. Terdapat efek yang berantai (Chain), chain adalah efek yang dapat merespons efek lain dalam urutan

waktu (Chain 1, 2, 3 dan seterusnya) urutan ini melambangkan urutan kartu dimainkan. Tentu jika ada efek chain pasti ada efek non-chain, artinya efek ini tidak dapat difollow up dan terjadi secara instan, dalam segi strategi kartu ini adalah kartu tipe instan yang tidak dapat dibalas oleh lawan. Efek chain tidah hanya dapat dilanjutkan oleh pemain saat ini tapi juga bisa dilanjutkan oleh efek trap musuh. Efek juga dapat berupa one time use (sekali pakai) atau efek yang terus menerus. Umumnya tergantung pada tujuannya kartu tipe terus menerus punya berbagai nama, seperti floodgate, ini adalah sebutan pemain untuk akrtu yang menghambat strategi lawan, memblokir efek, summon atau jenis kartu tertentu Contohnya seperti Skill Drain, Vanity's Emptiness, Rivalry of Warlods dan banyak lagi. Selain itu Kartu yang memberi peningkatan ATK atau DEF atau efek tambahan lainnya ke seluruh field adalah kartu bertipe Field Buff atau Field Spell Contohnya seperti Forest, Necrovalley, Sky Striker AirSpace – Area Zero. Kartu tipe lain yang mirip dengan Field buff adalah Continous Support atau Support Card, kartu ini memberikan keuntungan yang berulang ulang selama ia berada dalam field dan tidak dihancurkan oleh musuh Contohnya seperti Valhalla, Hall of the Fallen atau trap seperti Call of the Haunted.

Dalam permainan kompetitif Yu-Gi-Oh!, tidak hanya efek kartu secara individual yang penting, namun juga bagaimana menghadapi deck dari musuh. Konsep sinergi ini menjadi inti dalam strategi permainan karena beberapa efek hanya optimal jika digunakan dalam urutan (Chain) atau kondisi tertentu, contohnya seperti efek pemanggilan khusus, pengaktifan efek pencarian kartu, melakukan kombo multi-layer. Selain itu juga perlu memperhatikan meta yang ada saat ini dan banlist yang ada. Hal ini membuat pemain harus terus menyesuaikan deck mereka karena itu pemahaman mendalam mengenai tujuan sebuah deck adalah hal terpenting dibandingkan memahami cara menggunakan deck.

Deskripsi efek sering kali menggunakan kata kunci khusus seperti "destroy", "target", "banish", "tribute", atau "summon", yang menjadi indikator penting dalam sistem pencarian otomatis berbasis teks. Dalam konteks ini, pengolahan data dan pencarian informasi, tiap tiap kartu yang memiliki metadata penting (nama kartu, jenis, tipe monster, atribut, level dan deskripsi efek). Data itu dikumpulkan melalui proses web scraping dari database Yu-Gi-Oh berikut: https://www.db.yugioh-card.com/yugiohdb/card_list.action?clm=1&wname=CardSearch

Data disimpan dalam format yang terstruktur seperti JSON. Metadata ini digunakan untuk keperluan pencocokan dan pencarian berdasarkan teks Panjang yang merupakan hasil penggabungan tiap metadata kartu. Ini memungkinkan filtering dan pencocokan berdasarkan berbagai aspek dari kartu. Dengan struktur yang baik, sistem pencarian dapat menyaring dan menemukan kartu secara efisien berdasarkan kata kunci atau efek yang dicari pengguna.

B. Fuzzy Matching

Fuzzy matching adalah Teknik pencarian yang berbasis kemiripan string, yang sangat digunakan ketika sistem harus menangani data teks yang tidak sempurna, seperti adanya kesalahan dalam mengetik (typo), variasi nama ataupun input yang tidak lengkap. Dalam konteks ini, metode ini memungkinkan sistem untuk tetap menemukan kartu yang relevan meskipun pengguna ada melakukan kesalahan dalam menginput. Salah satu algoritma paling umum dalam melakukan fuzzy matching adalah Levenshtein Distance, yang dilakukan dengan menghitung jumlah minimum operasi yang dibutuhkan untuk dapat merubah satu string menjadi string lainnya. Operasi ini dapat berupa, penyisipan, penghapusan dan penggantian karakter. Semakin kecil jarak maka semakin mirip dua string tersebut. Selain itu, algoritma Jaro-Winkler Distance juga sering digunakan karena lebih sensitive terhadap kesamaan awalan string, yang sering kali relevan dalam nama kartu. Misalnya, nama-nama kartu yang ada dalam satu archetype yang biasanya dimainkan dalam satu deck dan saling memiliki sinergi satu sama lain seperti kartu kartu dengan awalan "Sky Striker", sehingga algoritma ini memberikan keunggulan dalam mendeteksi kemiripan struktur nama.

Levenshtein Distance adalah sebuah metode untuk mengukur Tingkat kemiripan dari dua buah string, yang dilakukan dengan cara menghitung jumlah minimum operasi yang diperlukan untuk mengubah satu string ke string lain. Setiap operasi yang sudah disebutkan sebelumnya diberi bobot atau cost yang sebesar satu, dan hasil akhirnya adalah angka yang menunjukkan "jarak". Semakin kecil angka yang didapat maka semakin mirip kedua string tersebut. Sebagai contoh, jika kita mengubah lata kitten menjadi sitting, diperlukan tiga Langkah yaitu mengganti k menjadi s, e menjadi i dan menambahkan g di akhri kata. Sehinggal jaraknya adalah 3. Metode ini sangatlah populer digunakan di berbagai bidang utamanya untuk pencarian teks yang toleran terhadap kesalahan pengetikan. Levenhestein Distance bersifat simetris, artinya jarak dari string A dan B sama dengan B dan A dan hasilnya selalu 0 jika dua string tersebut sama/identic. Dalam praktiknya, algoritma ini digunakan dengan pendekatan dynamic programming, karena proses perhitungan yang melibatkan kombinasi submasalah yang tumpeng tindih. Keunggulan utam dari metode ini adalah karena ia memiliki kemampuan untuk mengenali adannya kemiripan meskipun ada kesalahan ketik yang ringan.

Dalam penggunaannya implementasi paling umum adalah dengan menggunakan pustaka FuzzyWuzzy, yang dibangun di atas Levenshtein Distance. Proses pencarian menggunakan Pustaka tersebut untuk mencari sejumlah entri teratas dengan limit yang diberi pengguna. Hasil tersebut akan difilter dengan threshold untuk memastikan hanya hasil dengan tingkat kemiripan yang memadai yang akan dimasukkan ke hasil akhir.

C. Regex

Regular Expression atau Regex adalah suatu pola atau aturan yang digunakan untuk mencocokkan rangkaian karakter dalam string. Dalam pemrosesan teks, regex sangat berguna untuk melakukan pencarian, validasi, ekstraksi berbasis suatu pola tertentu. Regex juga mampu menangani pola-pola kompleks seperti *digit tertentu (\d{3,})*, *huruf kapital di awal ([A-Z][a-z]+)*, atau *struktur email dan nomor telepon*. Kemampuan ini membuat regex menjadi alat yang sangat fleksibel untuk sistem yang berbasis teks bebas.

Regex dapat digunakan baik dalam bahasa pemrograman seperti Python (melalui modul re) maupun Java. Dalam implementasinya, fungsi seperti re.search() dan re.findall() digunakan untuk mencari semua kemunculan pola dalam teks, lalu menyorotnya sebagai bagian penting dari hasil pencarian. Misalnya, sistem dapat menyoroti kata “destroy” di tengah efek panjang sebuah kartu, memberikan pengalaman pengguna yang lebih informatif dan terarah. Secara umum, Regex merupakan komponen kunci dalam sistem pencarian teks cerdas, khususnya untuk kasus dengan format pola yang berulang dan variasi input tinggi seperti dalam basis data efek kartu permainan.

Penerapan regex juga sangat cocok dikombinasikan dengan teknik fuzzy matching. Dalam hal ini, regex digunakan untuk melakukan filter awal berdasarkan pola eksak seperti keyword atau format lain, sementara fuzzy matching digunakan untuk mencocokkan teks yang lebih bebas dan mungkin tidak sepenuhnya cocok secara karakter demi karakter. Kombinasi ini menghasilkan sistem pencarian yang lebih efisien dan akurat, karena pengguna dapat menemukan kartu dengan efek yang relevan, bahkan jika input mereka tidak persis sama dengan data deskripsi asli. Di sisi lain, regex juga memungkinkan sistem untuk menampilkan hasil dengan highlight visual pada keyword yang ditemukan, meningkatkan pengalaman pengguna dalam menelusuri hasil pencarian.

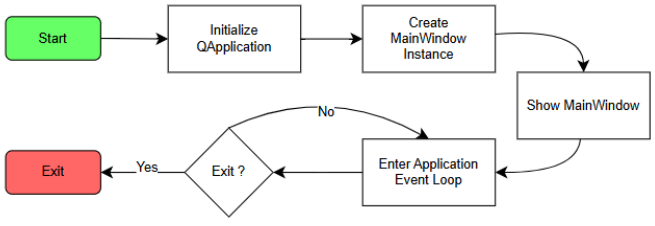
III. Implementasi

Implementasi Program dapat dilihat lebih detail di github berikut: https://github.com/Narrr21/Kartu_Yu-Gi-Oh/

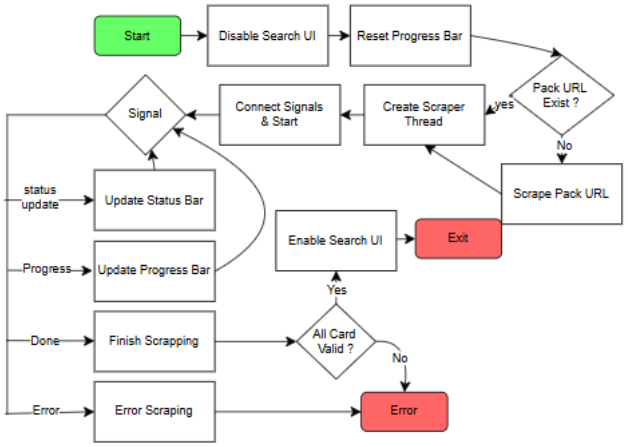
Program dibuat dengan menggunakan Bahasa Python dengan PyQT sebagai library untuk UI, FuzzyWuzzy untuk Fuzzy match dan Re untuk Regex. Scraping data dilakukan dari situs web DB Yu-Gi-Oh dan disimpan sebagai JSON, scraping juga menyimpan cache untuk menghindari keperluan scraping ulang dan mempermudah pengetesan.

Kode akan memiliki 2 folder (selain folder venv) yaitu JSON dan src. JSON untuk menyimpan cache dan src sebagai tempat implementasi kode. Src dibagi menjadi beberapa bagian penting Home.py yang digunakan untuk Struktur window utama Main.py digunakan sebagai titik awal dan tempat semua kode akan berjalan. Scraper.py yang digunakan untuk web scraping. DB.py yang digunakan untuk menghandle pencarian, penambahan atau penghapusan database JSON. Search.py yang digunakan untuk memanggil thread searching, Config.py yang digunakan untuk menyimpan variable variable penting yang bisa sering kali dirubah untuk keperluan pengetesan. Dan Toast.py untuk meningkatkan User Experience.

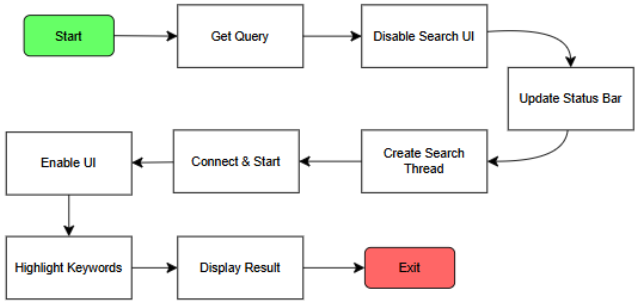
A. *Flowchart*



Untuk Main Apps dimulai dengan menginitialize QApps dan membuat main window UI dan sebagai interface utama dan memasuki loop utama aplikasi.



Scraping dimulain dengan dijalankan function scrape_initial_data(). Yang dilakukan adalah dengan mendisable Search UI sementara dan mereset Progress Bar yang ada. Setelah itu sistem akan mengecek apakah Pack URL ada di dalam folder JSON. Jika tidak ada akan di scraping terlebih dahulu link untuk setiap pack. Jika sudah ada maka akan dilanjutkan untuk dimuat dengan dibuat Thread Scraper yang mendengarkan update dari thread seperti Progress, Status (Saat berganti pack yang akan di scrap). Setelah selesai akan di cek validitas data dan keluar dari sistem. Jika pada alur ada kesalahan akan segera ditangkap dan menjadi pesan Error yang akan ditampilkan lewat Toast



Search dimulai dari saat user memasukkan Query, query ini terdiri dari keyword apa yang dimasukkan oleh user dan apakah user menginginkan Best match saja atau multi match. Sama halnya dengan sebelumnya pertama tama UI akan di disable untuk sementara dan Melakukan update status bar, setelah itu dibuat Search Thread dan masuk ke function find_best_match atau find_multiple_match tergantung pada input query. Kemudian dilanjutkan dengan highlight keyword kemudian di display.

B. *Algoritma Scraping*

```
    def extract_card_info(self, card_div) -
> Optional[Card]:
        name =
```

```
self.extract_card_name(card_div)
        if not name:
            return None
        attribute =
self.extract_attribute(card_div)
        level =
self.extract_level_rank(card_div)
        card_type =
self.extract_card_type(card_div)
        atk, defense =
self.extract_atk_def(card_div)
        description =
self.extract_description(card_div)
        rarity =
self.extract_rarity(card_div)

        return Card(name, attribute, level,
card_type, atk, defense, description,
rarity)

    def extract_card_name(self, card_div):
        selectors = [
            'span.card_name',
            '.card_name_flex_1',
            '.card_name',
            'a[title]',
            '.t_title a'
        ]

        for selector in selectors:
            element =
card_div.select_one(selector)
            if element:
                name = element.get('title')
or element.text.strip()
                if name:
                    return name
        return None
```

Sebagian kode diatas adalah bagian dari Scraper.py pada sistem ini. Untuk mengekstrak informasi kartu digunakan pustaka requests untuk membuat permintaan HTTP dan BeautifulSoup untuk mengurai konten HTML. Fungsionalitas inti dari scraper dipecah menjadi beberapa fungsi dan kelas. Seperti fungsi scrape_pack_urls yang memulai proses dengan identifikasi URL dari berbagai paket kartu yang tersedia di halaman beranda. Ia menavigasi struktur HTML untuk menemukan nama pack dan URL yang sesuai dan disimpan di JSON untuk di proses nantinya.

Thread yang dibuat adalah bagian paling penting Scraper.py yaitu ScraperThread. Thread ini beroperasi di thread terpisah yaitu QThread yang disediakan oleh PyQt5 hal ini untuk menjaga aplikasi tetap responsive selama proses scraping yang berpotensi memakan waktu yang lama. Sebelum proses scraping dimulai, sistem akan mencoba terlebih dahulu memuat data kartu dari cache (Disimpan di JSON jika sudah pernah scraping). Jika cache tidak ada, akan dimulai pembacaan URL untuk setiap pack dari Pack_URL JSON yang dibuat dari fungsi scrape_pack_urls. Kemudian secara berurutan akan mengunjungi setiap Pack dan memanggil scrape_cards_from_url. Fungsi tersebut digunakan untuk mengambil data semua Kartu pada suatu pack dan pada saat ini akan mengirim update ke progress bar dan status_update.

Pada function inilah dipanggil function yang sudah diberikan diatas yaitu Extract card info. Fungsi ini akan mengambil detail dari suatu kartu yang ditemukan, dimulai dari nama, atribut, level, type, atk, def, deskripsi efek dan rarity. Hal ini adalah kunci Dimana semua informasi metadata kartu di ambil dalam proses scraping. Tiap pengambilan informasi dibuatkan function yang mengurusnya masing masing. Pada contoh hanya card_name saja yang diberikan untuk menghindari banyaknya informasi pada bagian ini. Pada extract card name dapat dilihat jika terdapat beberapa selector yang disiapkan untuk mengantisipasi kemungkinan penggunaan tag html yang berbeda untuk sebuah nama. Dan sistem hanya akan mencari 1 yang valid dari semua selector jika tidak ada selector yang valid maka nilai None akan diberikan.

C. Algoritma Searching

```
    def find_multiple_matches(self, query:
str) -> List[Tuple[Card, int, str]]:
        if not self.cards:
            return []

        choices =
list(self.search_corpus.keys())
        matches = process.extract(query,
choices, limit=config.MULTI_SEARCH_LIMIT)

        results = []
        for matched_string,
similarity_score in matches:
            if similarity_score >=
config.FUZZY_SCORE_CUTOFF_MULTI:
                matched_card =
self.search_corpus[matched_string]

results.append((matched_card,
similarity_score, query))
```

```
        return results
```

Kode ini adalah bagian dari DB.py disini DB mengelola penyimpanan, dan pencarian kartu. Kelas Card juga ada disini sebagai dataclass untuk penyimpanan dan pengolahan kartu. Fungsi yang ditampilkan adalah find_multiple_match yang seharusnya menjadi function yang akan sering digunakan oleh user untuk mencari banyak kartu sekaligus. Fungsi ini akan menemukan beberapa kartu yang relevan dengan query menggunakan fuzzywuzzy.process.extract(). Fungsi ini akan mengembalikan tuple Card yang cocok, beserta skor kemiripan dan query asli untuk setiap pencocokan yang memenusi threshold.

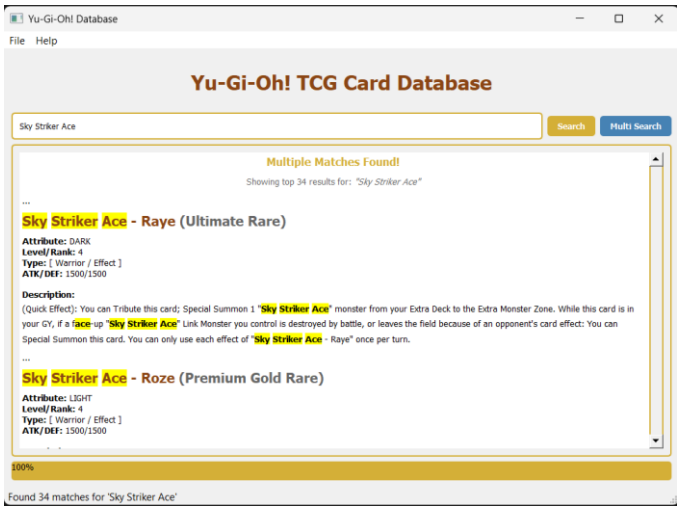
Fungsi ini didudkung dengam fungsi display yang ada pada Home.py untuk ditampilkan, Digunakan dengan menggunakan fungsi pada Card dataclass yang mempunyai fungsi to_html yang mengatur highlight dengan regex berdasarkan hasil query dari fuzzy

IV. Hasil dan Pembahasan

Pengetesan dilakukan dengan mencari kartu kartu, misalnya dari archetype Sky Striker, atau Kartu kartu monster jenis lain seperti Fusion, Xyz dst.

Sebagai konsistensi akan digunakan kartu yang di scrap pada tanggal 16 Juni 2025. Kartu ada pada folder JSON dengan nama card_cache_test.json. Untuk menggunakannya Ganti nama menjadi card_cache.json dan jalankan aplikasi.

Pengetesan:



Gambar 1 Contoh hasil searching

Kondisi Tes	Query	Hasil
DB : test 1	Sky Striker Ace	Time: 0.7794s Total: 34 result Top : Sky Striker Ace - Raye
DB : test 1	Blue eye white dagon	Time: 0.7469s Total : 40 result Top : Blue-Eyes White Dragon
DB : test 1	Fussion Summoned Dark Paladin	Time: 0,7893s Total: 1055 result Top: Dark Paladin
DB : test 1	This card gains 500 ATK for each material it had when it banished.	Time: 0,9078s Total: 3088 result Top: Galaxy-Eyes Photon Dragon
DB : test 1	Fiend Synchro Tuner Effect	Time: 0,8029s Total: 2081 result Top: Fabbled Gammajin
DB : test 1	3050 1950	Time: 0,7851 s Total: 12 results Top: Fabled Levilazibul
DB : test 2	Blue eye white dagon	Time : 0,1119s Total : 5 result Top : Blue-Eyes White Dragon
DB : test 2	3050 1950	Time: 0.1049 s Total: 4 result Top: Fabbled Levilazibul

Test diatas dilakukan untuk dua DB (tes 1 dan 2). Tes 1 dengan jumlah pack yang jauh lebih banyak dibanding tes 2. Query yang digunakan diatas untuk mengetes fungsionalitas awal dari kedua database, seperti nama kartu, kondisi typo, deskripsi, tipe kartu dan nilai ATK atau Defense. Semua dapat selesai dengan baik dalam waktu sub-1 detik

Pengujian selanjutnya adalah mencari tahu seberapa akurat hubungan dari kartu yang akan dicari, pada kasus ini akan digunakan archetype Sky Striker sebagai deck utama dan mencari tahu kartu apa yang dibutuhkan

Query yang digunakan adalah: #Sky# #Striker# Summon Special GY Discard Banish Search Destroy Card Gain Deck

Tanda # yang ada digunakan sebagai tanda jika kata itu harus lah ada pada hasil. Dari query pada data test_1 didapatkan 17 hasil yang memenuhi. Yaitu:

- Pillar of the Future – Cyanos
- Sage of Strength – Akash
- Sage of Wisdom – Himmel
- Sage of Benevolance – Ciela
- Sky Striker Ace – Azalea
- Sky Striker Ace – Azalea Temperance

- Sky Striker Ace – Zeke
- Sky Striker Ace – Raye
- Sky Striker Ace – Hayate
- Sky Striker Ace – Shizuku
- Sky Striker Mobilize – Linkage!
- Surgical Striker – H.A.M.P.
- Sky Striker Airspace – Area Zero
- Sky Striker Mecha – Shark Cannon
- Sky Striker Ace – Kaina
- Sky Striker Ace – Roze
- Sky Striker Ace – Kagari

Dari hasil pencarian diatas ditemuan 17 kartu yang relevan dengan archetype Sky Striker, yang menunjukkan hubungan kuat antara kata kunci dan sinergi internal yang ada dalam deck. Pendekatan ini pencarian berbasis kata kunci efektif dalam mengidentifikasi kartu-kartu yang memiliki peran strategis penting. Baik dalam core engine deck maupun pendukung playstyle deck itu. Dari data diatas kartu kartu Ace seperti Raye, Roze, dan Kagari adalah contoh utama dair monster yang mendukung mekanisma pemanggilan berulang dan pengisian GY (Graveyard). Kartu lain seperti Sharck Cannon dan H.A.M.P. memperkuat kontrol kepada lawan dengan menggunakan banish dan destroy. Selain itu juga ada Kartu Linkage! Dan Area Zero yang digunakan sebagai kartu seach dan deck thinning, yang penting untuk konsistensi strategi deck ini dalam duel yang kompetitif.

Lebih lanjut lagi, untuk deskripsi kartu yang punya struktur jelas terarah seperti Pillar of the Future – Cyanos mampu langsung memanggil Ace – Roze dari Deck atau GY dengan hanya membuang 1 spell di tangan, Hal ini mengaktifkan combo line yang kuat dan efisien dengan mematuhi batasan Special Summon dari extra deck yang hanya untuk Machine monsters. Hal ini tidak menjadi masalah untuk archetype ini sehingga Jajaran kartu Sage seperti Akash, Himmel, Ciela menambahkan perlindungam, pengambilan spell yang di banish, gangguan terhadap musuh dengan biaya mendiscard suatu spell (yang justru adalah tujuan utama deck ini) artinya sinergi ini sangatlah baik dan memperkuat kartu lainnya. Untuk link monster seperti Azalea dan Zeke digunakan untuk utility tambahan berupa banish atau destroy ke target, sementara Kagari dan Shizuku memberikan keunggulan sumber daya dengan mencari dan mengembalikan Spell dari GY atau Deck yang meningkatkan kemampuan deck ini untuk memberikan player kontrol yang baik. Strategi ini berbasis tempo, manajemen sumber daya (umumnya Spell) dan adaptasi melawan banyak situasi (Karena kontrol yang baik)

V. KESIMPULAN

Dengan demikian, aplikasi pencarian kartu Yu-Gi-Oh! Berhasil dikembangkan sebagai Solusi yang cukup efektif untuk membantu duelist dalam memahami dan menemukan sinergi antar kartu. Tentu cara ini tetap mengalami keterbatasan untuk sinergi kartu di luar archetype tertentu, hal ini dikarenakan kartu dengan jenis yang berbeda bisa dikombinasikan untuk suatu archetype tanpa adanya satu nama yang sama di antara mereka. Sistem ini sudah mencukupi untuk mencari sinergi kartu dibawah nama yang sama dan sebagai planning awal untuk mengetahui hubungan antar kartu secara sederhana.

Sistem ini mengimplementasika pencarian yang berbasis teks dengan memanfaatkan fuzzy matching untuk menangani kesalahan input dan dimungkinkan untuk mencari berdasarkan meta data suatu kartu (Nama, atribut, tipe, deskripsi dll).

Data kartu diambil langsung dari website DB kartu Yu-Gi-Oh! dan disimpan secara terstruktur dalam sebuah JSON yang disimpan sebagai cache untuk lebih jauh lagi mengurangi scraping yang tidak perlu. Penggunaan fitur highlight dengan menggunakan Regular Expression (Regex) terbukti meningkatkan pengalaman pengguna untuk secara cepat mengidentifikasi bagian mana dari kartu yang memiliki kata kunci tersebut. Dari data pengujian juga didapat jika pegujian berlangsung secara nyaman, karena tidak terlalu lama untuk melakukan pencarian, masih ada dalam batas sub-1 detik. Sehingga tidak terlalu terasa ada waktu dalam pencarian. Pencarian juga dapat menggunakan special karakter (#) untuk membungkus suatu kata kunci agar dipastikan ada pada suatu kartu. Hal ini untuk menambah control user pada query agar memastikan hanya menampilkan kartu yang dimaksud oleh user.

Tentunya sistem masih memiliki banyak kekurangan dan banyak hal yang dapat ditingkatkan dan diperbaiki. Penambahan filter pada kartu adalah cara yang baik untuk lebih lanjut mempermudah pencarian. Selain itu penambahan paging agar pengguna tidak perlu scrolling jauh jika ada banyak hasil dan lebih mudah kembali membaca kartu di bagian tengah list. Selain dari itu, pengambilan image kartu juga akan sangat membantu duelist untuk tau seperti apa kartu yang disebutkan. Karena untuk beberapa duelist, image dari kartu terkadang jauh lebih penting daripada efeknya. Selain itu juga belum ada tag kartu mana yang termasuk ke banlist dan deck deck yang sering digunakan selama ini, selain itu tidak ada campur tangan tren juga dalam pencarian relevansi. Hal ini dapat mempermudah sistem menebak kartu apa yang mungkin relevan tanpa ada di nama yang sama dengan kartu yang dicari.

Link Youtube : https://youtu.be/FREN8rAxqUs

VI. UCAPAN TERIMAKASIH

Pertama-tama, penulis memanjatkan puji dan Syukur kepada Tuhan YME, atas berkat dan nikmatnya diberikan kelancaran untuk menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih kepada kedua orang tua, serta teman teman yang sudah mendukung selama pembuatan makalah ini.

Kemudian penulis juga memberikan rasa terimakasih khusus kepada Bapak Rinaldi Munir selaku dosen pengajar mata kuliah IF2211 Strategi Algoritma kelas K2, yang telah memberikan penulis ilmu dan kesempatan untuk mengerjakan makalah ini.

REFERENSI

[1] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, Draft 3rd ed. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/. [Diakses: 23-Jun-2025].

[2] [2] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[3] M. L. Khodra, *String Matching dengan Regular Expression*, Institut Teknologi Bandung, 2025.

[4] "Yu-Gi-Oh! Official Card Database," *Konami Digital Entertainment*. [Online]. Tersedia: https://www.db.yugioh-card.com/yugiohdb/. [Diakses: 24 Juni 2025].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 23 Juni 2025

Nadhif Al Rozin 13523076