

# Integrasi Named Entity Recognition dan Trend-Based Query Expansion sebagai Upaya Peningkatan Relevansi *Search Engine*

Hafid Sasayuda Ambardi

Departemen Ilmu Komputer dan  
Elektronika  
Universitas Gadjah Mada  
Yogyakarta, Indonesia

[hafidsasayudaambardi@mail.ugm.ac.id](mailto:hafidsasayudaambardi@mail.ugm.ac.id)

Muhammad Dafa Wisnu Galih

Departemen Ilmu Komputer dan  
Elektronika  
Universitas Gadjah Mada  
Yogyakarta, Indonesia

[muhammaddafawisnugalih@mail.ugm.ac.id](mailto:muhammaddafawisnugalih@mail.ugm.ac.id)

Joseph Greffen Komala

Departemen Ilmu Komputer dan  
Elektronika  
Universitas Gadjah Mada  
Yogyakarta, Indonesia

[josephgreffenkomala@mail.ugm.ac.id](mailto:josephgreffenkomala@mail.ugm.ac.id)

**Abstrak –** Dalam era digital saat ini, ahli industri semakin membutuhkan search engine yang tidak hanya akurat dan fleksibel tetapi juga mampu memberikan wawasan mendalam untuk memaksimalkan potensi e-commerce. Penelitian ini bertujuan untuk mengembangkan dan mengimplementasikan sebuah sistem pencarian yang inovatif, yang dapat secara efektif menentukan produk-produk yang sedang tren, mengkategorisasikan produk dengan lebih efisien, serta menerapkan kombinasi framework dari teknologi terbaru guna meningkatkan performa dan relevansi hasil pencarian. Sistem ini menggunakan pendekatan pencarian multimodal yang memungkinkan pengguna mencari produk melalui teks, gambar, atau kombinasi keduanya, memberikan fleksibilitas yang lebih besar. Qdrant lokal digunakan untuk memastikan kecepatan dan responsivitas pencarian, sementara algoritma max\_similarity diterapkan untuk meningkatkan akurasi hasil pencarian. Selain itu, integrasi Named Entity Recognition (NER) memungkinkan sistem ini memberikan wawasan mendalam mengenai produk dengan mengidentifikasi entitas penting seperti jenis produk, material, dan warna.

**Keywords—**Trend, Search Engine, Named Entity Recognition, Contrastive Learning

## I. PENDAHULUAN

### A. Latar Belakang

Tren di media sosial berkembang dengan sangat cepat dan sering kali berubah dalam waktu singkat. Tercatat bahwa sekitar 62.3% atau 5.07 miliar orang dari populasi dunia menggunakan media sosial [1]. Dengan pengguna yang banyak dan pertukaran informasi terjadi dengan sangat cepat, menyebabkan tren juga berubah dengan cepat. Platform seperti X, TikTok, dan Instagram memainkan peran penting dalam mempengaruhi apa yang sedang populer, dengan pengguna dan influencer sering kali menjadi pendorong utama perubahan tren ini.

Munculnya *social commerce* yang secara langsung mengintegrasikan tren di media sosial dengan ecommerce

mempermudah pengguna untuk membeli produk yang sedang populer. Sebuah survei berskala global menunjukkan bahwa tujuh dari sepuluh pembeli online di seluruh dunia, berusia antara 27 dan 42 tahun, pernah melakukan pembelian langsung melalui media sosial pada kuartal kedua tahun 2023 [2]. Dari sudut pandang penjual, TikTok sebagai salah satu media sosial terbesar di dunia dapat meningkatkan omset penjualan mereka secara signifikan dengan pengaruh sebesar 64.5% [3]. Hal tersebut menunjukkan bahwa pemanfaatan tren yang ada di media sosial akan meningkatkan ketertarikan pengguna dan mendorong angka penjualan secara signifikan.

Konsekuensi dengan terus berkembangnya tren adalah istilah baru yang terus bermunculan, yang menyebabkan pengguna seringkali kesulitan mendeskripsikan produk apa yang mereka inginkan. Salah satu fitur yang dapat membantu pengguna adalah *multimodal search* dalam mesin pencari, yaitu dengan menggabungkan pencarian teks dan gambar. Penelitian [4] menjelaskan bahwa *multimodal search* mampu memberikan kinerja yang lebih baik dibandingkan dengan fitur unimodal – hanya teks saja – dalam mengidentifikasi kepuasan pengguna. *Multimodal search* dapat membantu mengatasi konsekuensi tren cepat tersebut dengan memungkinkan pengguna mencari produk berdasarkan gambar produk yang pernah mereka lihat tetapi tidak dapat pengguna deskripsikan secara rinci.

Hasil pencarian dengan *multimodal search engine* memerlukan perincian yang cukup mendalam melalui kategorisasi produk dan *query*. Named Entity Recognition (NER) menjadi metode yang potensial untuk mengekstraksi *tag* dari keduanya. Metode tersebut dapat mengenali dan mengklasifikasikan entitas dalam suatu teks, seperti nama produk, merek, warna, atau kategori. Beberapa riset menunjukkan bagaimana NER diimplementasikan dalam bidang medis dan pencarian artikel publikasi [5-8]. Kami melihat potensi tersebut juga dapat diimplementasikan dalam ekstraksi entitas baik pada query maupun judul produk di *e-commerce search engine* untuk mendapatkan produk yang sesuai dengan keinginan pengguna.

Dalam penelitian ini, kami mengombinasikan *multimodal search engine* yang dapat menangkap tren yang

sedang berkembang dengan NER sebagai optimisasi hasil pencarian. *Multimodal search engine* yang memanfaatkan data visual dan teks dapat memberikan konteks yang lebih luas dan akurat dalam pencarian produk. Dengan mengintegrasikan NER, sistem dapat secara efektif mengenali tren, mengkategorikan, dan menyaring hasil pencarian untuk memaksimalkan nilai relevansi produk. Kombinasi ini tidak hanya meningkatkan pengalaman pengguna dalam menggunakan *search engine*, tetapi juga memberikan meningkatkan angka jual beli dalam pasar *e-commerce* yang dinamis.

### B. Rumusan Masalah

Berdasarkan latar belakang yang telah diajukan, berikut kami rumuskan beberapa masalah inti yang dapat menjadi fokus kami dalam penelitian ini:

1. Bagaimana cara mengidentifikasi dan menangkap tren yang cepat berubah di media sosial, serta mengintegrasikan informasi tersebut ke dalam sistem *e-commerce*?
2. Bagaimana fitur *multimodal search* dapat memudahkan pengguna dalam mencari produk yang mereka inginkan sekaligus meningkatkan relevansi hasil pencarian?
3. Bagaimana Named Entity Recognition (NER) dapat diintegrasikan dalam sistem pencarian *e-commerce* untuk mengoptimalkan kategorisasi produk dan query?
4. Bagaimana hasil kombinasi terbaik antara multimodal search engine, trend extraction dan NER dapat meningkatkan pengalaman pengguna dalam mencari produk?

### C. Tujuan Penelitian

Dalam mengembangkan solusi dari rumusan masalah sebelumnya, penelitian ini memiliki beberapa poin tujuan berikut:

1. Mengembangkan metode untuk mengidentifikasi dan mengintegrasikan tren dari media sosial ke dalam sistem *search engine e-commerce*.
2. Mengevaluasi bagaimana fitur *multimodal search* yang menggabungkan pencarian teks dan gambar, dapat memudahkan pengguna menemukan produk yang dicari.
3. Mengimplementasikan NER dalam *search engine e-commerce* untuk merincikan proses kategorisasi produk dan kueri.
4. Menganalisis hasil terbaik dari kombinasi multimodal search engine, ekstraksi tren, dan NER dalam peningkatan kepuasan pengguna ketika mencari produk.

### D. Manfaat Penelitian

Dengan mengembangkan solusi dan memenuhi tujuan penelitian, berikut beberapa manfaat yang terwujud dengan adanya penelitian ini:

1. Meningkatkan kemampuan sistem *ecommerce* dalam menawarkan produk yang sesuai dengan tren terkini, sehingga dapat menarik lebih banyak pengguna dan meningkatkan konversi penjualan.

2. Mempermudah pengguna dalam mencari produk yang sesuai dengan keinginan mereka, meningkatkan kepuasan pengguna, dan mengurangi waktu pencarian produk.
3. Menyediakan hasil pencarian yang lebih relevan dan tepat sasaran dengan mengklasifikasikan entitas penting dalam teks, yang pada gilirannya meningkatkan pengalaman pengguna dan efisiensi pencarian
4. Menyediakan wawasan tentang bagaimana kombinasi teknologi ini dapat memperbaiki interaksi pengguna dengan sistem pencarian, meningkatkan kepuasan pelanggan, dan mendorong angka penjualan di pasar *e-commerce* yang dinamis.

## II. METODOLOGI

### A. Diskusi dengan Ahli Industri

Dalam upaya untuk memahami secara lebih mendalam mengenai topik permasalahan yang terkait dengan pendefinisian *problem* dan optimisasi *search engine*, kami melaksanakan diskusi dengan beberapa ahli yang kami anggap cukup memiliki *domain knowledge* di bidang tersebut. Langkah ini cukup penting untuk mendapatkan pengetahuan, gagasan, serta jawaban baik secara teoritis maupun lebih aplikatif.

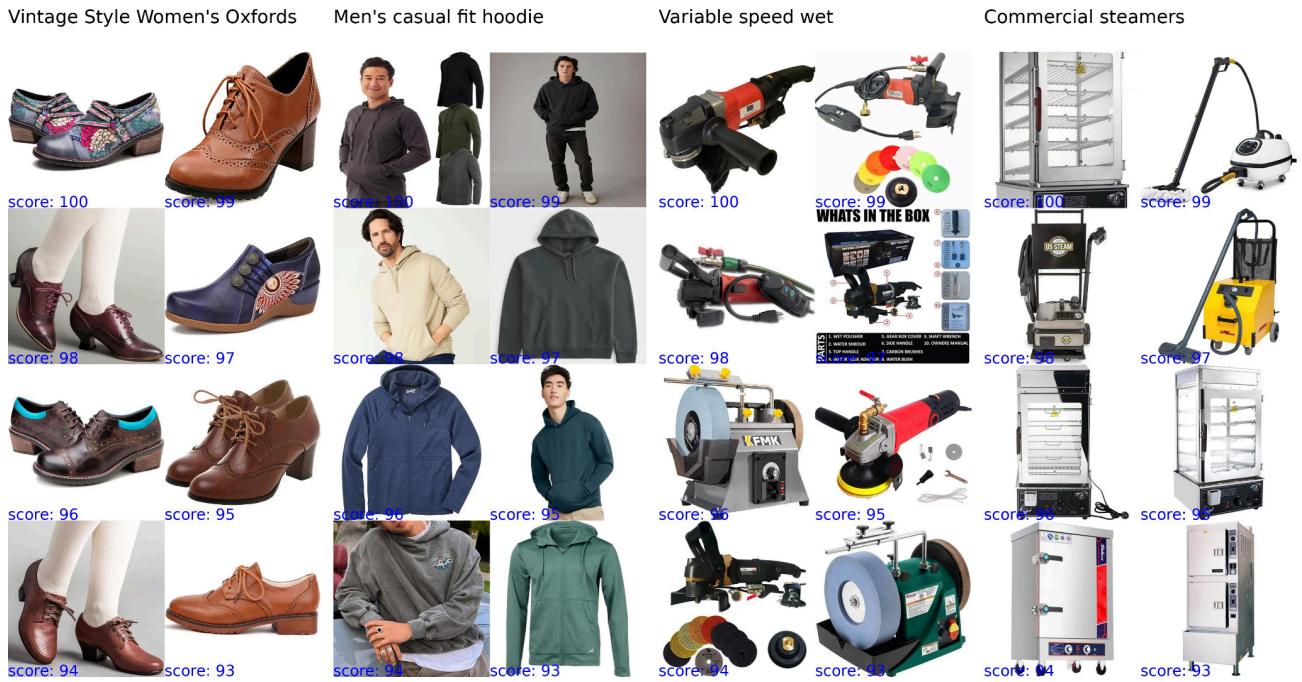
Dengan adanya diskusi tersebut diharapkan juga dapat men-trigger ide dan solusi berdasarkan problematika di lapangan. Diskusi dilaksanakan dengan tiga orang praktisi yang telah meniti karir di beberapa perusahaan unicorn *e-commerce* dan start-up di bidang teknologi.

**Mohammad Rinaldi.** Mohammad Rinaldi pernah bekerja di perusahaan besar seperti Intel, Dana, dan Traveloka, khususnya dalam bidang Sektor Publik dan Pemerintahan. Beliau sering mewakili perusahaan, dan memberikan beberapa insight dari segi bisnis:

1. Preferensi orang sering dapat dikelompokkan, dan mengidentifikasi apa yang diinginkan setiap kelompok sangatlah penting. Hal ini berlaku untuk promosi, kualitas produk, dan bahkan hasil mesin pencari.
2. Pelanggan sering kali kewalahan jika dihadapi terlalu banyak pilihan, jadi semakin sedikit filter dan pengaturan yang harus diisi seseorang untuk menemukan apa yang diinginkannya, semakin baik sebuah bisnis akan berkembang.

**Andre Setiawan.** Dengan pengalaman karirnya sebagai *Product Manager* di Shopee dan Dana, diskusi menghasilkan wawasan dan cara pandang baru mengenai cara kerja *search engine* di Shopee dan tantangan yang dihadapinya. Poin-poin utama pembahasan antara lain: efektivitas *search engine*, keluhan pengguna, dukungan untuk penjual kecil, serta manfaat *search engine* untuk perusahaan, dan evaluasi serta dampak perubahan pada *search engine*.

Dari diskusi dengan pembicara, kami memperoleh wawasan mengenai tiga masalah umum yang sering dihadapi dalam



Gambar 1. Contoh Kueri, Gambar, dan Score di Dataset Marqo-GS-10M

e-commerce:

1. Kesulitan dalam menentukan produk mana yang sebaiknya direkomendasikan untuk query yang pendek.
2. Kesulitan dalam merekomendasikan produk yang relevan dengan tren terbaru.
3. Kesulitan dalam mencari produk yang akurat ketika terdapat kesalahan pengetikan (typo).

**Pandu Kerr.** Sebagai Vice President Marqo, sebuah startup yang fokus menyediakan solusi mesin pencari melalui pencarian berbasis tensor, dari pertemuan tersebut kami mencapai beberapa kesimpulan:

1. Basis Data Vektor adalah cara yang tepat karena dapat menangkap makna semantik dan memungkinkan pencarian multimodal.
2. Pencarian multimodal adalah cara yang paling representatif untuk melakukan pencarian dibandingkan dengan pencarian teks saja. Hal tersebut disebabkan oleh banyak aspek seperti pola, bentuk, dan detail visual lainnya yang seringkali tidak dijelaskan sepenuhnya dalam deskripsi produk.
3. Pandu Kerr merekomendasikan penggunaan model Generalized Contrastive Learning, karena model ini mampu mempelajari sejumlah teks untuk satu gambar, sehingga semakin memperkaya pemahaman semantik Model Machine Learning.

Berdasarkan langkah diskusi dengan ahli yang dilanjutkan dengan brainstorming mengenai problem dan solusi yang diangkat, kami mendapatkan beberapa ide pokok mengenai topik-topik apa saja yang digunakan pada

penelitian ini, yaitu Multimodal Search Engine dan Named Entity Relation (NER) sebagai dasar dari pencarian Trend Extraction.

#### B. Multimodal Search Engine

*Multimodal Search Engine* adalah jenis search engine yang memungkinkan user untuk menggunakan lebih dari 1 modalitas dalam waktu yang bersamaan. Modalitas ini bisa berupa, teks, gambar, suara dan video. Sebagai contoh, dalam *multimodal search engine*, pengguna dapat memasukan teks dan gambar sebagai input sekaligus untuk melakukan suatu kueri. Secara umum *multimodal search engine* memberikan pilihan lebih banyak kepada pengguna untuk melakukan kueri.

**Dataset.** Kami menggunakan sebagian dari data Marqo-GS-10M yang secara keseluruhan berisi sekitar 100 ribu (98.286) kueri dari hasil generasi dengan model GPT-4 [9]. Dengan menggunakan kueri tersebut, sekitar 10 juta produk didapatkan dengan metode *scraping* di platform Google Shopping. Data yang diambil untuk setiap produk meliputi judul, gambar, hingga urutan posisi produk. Sebagai indikator skor relevansi, diambil berdasarkan posisi peringkat produk dari pencarian Google Shopping.. Untuk menghitung skor, kami menggunakan rumus  $s = 101 - rank$ , sehingga skornya berkisar antara 1 hingga 100 [9].

Dengan besarnya jumlah data dan keterbatasan gawai dan waktu yang dimiliki, kami hanya menggunakan sebagian data yaitu sekitar 15 ribu kueri dan 100 ribu data dengan kategori fesyen wanita. Gambar 1. menunjukkan contoh kueri serta gambar dan skor produk dari dataset. Kategori ini dipilih karena memiliki diversifikasi yang tinggi, dimana hal tersebut sangat penting untuk model nantinya dapat memahami produk secara multimodal. Dalam pembahasan

selanjutnya, data kueri dari dataset ini akan disebut *Query History*.

**Model.** Sebagai tindak lanjut dari dataset yang telah dikurasi, Marqo dalam publikasi yang sama mengembangkan model dengan konsep Generalized Contrastive Learning (GCL) [9]. Model tersebut menggunakan model CLIP-Vit-L-14 [11] sebagai *baseline*. Marqo melakukan *fine-tuning* terhadap model tersebut dengan menggunakan keseluruhan data Marqo-GS-10M. Langkah tersebut meningkatkan akurasi model secara drastis, meningkat sebesar 94.5% dalam NDCG@10 untuk evaluasi *in-domain* dan sebesar 26.3 – 48.8 persen untuk evaluasi *cold-start*, relatif terhadap model CLIP-Vit-L-14.

**Vector Database.** Vector Database mewakili data menggunakan embedding vektor, yang pada umumnya dihasilkan oleh model pembelajaran mesin yang sudah dilatih menggunakan dataset spesifik dan bertujuan untuk menangkap makna semantik. Kami memilih Qdrant sebagai database vektor kami karena kinerjanya yang kuat, fleksibilitas, dan skalabilitas. Qdrant menyediakan kemampuan pengindeksan dan pencarian efisien yang penting untuk mengelola kumpulan data besar dan melakukan kueri real-time, berkat landasannya di Rust, bahasa pemrograman yang sangat efisien. Selain itu, Qdrant mendukung representasi multivektor, memungkinkan kami menyimpan lebih dari satu embeddings untuk satu produk. Fitur ini memfasilitasi pencarian kesamaan tanpa mengubah representasi vektor asli, berbeda dengan metode lain seperti pooling atau sum.

Keuntungan lain menggunakan Qdrant adalah dukungannya terhadap penilaian Max Similarity, yang mengevaluasi kesamaan maksimum antara setiap pasangan vektor. Pendekatan ini membantu memastikan bahwa informasi yang tidak ada dalam satu vektor dapat dilengkapi dengan vektor lain, sehingga memberikan gambaran keseluruhan produk yang lebih akurat.

$$score = \sum_{i=1}^N \max_{j=1}^M Sim(vectorA_i, vectorB_j)$$

Dimana N merupakan jumlah vektor dalam matriks pertama, M adalah jumlah vektor dalam matriks kedua, dan *Sim* adalah fungsi kesamaan. Fungsi kesamaan yang digunakan adalah kesamaan *cosine*.

### C. Named Entity Recognition (NER)

Named Entity Recognition (NER) adalah salah satu cabang Natural Language Processing (NLP) yang memungkinkan model untuk mengenali dan mengkategorikan informasi penting dalam teks, seperti nama orang, tempat, tanggal, dan organisasi. Dalam penelitian ini, kami menggunakan NER untuk mengkategorisasi kueri input sebelum diproses di *search engine*.

**Dataset.** Penelitian [12] mengajukan dataset QueryNER, berisi sekitar 7,4 ribu kueri yang dipilih dari dataset Shopping Queries Data Set [13]. Tiap kueri ditokenisasi dan dilabeli secara manual untuk tiap tokennya di-*assign* ke 19 kategori seperti *core\_product\_type*, *product\_name*, dan

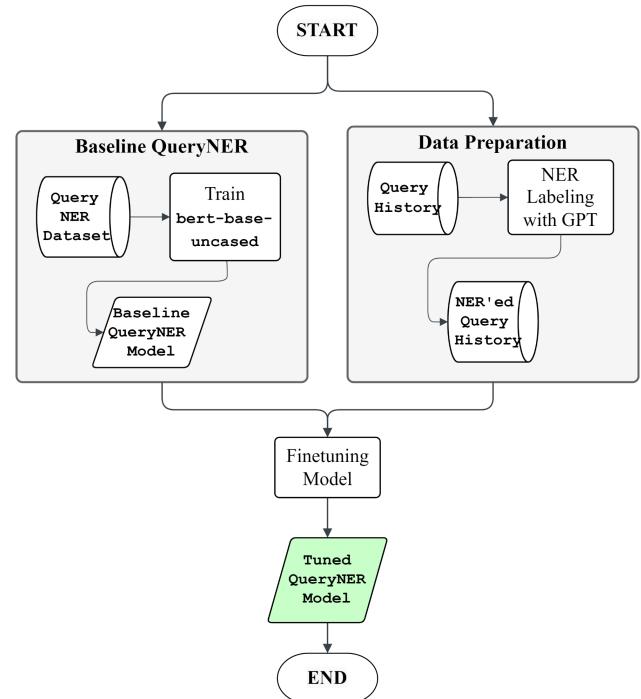
*color*.

Selain data tersebut, kami juga melakukan generasi data dengan format serupa, menggunakan data *Query History* yang juga digunakan dalam *Multimodal Search Engine*. Proses generasi data dilakukan dengan melakukan *prompt* terhadap model GPT-4o [10] melalui OpenAI API, menghasilkan sejumlah sejumlah 13.592 baris data. Nantinya data ini akan digunakan untuk fine tuning model QueryNER yang dipilih. Tabel I. menunjukkan data QueryNER. Kolom NER Tags bertindak sebagai label yang di-*mapping* ke total 34 kategori yang seperti *core\_product\_type*, *modifier*, *department*, *material*, dan *occasion*.

TABEL I. DATA QUERYNER

Tokens	Ner Tags
[ "teeth", "whitening", "sensitive", "teeth" ]	[ 9, 10, 17, 18 ]
[ "white", "duvet", "cover", "queen" ]	[ 3, 9, 10, 1 ]
[ "eminem", "iphone", "6", "case" ]	[ 7, 25, 26, 9 ]
....	....

**Model.** Selain mengajukan dataset QueryNER, [12] juga melakukan *finetuning* terhadap model *bert-base-uncased* [14] agar secara spesifik dapat melakukan tugas NER. Ada dua hasil model yang didapatkan dari penelitian tersebut, yaitu model dengan data QueryNER dan data QueryNER yang diaugmentasi dengan beberapa metode transformasi teks. Performa kedua model tersebut yang diukur dengan metriks F1 secara berturut-turut sebesar 61.29 dan 55.19.



Gambar 2. Diagram Alir Fine Tuning Model QueryNER

Pada akhirnya, kami menggunakan model QueryNER dengan augmentasi karena meskipun memiliki skor F1 lebih rendah, tetapi model tersebut lebih *robust* terhadap transformasi pada kueri seperti *typo* atau perbedaan posisi kata.

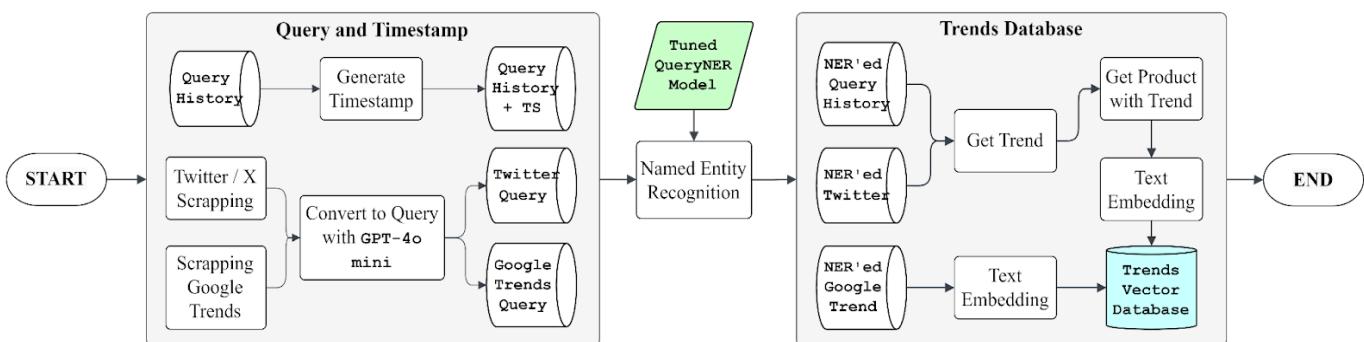
**Fine Tuning.** Untuk meningkatkan performa model *augmented* QueryNER, kami melakukan *fine tuning* dengan menggunakan data hasil generasi *Query History*. Dengan langkah ini, model akan mampu mengimplementasikan NER pada data produk dengan lebih baik.

Gambar 2. menunjukkan mekanisme fine tuning model QueryNER yang dimulai dengan mempersiapkan model dasar menggunakan dataset QueryNER awal. Lalu, data *Query History* yang belum berlabel diberi anotasi *entity name* untuk setiap tokennya menggunakan model GPT-4o. Langkah tersebut menghasilkan data *Query History* yang telah diberi label NER. Model *baseline* kemudian disempurnakan dengan menggunakan data berlabel baru ini, menghasilkan model QueryNER yang telah dilatih dengan dataset spesifik (*Query History*). Proses ini bertujuan untuk meningkatkan akurasi dan performa model dalam mengenali *entity name* pada query.

isu apa yang sedang terjadi di setiap harinya.

Gambar 3. menunjukkan langkah untuk Data **Query History** diberi *timestamp*. Kami juga mengambil data dari X(Twitter) dan Google Trends, kemudian diubah menjadi query menggunakan model GPT-4o mini. Hasil query dari **Query History**, Twitter, dan Google Trends kemudian diproses menggunakan model Named Entity Recognition (NER) yang telah di-tuning untuk mengidentifikasi entitas dalam query tersebut. Data yang telah diidentifikasi entitasnya ini disimpan dalam database. Dari database **Query History** dan **X (Twitter)** yang telah diidentifikasi entitasnya akan dicari produk produk yang entitasnya sedang tren, produk tersebut kemudian menjadi representasi vektor dan disimpan dalam database vektor tren. Selanjutnya dari Database **Google Trend** setiap kueri akan dimasukan dan ubah menjadi representasi vektor dan dimasukan ke dalam vektor embedding.

Pada saat ada kueri masuk, kueri akan langsung dicari kemiripannya dengan vektor yang ada di dalam database vektor tren untuk mendapat vektor yang terdekat. jika vektor yang terdekat itu dan vektor embedding memiliki nilai kesamaan diatas *similarity threshold*, yaitu diatas 1.45,



Gambar 3. Diagram Alir Pembuatan *Trends Vector Database*.

Nantinya, model NER ini akan digunakan untuk mengkategorisasi kueri pada proses pengambilan tren untuk kemudian dimasukkan ke Trend Vector Database.

#### D. Trend-Based Query Expansion (TBQE)

Kami menggunakan 3 dataset, yaitu:

**Query History** : Data ini mengandung 15.012 baris kueri. Data ini tidak memiliki *timestamp* sehingga dilakukan pembuatan *timestamp* secara sintesis untuk setiap kueri pada data ini.

**Scraping X(Twitter)**: Kami mengambil data pada X(Twitter) dengan 2 *hashtag* sebagai keyword, yaitu #fashionstyle dan #womanstyle pada rentang waktu 1 Januari 2023 dan 31 Desember 2023.

**Google Trend** : Kami mengambil data dari google trend mengenai “Today’s Trending”. Kami mengambil data pada hari tanggal 24 July 2024 hingga 1 Agustus 2024. Data ini mengandung 131 baris. Data ini memiliki kolom 3 kolom yaitu, timestamp,topik,jumlah searches. Kami mengambil data ini karena data ini dapat dengan jelas merepresentasikan

maka vektor terdekat itu akan diubah kembali menjadi text dan di *concat* dengan kueri yang dimasukan oleh pengguna. Jika nilai kesamaan vektor terdekat di bawah *threshold similarity* maka tidak di concat. lalu kueri tersebut akan digunakan untuk melakukan pencarian pada database produk. Untuk menghasilkan rekomendasi produk mengenai kueri tersebut

**Fungsi Get Trend.** Untuk mendapatkan informasi apakah suatu entitas sedang tren, kami membuat fungsi ‘get\_trend\_overall’. Fungsi get\_trend\_overall menerima beberapa parameter: data\_dict, query\_ts, df, dan threshold.

Pertama, fungsi ini mengubah *query\_ts* menjadi periode mingguan dan menginisialisasi *result\_dict* sebagai list kosong. Kemudian, untuk setiap *category* dan *value* dalam *data\_dict*, fungsi ini memfilter *df* untuk hanya menyertakan baris yang kolom kategorinya mengandung *value* pada kolom *category*. Selanjutnya, fungsi ini membuat kolom *week\_ts* yang merepresentasikan periode mingguan dari kolom *ts*. Setelah itu, data yang telah difilter dikelompokkan berdasarkan *week\_ts* dan menghitung

jumlah kemunculan tiap periode.

Fungsi ini juga membuat rentang periode mingguan dari '2023-01-01' hingga '2023-12-31', lalu menyesuaikan DataFrame yang telah dikelompokkan untuk menyertakan semua minggu dalam rentang ini, termasuk minggu yang datanya kosong, dan mengisinya dengan nilai 0. Setelah itu, fungsi menghitung *moving average* untuk data tersebut. Jika nilai *moving average* pada *query\_timestamp* melebihi ambang batas (threshold), maka nilai tersebut ditambahkan ke *result\_dict*. Akhirnya, *result\_dict* dikembalikan sebagai output fungsi, yang merupakan list yang berisi entitas yang sedang trend.

```

FUNCTION get_trend_overall(data_dict, query_ts, df,
    ↪ threshold):
    CONVERT query_ts to weekly period
    INITIALIZE result_dict as empty list

    FOR EACH category, values IN data_dict:
        FOR EACH value IN values:
            FILTER df WHERE COLUMN category CONTAINS
                ↪ value

            CREATE 'week_ts' column AS weekly period
                ↪ of 'ts' column

            GROUP filtered df BY 'week_ts' AND COUNT
                ↪ occurrences

            CREATE all_weeks AS weekly period range
                ↪ from '2023-01-01' to '2023-12-31'
            REINDEX grouped DataFrame TO INCLUDE
                ↪ all_weeks WITH MISSING VALUES
                ↪ FILLED WITH 0
            RECALCULATE rolling moving average

            GET moving average value FOR
                ↪ query_timestamp

            IF moving average value EXCEEDS threshold:
                ADD value TO result_dict

    RETURN result_dict

```

Gambar 4. Pseudocode untuk fungsi get\_trend\_overall()

#### E. Evaluation Metrics

Untuk melakukan pengukuran terhadap performa model, kami menggunakan metode kualitatif dan kuantitatif. Metode kuantitatif digunakan untuk mengukur tingkat kesesuaian hasil *search engine* dan kueri. Metode kualitatif digunakan untuk mengukur tingkat kesesuaian hasil *search engine* dengan tren atau isu-isu yang sedang terjadi.

##### 1) Kuantitatif

**Normalized Discounted Cumulative Gain.** NDCG adalah metrik pengukuran efektivitas sistem pencarian yang membandingkan hasil rekomendasi dari mesin pencari dengan hasil yang dianggap ideal. Metrik ini mengukur seberapa baik urutan hasil pencarian yang diberikan sesuai dengan urutan hasil ideal, dengan memberikan bobot lebih pada item yang ditempatkan lebih tinggi dalam daftar hasil.

Kami menggunakan NDCG untuk mengevaluasi seberapa dekat urutan hasil pencarian dari sistem yang diuji dengan urutan hasil yang dianggap optimal, yang diambil dari Google Shopping.

NDCG dapat dirumuskan sebagai berikut:

$$nDCG_k = \frac{DCG_k}{IDCG_k}$$

$$DCG_k = \sum_{i=1}^k \frac{rel_i}{log_2(i+1)}$$

$$IDCG_k = \sum_{i=1}^k \frac{rel_i}{log_2(i+1)}$$

Di mana:

- $i$  adalah posisi.
- $k$  adalah jumlah total sampel yang dipertimbangkan dalam evaluasi.
- $rel_i$  adalah relevansi hasil yang dinilai pada posisi ke- $i$ .
- IDCG adalah *ideal discounted cumulative gain*.
- DCG adalah *discounted cumulative gain*.
- nDCG = Normalized DCG

**Accuracy.** Metriks ini digunakan untuk mengukur proporsi prediksi yang benar dari seluruh prediksi yang dibuat oleh model. Akurasi digunakan sebagai metrik umum untuk mengevaluasi kinerja model.

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

Di mana:

- $TP$  (*True Positive*): Jumlah prediksi positif yang benar.
- $TN$  (*True Negative*): Jumlah prediksi negatif yang benar.
- $FP$  (*False Positive*): Jumlah prediksi positif yang salah.
- $FN$  (*False Negative*): Jumlah prediksi negatif yang salah.

**Accuracy.** Mengukur proporsi prediksi positif yang benar dari semua prediksi positif yang dibuat oleh model. *Precision score* menunjukkan tingkat ketepatan dari prediksi model positif.

$$Precision = \frac{TP}{TP + FP}$$

**Recall.** Mengukur proporsi contoh positif yang benar-benar teridentifikasi dengan benar oleh model. Skor Recall menunjukkan tingkat sensitivitas model dalam menemukan semua contoh positif.

$$Recall = \frac{TP}{TP + FN}$$

**F1-Score.** Metriks ini menggabungkan presisi dan recall menjadi satu metrik yang memberikan gambaran keseimbangan antara keduanya. F1-score berguna ketika ada ketidakseimbangan kelas dalam dataset.

$$F1 score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

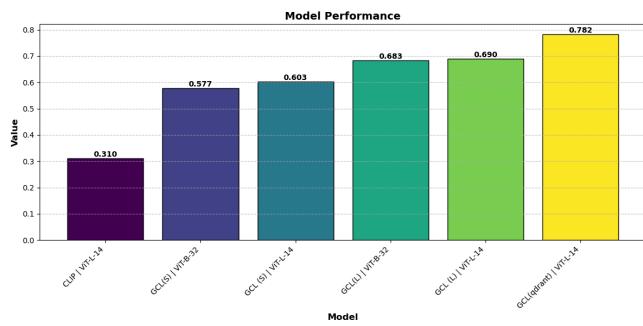
## 2) Kualitatif

**A/B Testing.** Kami melakukan A/B testing untuk membandingkan dua *search engine*, yaitu *search engine baseline* (tanpa metode trend) dan *search engine* (dengan implementasi NER dan trend). Kami melaksanakan A/B testing dengan mengumpulkan tanggapan secara online dan offline. Secara offline, kami melakukan survei terhadap individu di sekitar kawasan Universitas Gadjah Mada, yang merupakan pengguna mesin pencari dari berbagai e-commerce. Kami berfokus membandingkan tingkat relevansi dan trend. Responden akan diberikan 3 pertanyaan dengan masing masing pertanyaan akan diberikan 2 hasil *Search engine*, yaitu yang menggunakan Trend\_Based Query Expansion dan yang tidak menggunakaninya. Responden akan menilai hasil *Search Engine* masing-masing dengan 2 hal yang dijadikan pengukuran, yaitu skor relevansi dan skor trend. Skor relevansi adalah tingkat kesesuaian hasil *search engine* dengan kueri yang diberikan. Skor Trend adalah tingkat ke-up to date-an hasil *search engine* dengan trend yang sedang terjadi saat ini.

## III. HASIL DAN PEMBAHASAN

### A. Multimodal Search Engine

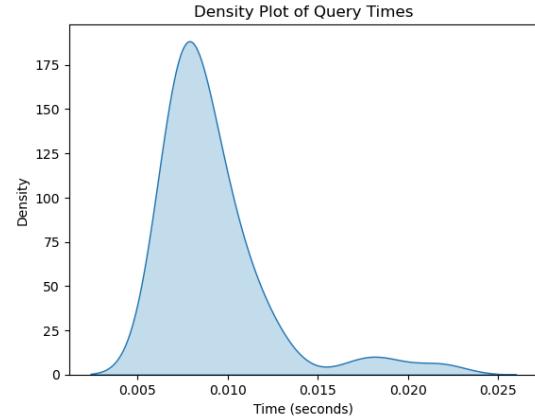
Untuk evaluasi performa mesin pencari multimodal, kami menggunakan metrik NDCG. Tujuan penggunaan metrik ini adalah karena NDCG mempertimbangkan bagaimana peringkat yang benar di bagian atas lebih relevan dibandingkan dengan peringkat di bagian bawah hasil pencarian. Dengan demikian, kami dapat memiliki metrik objektif yang juga memperhitungkan perilaku konsumen yang realistik. Dari hasil kami, kami menggunakan nDCG@100, yang berarti setiap hasil pencarian mengembalikan dan memberi peringkat pada 100 produk.



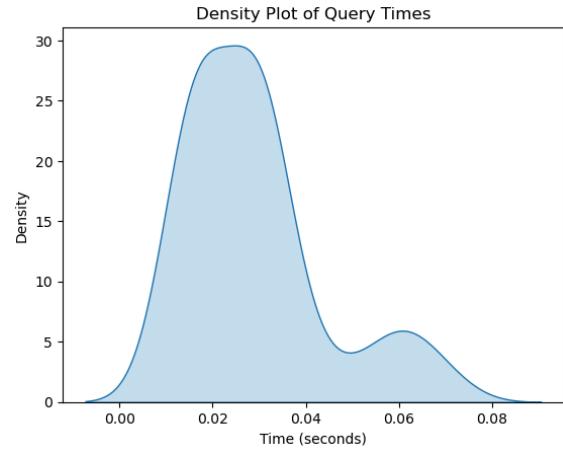
Gambar 5. Diagram Hasil Metriks nDCG

Bisa dilihat pada Gambar 5. bahwa model GCL menggunakan framework Qdrant memiliki performa terbagus dari metode lain pada dataset Marqo-GS. Karena performa model tersebut sudah relatif bagus jika dibandingkan dengan implementasi GCL lain, metode ini dapat digunakan sebagai dasar perkembangan fitur optimisasi tambahan.

**Latency.** Performa waktu yang didapatkan relatif baik. Latency untuk proses embedding query sampai dengan mendapatkan hasil adalah 34ms. Untuk proses pengambilan saja, membutuhkan waktu rata-rata 11 ms untuk mendapatkan hasil kueri.



Gambar 6. Plot Densitas *Query Times* (response time tanpa embedding).

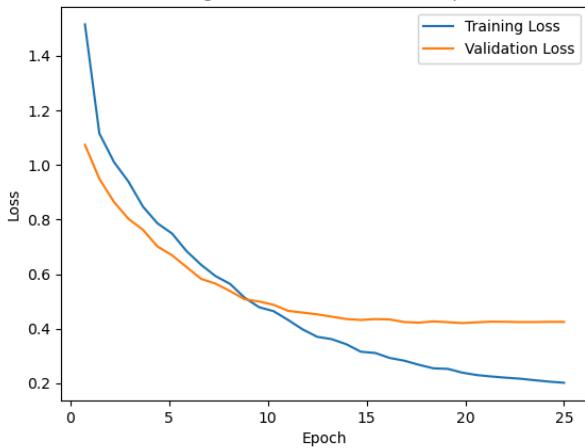


Gambar 7. Plot Densitas *Query Times* (response time dengan embedding).

**Kasus Negatif.** Selain itu, kami mengumpulkan 1.000 kueri yang tidak berhubungan dengan mode wanita, seperti senjata, kalender, speaker, dan sebagainya, serta memeriksa apakah mesin pencari kami dapat menyaring produk yang memang tidak ada dalam katalog. Dari semua kueri tersebut, 78,30% memiliki nilai kesamaan di bawah 1,6.

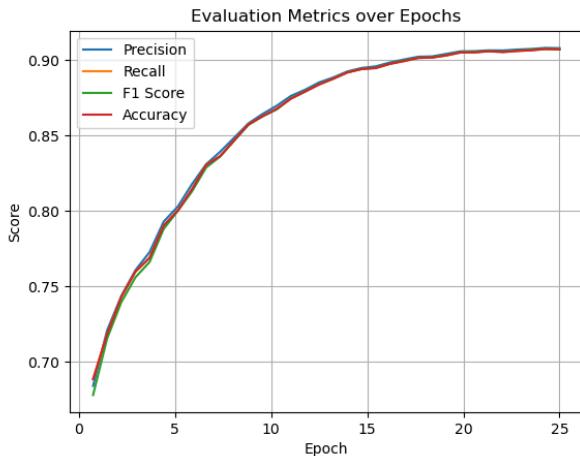
### B. Named Entity Recognition (NER)

**Hasil Fine Tuning.** Kami melaksanakan fine-tuning pada model QueryNER yang telah di-augmentasi dengan menggunakan data pelatihan NER hasil generasi sebelumnya. Data tersebut dibagi menjadi dua bagian, yaitu training dan evaluation split, dengan rasio 80:20. Proses fine-tuning dilakukan selama 25 epoch dengan batch size 16, memakan waktu total 22 menit 47 detik. Gambar 8 menunjukkan hasil training dan validation loss selama proses tersebut.



Gambar 8. Diagram Alir Fine Tuning Model QueryNER

Berdasarkan Gambar 8, terlihat bahwa pada awalnya baik training loss maupun validation loss cukup tinggi, namun keduanya menurun seiring dengan bertambahnya epoch. Training loss terus menunjukkan tren penurunan hingga epoch terakhir, sementara validation loss terlihat sudah mulai konvergen di sekitar angka 0.4 sejak epoch 18.



Gambar 9. Diagram Alir Fine Tuning Model QueryNER

Sejalan dengan tren training dan validation loss, Gambar 9. menunjukkan bahwa skor evaluasi precision, recall, F1 score, dan accuracy tidak banyak berubah di setiap epoch-nya. Dimulai dari sekitar angka 0.67, skor ini terus meningkat dan mulai konvergen di sekitar angka 0.90 sejak epoch 18. Hasil pada Gambar 8 dan 9 menunjukkan bahwa model sudah tidak melakukan optimasi weight secara signifikan setelah epoch 18 dan telah mencapai performa terbaiknya.

Tabel II. memperlihatkan perbedaan mencolok antara model QueryNER baseline (tanpa fine-tuning) dan model yang telah melalui proses fine-tuning. Pada data validasi, model baseline memiliki nilai loss sebesar 5.972, sementara model fine-tuned hanya memiliki nilai loss sebesar 0.425. Selain itu, metrik evaluasi seperti akurasi, presisi, recall, dan skor F1 juga menunjukkan perbedaan yang sangat signifikan. Rata-rata nilai metrik pada model fine-tuned

mencapai 0.907, sebesar 3 kali lipat lebih tinggi dibandingkan dengan model baseline. Hal ini menunjukkan bahwa fine-tuning secara substansial meningkatkan efektivitas model QueryNER.

TABEL II. PERFORMA MODEL QUERYNER BASELINE DAN FINE TUNED

Metriks Evaluasi	Model Baseline QueryNER	Model Fine Tuned QueryNER
Loss	5.972	<b>0.425</b>
Accuracy	0.307	<b>0.907</b>
Precision	0.394	<b>0.908</b>
Recall	0.307	<b>0.907</b>
F1 Score	0.312	<b>0.907</b>

Untuk mendapatkan pemahaman yang lebih baik mengenai seberapa besar fine-tuning meningkatkan kinerja model, kami memberikan contoh pada kueri berikut:

#### *"Affordable chic gowns for formal scene"*

Kami melakukan pengenalan *entity name* pada kueri ini menggunakan model QueryNER *baseline* dan model QueryNER *tuned*. Hasil pengenalan entitas tersebut disajikan dalam Tabel III.

TABEL III. STUDI KASUS MODEL QUERYNER BASELINE DAN FINE TUNED

Token	Baseline Model		Fine Tuned Model	
	Entity Name	Score	Entity Name	Score
affordable	modifier	0.93699	price	0.99592
chic	creator	0.31213	modifier	0.80477
gowns	core_product_type	0.99993	core_product_type	0.99993
formal	modifier	0.99986	occasion	0.99980
scene	modifier	0.99946	occasion	0.97604

Dalam membandingkan kinerja kedua model, terlihat adanya perbedaan hasil pengenalan *entity name*. Misalnya, istilah “*affordable*” yang awalnya diklasifikasikan sebagai *modifier* umum dengan skor 0.936992, namun model yang telah disesuaikan mengidentifikasinya dengan akurat sebagai terkait dengan *price* (harga), dengan skor yang meyakinkan yaitu 0.995923. Demikian juga, “*chic*” awalnya ditandai sebagai *creator* dengan skor rendah 0.312134, setelah di-*fine tuning* model mengubahnya menjadi *modifier* dan meningkatkan skornya menjadi 0.804775.

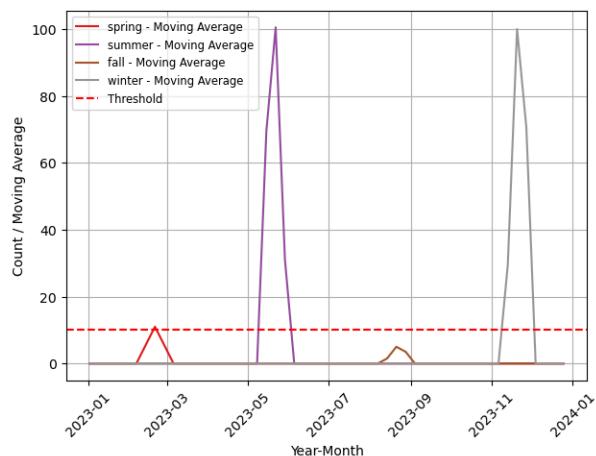
Istilah “*gowns*” tetap konsisten diidentifikasi sebagai *core\_product\_type* pada kedua model, tetapi model yang telah disesuaikan mencapai akurasi hampir sempurna dengan skor 0.999937 dibandingkan dengan 0.944794 pada model dasar. Untuk token “*formal*”, model yang di-*fine tuning* menunjukkan perbaikan dari klasifikasi *modifier* umum (0.999867) menjadi label yang lebih tepat secara kontekstual yaitu *occasion* (0.999805). Secara keseluruhan, model yang telah di-*fine tuning* menunjukkan perbaikan signifikan dalam presisi dan akurasi baik secara skor

maupun kontekstual yang menunjukkan pemahaman yang lebih mendalam dibandingkan dengan model *baseline*.

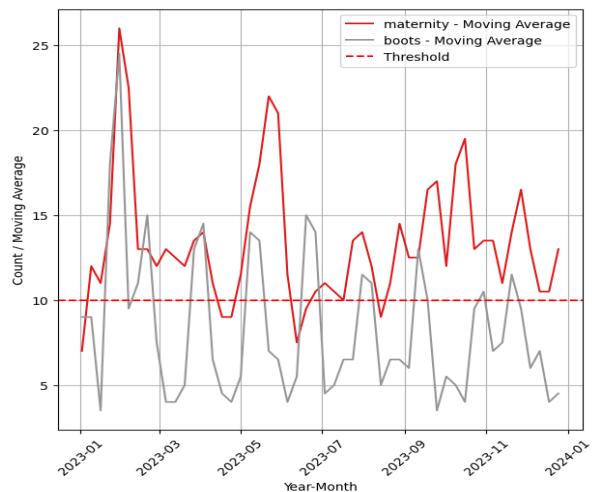
### C. Trend Analysis

Kami menganalisis berbagai tren dalam dataset kami. Hasilnya menunjukkan bahwa algoritma yang digunakan, yaitu fungsi GET\_TREND [Sebutkan pseudocode di atas], mampu mendeteksi tren yang ada.

Berdasarkan Gambar 10., terlihat adanya lonjakan nilai *moving average* yang disebabkan oleh peningkatan jumlah frekuensi dalam minggu tersebut. Gambar 10 juga menunjukkan bahwa tren yang terjadi dapat mengindikasikan musim yang sedang berlangsung



Gambar 10. Diagram Alir Fine Tuning Model QueryNER



Gambar 11. Diagram Alir Fine Tuning Model QueryNER

Berdasarkan Gambar 11, dapat dilihat bahwa tren juga bisa dideteksi tidak hanya untuk musim yang sedang terjadi. Kita juga bisa melihat apakah suatu barang sedang dicari banyak orang. Seperti gambar diatas terlihat bahwa ‘Boots’ cukup banyak di kueri sekitar bulan Februari. Menandakan barang tersebut sedang banyak dicari di bulan tersebut.

TABEL IV. DATA GOOGLE TREND

TimeStamp	Topik	Score
1/8/2024	Imane Khelif	500
1/8/2024	Algeria	200
31/7/2024	Katie Ledecky	500
31/7/2024	Real Madrid	200
31/7/2024	NFL	200

Data yang diambil dari Google Trends memiliki format yang berbeda. Dengan format ini, informasi mengenai topik yang sedang tren setiap hari bisa langsung diperoleh. Hal ini mengeliminasi kebutuhan untuk melakukan pemrosesan seperti pada data **Query History** dan data **hasil scrapping dari X (Twitter)**

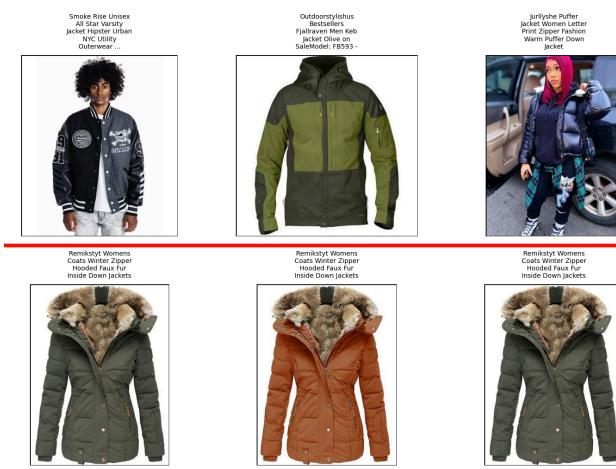
Dengan menganalisis *moving average* yang dihitung dari frekuensi pencarian setiap minggu, kita dapat mengetahui apa yang banyak dicari oleh orang. Mengetahui tren pencarian ini juga dapat merepresentasikan apa yang sedang terjadi, seperti musim yang berlangsung. Informasi ini bisa dimanfaatkan untuk meningkatkan relevansi produk dalam hasil mesin pencari dengan menyesuaikannya dengan apa yang sedang banyak dicari dan terjadi.

### D. Search Engine w/ Trend



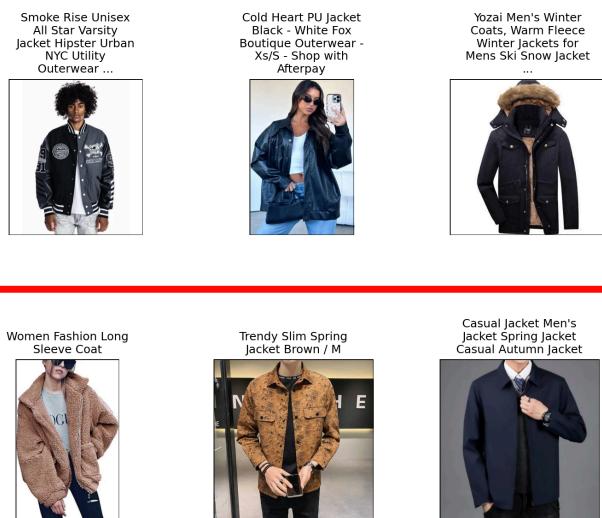
Gambar 12.

Search Engine w/ Google Trends (Olympic)



Gambar 13.

Search Engine w/ Query History (Jacket)



Gambar 14.

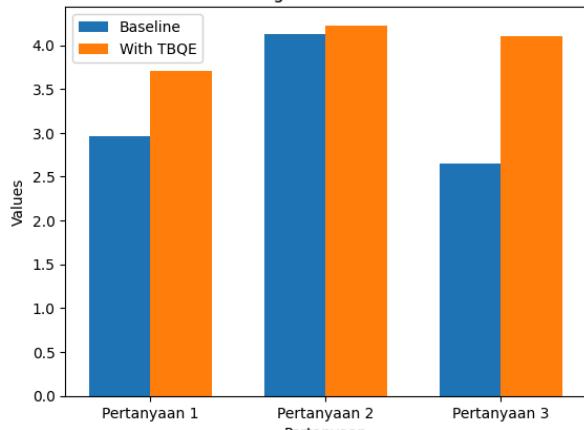
Search Engine w/ Twitter (X) Trends (Jacket)

Gambar 12, 13, dan 14 menunjukkan pengaruh perluasan tren terhadap setiap kueri, dengan menampilkan tiga produk teratas untuk sebuah kueri. Tiga gambar di atas menunjukkan hasil tanpa perluasan tren, sementara tiga gambar di bawah menunjukkan tiga peringkat menggunakan perluasan tren. Dapat dilihat bahwa konteks waktu sangat memengaruhi hasilnya. Misalnya, untuk peralatan Olimpiade, akan direkomendasikan merchandise Olimpiade karena merupakan topik yang sedang tren.

Untuk Query History, pada 20 November 2023, produk musim dingin sedang tren, sehingga yang direkomendasikan adalah jaket musim dingin. Pada Trend Twitter, karena kami mengumpulkan data berdasarkan cuitan yang sedang tren, maka rekomendasi baru yang diberikan lebih bergensi.

#### E. Search Engine w/ Trend

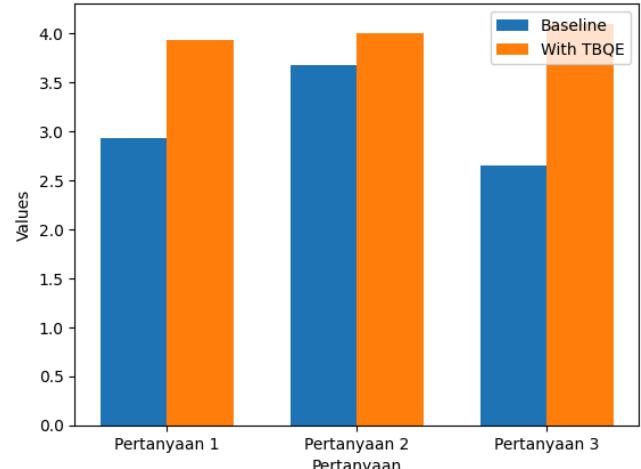
Perbandingan Model - Relevansi



Gambar 15.

Hasil Skor Relevansi

Perbandingan Model - Trend



Gambar 16. Hasil Skor Trend

Berdasarkan analisa data diatas, model TBQE memiliki relevansi yang lebih baik pada setiap pertanyaan. Model TBQE juga memiliki nilai lebih tinggi pada setiap pertanyaan. Sehingga dapat disimpulkan bahwa secara keseluruhan model TBQE memiliki kemampuan untuk mengenali tren saat ini lebih baik dari pada model Baseline.

\*TBQE:Trend-Based Query Expansion

#### IV. CONCLUSION

Sistem pencarian multimodal memberikan fleksibilitas dan kemudahan bagi pengguna untuk mencari produk melalui teks, gambar, atau kombinasi keduanya, memastikan pengalaman pencarian yang lebih efisien. Dengan penggunaan Qdrant lokal, performa pencarian menjadi cepat dan responsif, sementara implementasi *max similarity* meningkatkan akurasi hasil pencarian. Selain itu, penggunaan Named Entity Recognition (NER) memberikan wawasan mendalam mengenai produk dengan mengidentifikasi entitas penting seperti jenis produk, material, dan warna, sehingga hasil pencarian lebih terfokus dan sesuai preferensi pengguna. Sistem ini juga mampu mengenali tren terkini dalam pencarian produk, memastikan informasi yang *up-to-date* dan relevan dengan kebutuhan pasar yang dinamis.

#### DAFTAR PUSTAKA

- [1] A. Thompson, "Digital 2024: 5 billion social media users," *We Are Social UK*, Jan. 31, 2024. <https://wearesocial.com/uk/blog/2024/01/digital-2024-5-billion-social-media-users/>
- [2] S. Chevalier, "Share Social Buyers Age Group Worldwide," *Statista*, Mar. 28, 2024. <https://www.statista.com/statistics/1273928/share-social-buyers-age-group-worldwide/> (accessed Aug. 06, 2024).
- [3] M. Julianti, H. Mudjianto, P. Nagatitha, L. Luwiha, and W. Astuti, "TikTok's Social Media Marketing Gives Strong

- Impact on Product Sales Increase,” *Journal of Economic, Religious, and Entrepreneurship (JoERE)*, vol. 1, no. 1, pp. 31–38, Jun. 2023, doi: 10.60046/joere.v1i1.24
- [4] W. Wei, S. Li, S. Okada, and K. Komatani, “Multimodal User Satisfaction Recognition for Non-task Oriented Dialogue Systems,” in *Proceedings of the 2021 International Conference on Multimodal Interaction*, Oct. 2021. Accessed: Aug. 05, 2024. [Online]. Available: <http://dx.doi.org/10.1145/3462244.3479928>
- [5] A. J. Soto, P. Przybyla, and S. Ananiadou, “Thalia: semantic search engine for biomedical abstracts,” *Bioinformatics*, vol. 35, no. 10, pp. 1799–1801, Oct. 2018, doi: 10.1093/bioinformatics/bty871.
- [6] M. D. Lieberman, H. Samet, J. Sankaranarayanan, and J. Sperling, “STEWARD,” in *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, Nov. 2007. Accessed: Aug. 05, 2024. [Online]. Available: <http://dx.doi.org/10.1145/1341012.1341045>
- [7] J. Kim, S. So, H.-J. Lee, J. C. Park, J. Kim, and H. Lee, “DigSee: disease gene search engine with evidence sentences (version cancer),” *Nucleic Acids Research*, vol. 41, no. W1, pp. W510–W517, Jun. 2013, doi: 10.1093/nar/gkt531.
- [8] M. Khabsa, P. Treeratpituk, and C. L. Giles, “AckSeer,” in *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, Jun. 2012. Accessed: Aug. 05, 2024. [Online]. Available: <http://dx.doi.org/10.1145/2232817.2232852>
- [9] T. Zhu, M. C. Jung, and J. Clark, “Generalized Contrastive Learning for Multi-Modal Retrieval and Ranking,” *arXiv.org*, Apr. 12, 2024. <https://arxiv.org/abs/2404.08535>
- [10] OpenAI *et al.*, “GPT-4 Technical Report,” *arXiv.org*, Mar. 15, 2023. <https://arxiv.org/abs/2303.08774>
- [11] A. Radford *et al.*, “Learning Transferable Visual Models From Natural Language Supervision,” *arXiv.org*, Feb. 26, 2021. <https://arxiv.org/abs/2103.00020>
- [12] C. Palen-Michel, L. Liang, Z. Wu, and C. Lignos, “QueryNER: Segmentation of E-commerce Queries,” *arXiv.org*, May 15, 2024. <https://arxiv.org/abs/2405.09507>
- [13] C. K. Reddy *et al.*, “Shopping Queries Dataset: A Large-Scale ESCI Benchmark for Improving Product Search,” *arXiv.org*, Jun. 14, 2022. <https://arxiv.org/abs/2206.06588>
- [14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *arXiv.org*, Oct. 11, 2018. <https://arxiv.org/abs/1810.04805>