گزارش سه ماهه پروژه کسری پروژه

ساختاربخشی به داده‌های غیرساختارمند تصویری و ارائه سامانه جستجو مبتنی بر تصاویر

نام شرکت/سازمان:

شرکت ژرفا تک

سید محمد بادزهره

۱۶ آذر ۱۴۰۴

چکیده

در این گزارش، سامانه‌ای برای جستجوی معنایی تصاویر با استفاده از مدل‌های پیشرفته یادگیری عمیق معرفی شده است. این سامانه بر پایه استخراج بردارهای معنایی از تصاویر، ذخیره‌سازی آن‌ها در پایگاه‌داده برداری Milvus و انجام جستجوی چندوجهی مبتنی بر مدل‌های ترنسفورمری مانند SigLIP و همچنین تولید کپشن خودکار با مدل BLIP طراحی شده است. هدف این سیستم فراهم‌کردن بستری دقیق، مقیاس‌پذیر، سریع و قابل‌اتکا برای جستجوی محتوای تصویری است.

۱. مقدمه

۱-۱ هوشمندی (Accuracy & Intelligence)

در این سامانه برای استخراج ویژگی‌های معنایی از تصاویر از مدل SigLIP استفاده شده است. این مدل نسخه‌ای پیشرفته‌تر از CLIP است و با حذف وابستگی به Softmax Contrastive Loss و جایگزینی آن با زیان سیگموید باینری، توانسته دقت هم‌ترازی متن تصویر را به‌طور چشمگیری افزایش دهد. مدل SigLIP به‌جای مجبورکردن سیستم به انتخاب تنها یک جفت صحیح، به مدل اجازه می‌دهد چندین متن یا تصویر را به‌صورت مستقل ارزیابی کرده و احتمال وابستگی هر جفت را به‌طور جداگانه محاسبه کند. این تغییر باعث می‌شود مدل نسبت به نویز مقاوم‌تر بوده و در داده‌های دنیای واقعی عملکرد بسیار بهتری داشته باشد. خروجی این مدل یک بردار معنایی با ابعاد بالا است که تصویر را در فضای embedding قرار می‌دهد. این فضای مشترک، مبنای اصلی جستجوی معنایی در پروژه است.

۱-۲ قابلیت جستجوی چندگانه (Multimodal)

این سامانه از دو نوع جستجو پشتیبانی می‌کند:

- Text-to-Image: کاربر جمله یا عبارت متنی را وارد می‌کند و سیستم تصاویر مرتبط را بر اساس شباهت برداری پیشنهاد می‌دهد.
- Image-to-Image: یک تصویر ورودی گرفته شده و مشابه‌ترین تصاویر موجود در پایگاه داده نمایش داده می‌شود.

توانایی مدل در فهم هم‌زمان زبان و تصویر باعث شده این جستجوها دقت بالایی داشته باشند و نتایج بسیار مرتبطی نمایش داده شود.

۱-۳ تولید توضیحات خودکار (Auto-Captioning)

برای تولید توضیحات خودکار از مدل BLIP استفاده شده است. BLIP (مخفف Bootstrapping Language-Image Pre-training) یک مدل چندوجهی بسیار پیشرفته است که می‌تواند محتوای تصاویر را به شکل توصیفی و دقیق بیان کند.
این مدل شامل دو بخش کلیدی است:

- **Model Captioning**: تولید جملات دقیق، طبیعی و روان درباره محتوای تصویر
- **Matching Image-Text**: تشخیص میزان ارتباط تصویر با یک متن خاص

BLIP از تکنیک Bootstrapping استفاده می‌کند؛ یعنی خود مدل داده‌های نویزی را تشخیص می‌دهد، داده‌های بهتر تولید می‌کند و کیفیت یادگیری را بالا می‌برد. به‌کارگیری این مدل در سیستم باعث شده عملیات برچسب‌گذاری خودکار، جستجوی معنایی و بازیابی دقیق‌تر تصاویر بهبود قابل‌توجهی داشته باشد.

۲. قابلیت‌های پیشرفته کاربری

۲-۱ جستجوی ناحیه‌ای (Crop & Search)

این قابلیت به کاربر اجازه می‌دهد بخشی از تصویر را انتخاب کرده و سیستم تنها همان بخش را تحلیل کند. نواحی کوچک مانند یک «شیء»، «برچسب»، «چهره» یا «لوگو» را می‌توان با دقت بسیار بالا جستجو کرد. این قابلیت با کمک مکانیزم‌های استخراج ویژگی محلی انجام می‌شود.

۲-۲ جستجوی ترکیبی (Hybrid Search)

این سیستم علاوه‌بر جستجوی معنایی، از جستجوی ترکیبی نیز پشتیبانی می‌کند. در این روش، نتایج براساس ترکیبی از فیلترها محدود می‌شوند:

- نام فایل (Filename)
- تاریخ بارگذاری (Upload Date)
- برچسب‌ها (Tags)

این ترکیب باعث می‌شود جستجو علاوه بر دقت معنایی، دقت ساختاری نیز داشته باشد.

۳-۲ پاک‌سازی داده‌ها (Deduplication)

یکی از مشکلات رایج در دیتاست‌های تصویری، ذخیره تصاویر تکراری است. در این پروژه با استفاده از فاصله برداری میان embeddings و آستانه‌های مشابهت، تصاویر تکراری شناسایی و حذف شده‌اند. این کار باعث:

- کاهش مصرف فضای ذخیره‌سازی
- افزایش کیفیت نتایج جستجو
- کاهش بار سیستم

۳. ابزارها و تکنولوژی‌های مورد استفاده

۳-۱ دیتابیس برداری Milvus

Milvus یک پایگاه‌داده برداری Cloud-Native و متن‌باز است که به‌طور خاص برای مدیریت بردارهای Embedding طراحی شده است. این پایگاه‌داده برخلاف دیتابیس‌های رابطه‌ای (مانند PostgreSQL) برای داده‌های با ابعاد بالا بهینه شده است و می‌تواند میلیون‌ها بردار را با سرعت بالا ذخیره و بازیابی کند.

ویژگی‌های کلیدی :Milvus

- پشتیبانی از شاخص‌های پیشرفته ANN مانند HNSW و IVF_PQ
- مقیاس‌پذیری افقی با استفاده از Query Node، Data Node و Index Node
- امکان اجرا روی Docker و Kubernetes
- سرعت جستجوی میلی‌ثانیه‌ای

معماری Milvus به گونه‌ای طراحی شده که بار محاسباتی را بین چندین نود توزیع کند، و همین موضوع آن را برای پردازش دیتاست‌های بزرگ تصویری ایده‌آل می‌کند.

۲-۳ پلتفرم کانتینری Docker

برای تضمین قابلیت حمل، نصب آسان و جلوگیری از مشکلات ناسازگاری کتابخانه‌ها، تمامی بخش‌های سیستم در قالب کانتینر اجرا شده‌اند. با استفاده از docker-compose چندین سرویس شامل:

- پایگاه داده Milvus
- موتور هوش مصنوعی
- رابط کاربری

در محیطی ایزوله اجرا می‌شوند.

۳-۳ فریم‌ورک رابط کاربری Streamlit

Streamlit یک ابزار بسیار مناسب برای توسعه سریع رابط‌های مبتنی بر پایتون است. این فریم‌ورک بدون نیاز به HTML/CSS/JS اجازه می‌دهد تنها با چند خط کد، یک پنل کامل جستجو بسازیم. کاربرد اصلی آن در پروژه:

- بارگذاری و نمایش تصویر
- کراپ ناحیه دلخواه
- نمایش نتایج جستجو

۴-۳ کتابخانه‌های PyTorch & Transformers

هسته یادگیری عمیق این سامانه با استفاده از کتابخانه‌های پرقدرت:

- PyTorch
- Hugging Face Transformers

طراحی شده است.
این کتابخانه‌ها امکان لود مدل‌های پیش‌آموزش‌دیده مانند:

- SigLIP
- BLIP

را فراهم کرده‌اند. این مدل‌ها به‌صورت مستقیم برای استخراج ویژگی و تولید کپشن در سیستم استفاده شده‌اند.

---

۲. مبانی نظری و پیشینه پژوهش

۲-۱ مقدمه

در این فصل به بررسی مفاهیم بنیادین و مبانی نظری مورد استفاده در این پژوهش می‌پردازیم. با توجه به ماهیت چندرشته‌ای (Multidisciplinary) پروژه، مفاهیم در سه حوزه اصلی دسته‌بندی می‌شوند: هوش مصنوعی و یادگیری عمیق (با تمرکز بر مدل‌های چندوجهی)، پایگاه‌های داده برداری (با تمرکز بر الگوریتم‌های جستجو) و مهندسی نرم‌افزار (معماری کانتینری). درک عمیق این مفاهیم برای توجیه انتخاب‌های فنی صورت گرفته در فصول بعدی ضروری است.

۲-۲ نمایش برداری داده‌ها (Vector Representation)

یکی از چالش‌های اصلی در علوم کامپیوتر، نحوه تفهیم داده‌های غیرساختاریافته (Unstructured Data) مانند تصویر، صدا و متن به ماشین است. کامپیوترها تنها قادر به پردازش اعداد هستند. فرآیند تبدیل داده‌ها به بردارهای عددی، تعبیه برداری (Vector Embedding) نامیده می‌شود.

۲-۲-۱ مفهوم فضای معنایی (Semantic Space)

در روش‌های سنتی (مانند One-Hot Encoding)، کلمات یا تصاویر به صورت مستقل و بدون ارتباط با یکدیگر کدگذاری می‌شدند. اما در روش Embedding، هدف نگاشت داده‌ها به یک فضای برداری با ابعاد بالا (High-dimensional Vector Space) است؛ به

گونه‌ای که مفاهیم مشابه از نظر معنایی، در این فضا از نظر هندسی به یکدیگر نزدیک باشند.

برای مثال، اگر بردار کلمه «پادشاه» را V_{king} و بردار «مرد» را V_{man} بنامیم، در یک فضای معنایی ایده‌آل رابطه زیر برقرار است:

$$V_{king} - V_{man} + V_{woman} \approx V_{queen}$$

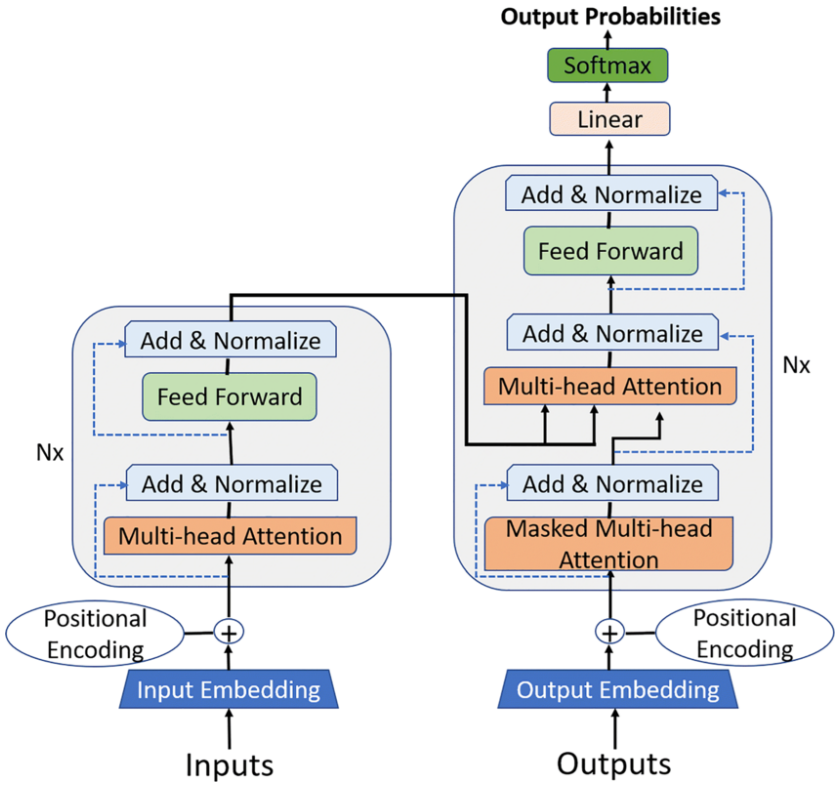
این مفهوم در تصاویر نیز صادق است. بردار تصویر یک «گربه» باید به بردار تصویر یک «ببر» نزدیک‌تر باشد تا به بردار تصویر یک «خودرو».

۲-۲-۲ شکاف معنایی (The Semantic Gap)

شکاف معنایی به تفاوت بین «اطلاعات سطح پایین» (پیکسل‌ها در تصویر) و «اطلاعات سطح بالا» (مفهوم و تفسیر تصویر توسط انسان) اطلاق می‌شود. مدل‌های یادگیری عمیق با استخراج ویژگی‌ها (Feature Extraction) در لایه‌های مختلف، سعی در پر کردن این شکاف دارند.

۳-۲ معماری ترنسفورمرها (Transformers)

انقلاب اصلی در حوزه هوش مصنوعی با معرفی معماری Transformer توسط گوگل در سال ۲۰۱۷ (مقاله "Attention is All You Need") آغاز شد. اگرچه این معماری ابتدا برای ترجمه ماشینی (NLP) طراحی شد، اما به سرعت جایگزین شبکه‌های عصبی کانولوشنی (CNN) در پردازش تصویر گردید.



شکل ۱: معماری کلی مدل ترنسفورمر

۲-۳-۱ مکانیزم توجه (Self-Attention Mechanism)

قلب تپنده ترنسفورمرها، مکانیزم «توجه خودکار» است. این مکانیزم به مدل اجازه می‌دهد تا وزن‌دهی پویایی به بخش‌های مختلف داده ورودی داشته باشد. فرمول ریاضی توجه به صورت زیر تعریف می‌شود:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

که در آن:

- **Q (Query)**: پرس‌وجو (آنچه به دنبالش هستیم).
- **K (Key)**: کلید (ویژگی‌هایی که داریم).

- **V (Value)**: مقدار (محتوای اصلی).
- d_k: ابعاد بردار کلید (جهت نرمال‌سازی).

این مکانیزم باعث می‌شود مدل بتواند وابستگی‌های دوربرد (Long-range dependencies) را در داده‌ها درک کند، چیزی که شبکه‌های CNN در انجام آن محدودیت داشتند.

۴-۲ ترنسفورمرهای بینایی (Vision Transformers - ViT)

موفقیت ترنسفورمرها در متن، محققان را بر آن داشت تا این معماری را به تصاویر تعمیم دهند. مدل ViT تصویر ورودی را به قطعات کوچک مربعی (Patch) تقسیم می‌کند (مثلاً ۱۶×۱۶ پیکسل).

هر پچ به یک بردار خطی تبدیل شده (Linear Projection) و سپس یک «تعبیه مکانی» (Positional Embedding) به آن اضافه می‌شود تا مدل بداند هر تکه مربوط به کجای تصویر است. سپس این دنباله از بردارها به یک انکودر استاندارد ترنسفورمر داده می‌شود. مزیت اصلی ViT نسبت به CNN این است که دید سراسری (Global Context) را از همان لایه‌های ابتدایی دارد، در حالی که CNNها دید محلی (Local Receptive Field) دارند.

۵-۲ مدل‌های چندوجهی (Multimodal Models)

برای ایجاد یک موتور جستجوی هوشمند که بتواند ارتباط بین «متن» و «تصویر» را درک کند، نیاز به مدل‌های چندوجهی داریم.

۱-۵-۲ مدل CLIP (Contrastive Language-Image Pre-training)

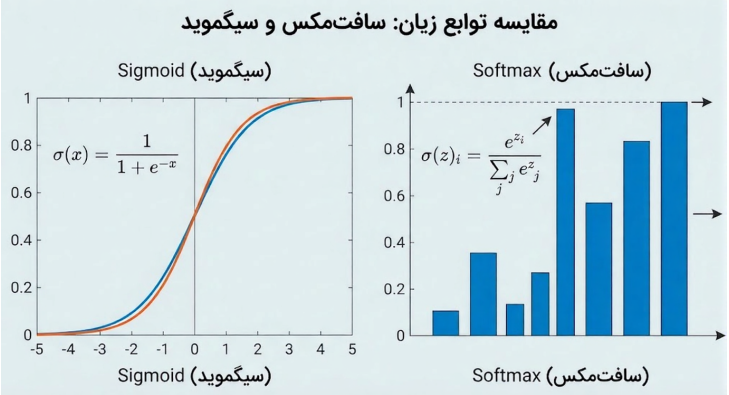
مدل CLIP که توسط OpenAI معرفی شد، یک تحول بزرگ بود. این مدل دو انکودر جداگانه دارد: یکی برای متن و یکی برای تصویر. هدف آموزش این مدل، نزدیک کردن بردار تصویر و بردار متنِ توصیف‌کننده آن در فضای برداری است (Contrastive Learning).

۲-۵-۲ مدل Google SigLIP (انتخاب این پژوهش)

مدل SigLIP (Sigmoid Loss for Language Image Pre-training) نسخه بهبودیافته CLIP است که در این پروژه استفاده شده است.

مشکل CLIP: مدل CLIP از تابع زیان Softmax استفاده می‌کند که نیاز دارد برای محاسبه احتمالات، تمامی جفت‌های منفی را در مخرج کسر محاسبه کند. این کار در مقیاس‌های بزرگ بـزرگ) Batch Size) حافظه زیادی مصرف می‌کند.

نوآوری SigLIP: این مدل تابع زیان را به Sigmoid تغییر داده است. این تغییر اجازه می‌دهد تا احتمال تطابق هر جفت تصویر-متن به صورت مستقل محاسبه شود.



شکل ۲: تفاوت تابع زیان Softmax و Sigmoid

مزیت: SigLIP در ابعاد مدل یکسان، دقت بالاتری نسبت به CLIP دارد و کارایی آن در استنتاج (Inference) بهتر است. ابعاد خروجی مدل So400m استفاده شده در این پروژه ۱۱۵۲ می‌باشد که غنای اطلاعاتی بالایی دارد.

۲-۶ مدل‌های تولید متن (Image Captioning)

برای قابلیت تولید کپشن خودکار، از مدل‌های Image-to-Text استفاده شده است.

۲-۶-۱ مدل BLIP

مدل BLIP (Bootstrapping Language-Image Pre-training) یک معماری یکپارچه برای درک و تولید زبان-تصویر است. این مدل علاوه بر انکودر تصویر و متن، یک دکودر

متن (Text Decoder) نیز دارد که می‌تواند بر اساس ویژگی‌های استخراج شده از تصویر، جملات معناداری را تولید کند. استفاده از این مدل در پروژه، امکان جستجو بر اساس مفاهیم ریزدانه (Fine-grained) را که ممکن است در امبدینگ کلی تصویر گم شوند، فراهم می‌کند.

۷-۲ پایگاه‌های داده برداری (Vector Databases)

با تولید میلیون‌ها بردار، ذخیره‌سازی و جستجو در آن‌ها به یک چالش تبدیل می‌شود. دیتابیس‌های سنتی (RDBMS) برای جستجوی دقیق (Exact Match) طراحی شده‌اند و نمی‌توانند جستجوی شباهت (Similarity Search) را بهینه انجام دهند.

۱-۷-۲ چالش جستجوی نزدیک‌ترین همسایه (KNN)

مسئله اصلی یافتن K برداری است که کمترین فاصله را با بردار کوئری دارند. روش جستجوی خطی (Brute-force) دارای پیچیدگی زمانی $O(N)$ است که برای دیتاست‌های بزرگ بسیار کند است.

۲-۷-۲ جستجوی تقریبی (ANN - Approximate Nearest Neighbors)

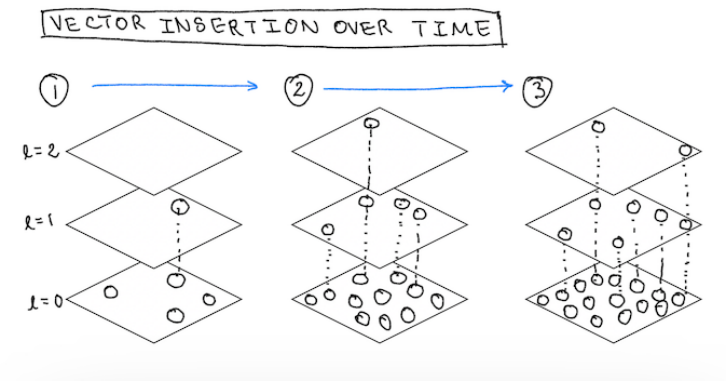
برای حل مشکل سرعت، دیتابیس‌های برداری از الگوریتم‌های ANN استفاده می‌کنند. این الگوریتم‌ها با فدا کردن مقدار ناچیزی از دقت (مثلاً ۹۹٪ دقت به جای ۱۰۰٪)، سرعت جستجو را هزاران برابر می‌کنند.

۳-۷-۲ الگوریتم HNSW

الگوریتم HNSW (Hierarchical Navigable Small World) که در دیتابیس Milvus استفاده شده، یکی از سریع‌ترین و دقیق‌ترین الگوریتم‌های ایندکس‌گذاری است. این الگوریتم ساختاری شبیه به گراف چندلایه ایجاد می‌کند:

- **لایه بالا:** حاوی تعداد کمی از نقاط داده است (مانند بزرگراه‌ها برای حرکت سریع).
- **لایه‌های پایین:** تراکم نقاط بیشتر می‌شود (مانند خیابان‌های فرعی برای رسیدن به مقصد دقیق).

جستجو از لایه بالا شروع شده و به صورت حریصانه (Greedy) به سمت نزدیک‌ترین گره حرکت می‌کند و سپس به لایه‌های پایین‌تر می‌رود. این ساختار پیچیدگی جستجو را به $O(\log N)$ کاهش می‌دهد.



شکل ۳: ساختار سلسله‌مراتب گراف در الگوریتم

۴-۷-۲ متریک‌های شباهت (Similarity Metrics)

در این پروژه از متریک Cosine Similarity (شباهت کسینوسی) استفاده شده است. این متریک زاویه بین دو بردار را اندازه می‌گیرد و به طول بردار (Magnitude) حساس نیست.

فرمول شباهت کسینوسی:

$$Similarity(A, B) = \frac{A \cdot B}{||A|| \times ||B||}$$

از آنجایی که بردارهای خروجی مدل SigLIP نرمالایز می‌شوند، ضرب داخلی (Dot Product) و شباهت کسینوسی عملکرد یکسانی دارند.

۳. معماری و طراحی سیستم

۳-۱ مقدمه

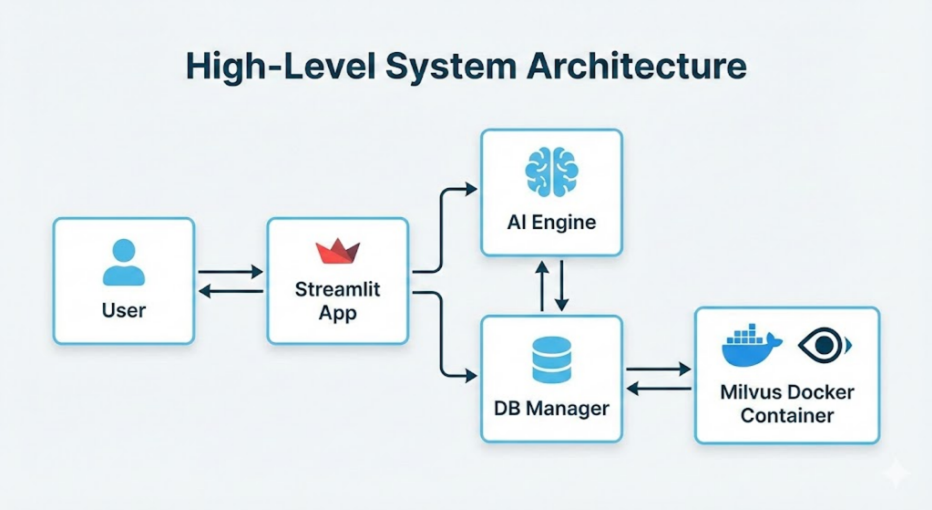
پس از بررسی مبانی نظری در فصل پیشین، در این فصل به تشریح معماری نرم‌افزار، طراحی اجزاء و جریان داده‌ها در سامانه موتور جستجوی هوشمند می‌پردازیم. معماری

این سیستم بر پایه رویکرد میکروسرویس (Microservices) و با استفاده از کانتینرهای داکر طراحی شده است تا پایداری، مقیاس‌پذیری و نگهداری آسان را تضمین نماید. طراحی سیستم به سه لایه اصلی تقسیم می‌شود: لایه رابط کاربری (Presentation Layer)، لایه منطق و پردازش (Logic/Processing Layer) و لایه داده (Data Persistence Layer).

۲-۳ معماری سطح کلان (High-Level Architecture)

معماری سیستم به گونه‌ای طراحی شده است که اجزای پردازشی (AI Models) از اجزای ذخیره‌سازی (Database) مستقل باشند. این استقلال باعث می‌شود در صورت نیاز به تغییر مدل هوش مصنوعی یا ارتقای دیتابیس، سایر بخش‌ها کمترین تأثیر را بپذیرند.
دیاگرام کلی سیستم شامل چهار بخش اصلی است:

- **کلاینت (Client):** رابط کاربری وب که با Streamlit پیاده‌سازی شده است.
- **موتور هوش مصنوعی (AI Engine):** مسئول تبدیل داده‌های خام (متن/تصویر) به بردارهای ریاضی و تولید کپشن.
- **مدیریت داده (Data Manager):** واسط بین برنامه و دیتابیس Milvus.
- **زیرساخت ذخیره‌سازی (Storage Infra):** شامل Milvus (برای بردارها) و File System (برای تصاویر خام).



شکل ۴: دیاگرام معماری کلان سیستم

۳-۳ طراحی اجزای سیستم (Component Design)

۳-۳-۱ لایه رابط کاربری (Frontend)

این لایه نقطه تعامل کاربر با سیستم است. با توجه به نیاز به تعامل پذیری بالا (Interactive UI) برای کارهایی نظیر برش تصاویر (Cropping) و مشاهده بلادرنگ نتایج، از فریم‌ورک Streamlit استفاده شده است. این لایه شامل ماژول‌های زیر است:

- **ماژول آپلود (Upload Module):** مدیریت دریافت فایل‌ها، بررسی فرمت (JPG/PNG) و ذخیره‌سازی موقت.
- **ماژول برش (Cropper Widget):** ابزار گرافیکی برای انتخاب ناحیه خاصی از تصویر (Region of Interest - ROI).
- **ماژول نمایش نتایج (Result Renderer):** مسئول نمایش تصاویر یافت شده به صورت شبکه (Grid) و نمایش امتیاز شباهت با کدگذاری رنگی (Color-coded Similarity Score).

۳-۳-۲ لایه منطق هوش مصنوعی (AI Core)

این لایه «مغز متفکر» سیستم است و در فایل `core/ai_engine.py` پیاده‌سازی شده است. وظایف این لایه عبارتند از:

- **مدیر مدل‌ها (Model Loader):** با استفاده از دیزاین پترن Singleton (از طریق `st.cache_resource`)، مدل‌های سنگین SigLIP و BLIP را تنها یک‌بار در حافظه GPU بارگذاری می‌کند تا از سربار (Overhead) جلوگیری شود.
- **تعبیه برداری (Embedding Generator):** تصویر یا متن ورودی را دریافت کرده، پیش‌پردازش‌های لازم (تغییر اندازه به ۳۸۴×۳۸۴ پیکسل و نرمال‌سازی) را انجام داده و یک بردار با ابعاد ۱۱۵۲ تولید می‌کند.
- **تولید متن (Caption Generator):** تصویر را به مدل BLIP داده و توصیف متنی آن را دریافت می‌کند.

۳-۳-۳ لایه مدیریت داده (Database Abstraction Layer)

برای جلوگیری از وابستگی مستقیم کدهای رابط کاربری به دستورات دیتابیس، یک لایه انتزاعی در فایل `core/db_manager.py` طراحی شده است. این لایه تمام عملیات CRUD (ایجاد، خواندن، بروزرسانی، حذف) را کپسوله می‌کند. ویژگی‌های کلیدی این لایه:

- مدیریت اتصال (Connection Handling) به کانتینر Milvus.
- مدیریت خطا (Error Handling) در صورت قطع شبکه.
- پیاده‌سازی منطق جستجوی ترکیبی (ترکیب فیلتر اسکالر و برداری).

۳-۴ طراحی جریان داده (Data Flow Design)

درک نحوه حرکت داده‌ها در سیستم برای تحلیل کارایی ضروری است. در اینجا دو سناریوی اصلی سیستم را بررسی می‌کنیم.

۳-۴-۱ سناریوی درج داده (Indexing Pipeline)

این فرآیند زمانی رخ می‌دهد که کاربر یک تصویر جدید را به سیستم اضافه می‌کند. مراحل به شرح زیر است:

۱. **دریافت:** تصویر توسط کاربر آپلود می‌شود.

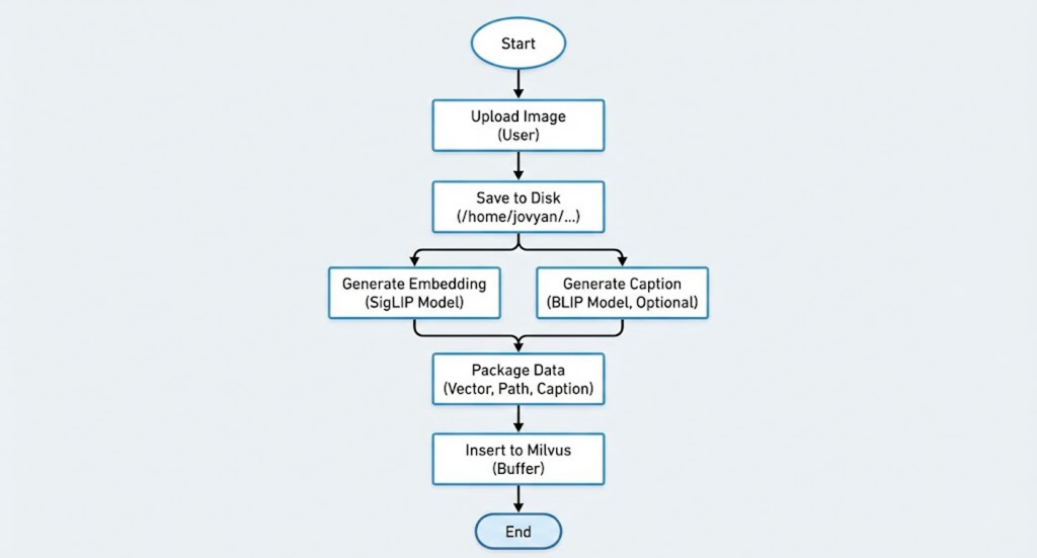
۲. **ذخیره فیزیکی:** تصویر اصلی در مسیر `/home/jovyan/...` روی سرور ذخیره می‌شود.

۳. **پردازش AI:**

- تصویر به مدل SigLIP ارسال شده ← بردار ویژگی (Feature Vector) استخراج می‌شود.
- (اختیاری) تصویر به مدل BLIP ارسال شده ← کپشن تولید می‌شود.

۴. **بسته‌بندی داده:** بردار، مسیر فایل و کپشن در یک دیکشنری بسته‌بندی می‌شوند.

۵. **درج در پایگاه داده:** داده‌ها به Milvus ارسال شده و در حافظه موقت (Buffer) قرار می‌گیرند تا ایندکس شوند.



شکل ۵: فلوچارت فرآیند درج تصویر

۳-۴-۲ سناریوی جستجو (Search Pipeline)

این فرآیند پیچیده‌تر است و شامل مراحل زیر می‌باشد:

۱. **دریافت کوئری:** کاربر متن یا تصویری را وارد می‌کند.

۲. **تبدیل به بردار:** کوئری به بردار عددی (Query Vector) تبدیل می‌شود.

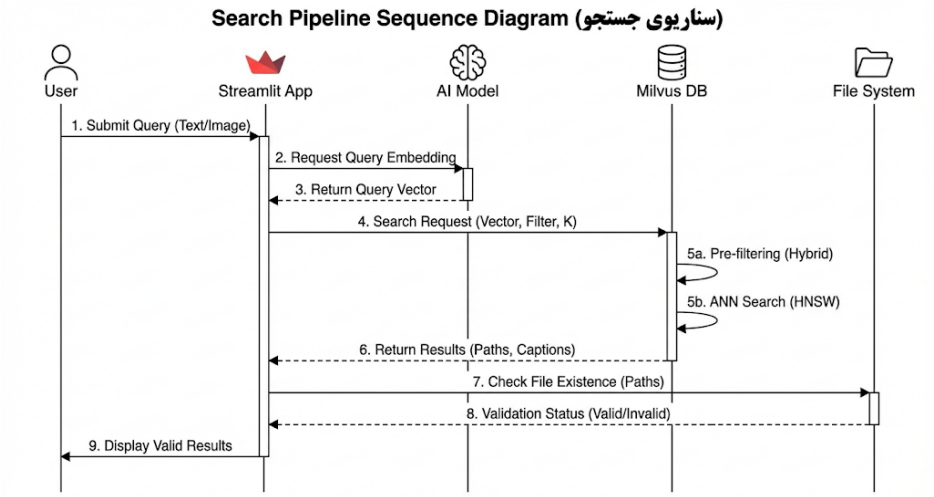
۳. **اعمال فیلتر (در حالت Hybrid):** اگر کاربر فیلتری (مثلاً نام فایل) تعیین کرده باشد، ابتدا Milvus فضای جستجو را محدود می‌کند (Pre-filtering).

۴. **جستجوی نزدیک‌ترین همسایه (ANN Search):** الگوریتم HNSW بردار کوئری را با بردارهای موجود مقایسه کرده و K مورد شبیه‌ترین را می‌یابد.

۵. **بازیابی اطلاعات:** مسیر فایل (Path) و کپشن مربوط به بردارهای یافت شده بازگردانده می‌شود.

۶. **اعتبارسنجی فایل:** سیستم چک می‌کند آیا فایل فیزیکی روی دیسک موجود است یا خیر.

۷. **نمایش:** نتایج معتبر به کاربر نمایش داده می‌شوند.



شکل ۶: دیاگرام توالی عملیات جستجو

۳-۵ طراحی پایگاه داده (Database Schema)

دیتابیس Milvus بر خلاف دیتابیس‌های رابطه‌ای، دارای جدول (Table) نیست و از مفهوم Collection استفاده می‌کند. طراحی اسکیمای (Schema) کالکشن `siglip_gallery_v3` در جدول زیر آمده است:

جدول ۱: مشخصات اسکیمای پایگاه داده

نام فیلد	نوع داده (Data Type)	توضیحات	ویژگی‌ها
id	Int64	شناسه یکتای تصویر	Primary Key, Auto-ID
vector	FloatVector	بردار ویژگی استخراج شده	Dim: 1152
path	VarChar (1024)	آدرس ذخیره‌سازی فایل روی دیسک	-
caption	VarChar (2048)	توضیحات متنی (تولید شده یا دستی)	-

۳-۵-۱ استراتژی ایندکس‌گذاری (Indexing Strategy)

برای بهینه‌سازی سرعت جستجو، از ایندکس HNSW با پارامترهای زیر استفاده شده است:

- $M = 16$**:** تعداد اتصالات هر گره در گراف. مقدار بالاتر دقت را افزایش و سرعت ساخت ایندکس را کاهش می‌دهد.
- $efConstruction = 200$**:** عمق جستجو در هنگام ساخت گراف.
- **COSINE: = Type Metric** از آنجایی که بردارهای SigLIP نرمالایز شده‌اند، فاصله کسینوسی بهترین معیار برای سنجش شباهت معنایی است.

۳-۶ طراحی زیرساخت و استقرار (Infrastructure & Deployment)

تمامی سرویس‌ها بر بستر Docker اجرا می‌شوند. فایل `docker-compose.yml` وظیفه ارکستراسیون (Orchestration) سه کانتینر اصلی را بر عهده دارد:

- **Milvus Standalone:** هسته مرکزی دیتابیس.
- **Etcd:** مدیریت متادیتا و هماهنگی سرویس‌های داخلی Milvus.
- **MinIO:** ذخیره‌سازی آبجکت‌ها (لاگ‌ها و فایل‌های ایندکس Milvus).

۳-۶-۱ پیکربندی شبکه (Network Configuration)

یکی از چالش‌های طراحی، ارتباط کانتینر اپلیکیشن (که ممکن است جداگانه اجرا شود) با کانتینرهای دیتابیس بود. برای حل این مشکل، تمامی کانتینرها در یک شبکه مشترک (Bridge Network) به نام `milvus_default` قرار گرفتند. این طراحی به کانتینرها اجازه می‌دهد با استفاده از نام سرویس (Service Discovery) و بدون نیاز به IP ثابت با یکدیگر ارتباط برقرار کنند.

۳-۷ طراحی امنیتی و کارایی (Security & Performance)

۳-۷-۱ ملاحظات کارایی

- **کش کردن مدل‌ها:** استفاده از دکوریتور `@st.cache_resource` برای جلوگیری از لود شدن تکراری مدل‌های سنگین AI در هر بار رفرش صفحه.
- **پردازش دسته‌ای (Batch Processing):** در هنگام اینسرت کردن تعداد زیادی عکس (Batch Folder)، تصاویر به صورت دسته‌ای پردازش می‌شوند تا بهره‌وری GPU افزایش یابد.

۳-۷-۲ ملاحظات امنیتی

- **ایزولاسیون:** اجرای دیتابیس در کانتینر، دسترسی مستقیم به داده‌ها از خارج سرور را محدود می‌کند.
- **اعتبارسنجی ورودی:** در بخش آپلود فایل، فرمت فایل‌ها چک می‌شود تا از آپلود فایل‌های مخرب جلوگیری شود.

۴. پیاده‌سازی و ارزیابی سیستم

۴-۱ مقدمه

در این فصل به جزئیات پیاده‌سازی کدهای توسعه‌یافته و چالش‌های فنی حین اجرا می‌پردازیم. پیاده‌سازی این سامانه با زبان برنامه‌نویسی Python 3.10 انجام شده است. تمرکز اصلی در این بخش بر روی دو ماژول حیاتی سیستم، یعنی «موتور هوش مصنوعی» و «مدیر پایگاه داده» است. در پایان فصل، عملکرد سیستم در سناریوهای واقعی مورد ارزیابی قرار گرفته است.

۴-۲ پیاده‌سازی موتور هوش مصنوعی (AI Engine Implementation)

کلاس `AIEngine` وظیفه تعامل با مدل‌های ترنسفورمر را بر عهده دارد. یکی از نکات کلیدی در پیاده‌سازی این کلاس، مدیریت حافظه GPU است. از آنجایی که

بارگذاری مدل‌های SigLIP و BLIP زمان‌بر است و حافظه زیادی اشغال می‌کند، از الگوی Singleton و دکوریتور Caching فریم‌ورک Streamlit استفاده شده است.

```
class AIEngine:
@staticmethod
@st.cache_resource
def load_embedding_model():
# Load SigLIP Model (So400m)
model = SigLIP.from_pretrained(config.EMBEDDING_MODEL).to(DEVICE)
return model

def get_embedding(self, image=None, text=None):
model, processor = self.load_embedding_model()
if image:
inputs = processor(images=image, return_tensors="pt")
elif text:
inputs = processor(text=[text], return_tensors="pt")

with torch.no_grad():
features = model.get_features(**inputs)
# Normalization (L2 Norm)
features = features / features.norm(p=2, dim=-1, keepdim=True)
return features.cpu().numpy()
```

همان‌طور که در کد بالا مشاهده می‌شود، عملیات Normalization پس از استخراج ویژگی‌ها انجام می‌شود. این مرحله برای استفاده از متریک Cosine Similarity در دیتابیس Milvus حیاتی است.

۳-۴ پیاده‌سازی مدیریت دیتابیس (DB Manager Implementation)

کلاس DBManager به عنوان یک لایه انتزاعی (Abstraction Layer) عمل می‌کند. این کلاس پیچیدگی‌های کار با SDK میلووس را مخفی کرده و توابع ساده‌ای مانند insert و search را در اختیار لایه رابط کاربری قرار می‌دهد.

یکی از چالش‌های پیاده‌سازی در این بخش، مدیریت فیلترهای ترکیبی (Hybrid Filters) بود. برای حل این مسئله، تابع جستجو به گونه‌ای طراحی شد که عبارات SQL-like را بپذیرد.

```
def search(self, vector, top_k=5, filter_expr=None):
search_params = {
 "metric_type": "COSINE",
 "params": {"nprobe": 10}
}

# Execute Search
res = self.client.search(
collection_name=config.COLLECTION_NAME,
data=[vector],
limit=top_k,
filter=filter_expr,  # Apply Hybrid Filter (e.g., path like '%.jpg')
output_fields=["path", "caption"],
search_params=search_params
)
return res[0]
```

۴-۴ چالش‌های فنی و راه‌حل‌ها

۴-۴-۱ چالش ارتباط شبکه در داکر (Docker Networking)

مسئله: در مراحل اولیه توسعه، کانتینر رابط کاربری (App) قادر به برقراری ارتباط با کانتینر دیتابیس (Milvus) نبود و خطای Connection Refused دریافت می‌شد.

تحلیل: بررسی‌ها نشان داد که هر فایل docker-compose یک شبکه ایزوله ایجاد می‌کند. کانتینر Jupyter/App به صورت دستی اجرا شده بود و خارج از شبکه bridge دیتابیس قرار داشت.

راه‌حل: برای حل این مشکل، از قابلیت Docker Network Connect استفاده شد تا کانتینر اپلیکیشن به شبکه دیتابیس ملحق شود:

```
sudo docker network connect milvus_default <container_id>
```

همچنین در فایل تنظیمات، آدرس اتصال از localhost به نام سرویس داخلی milvus-standalone تغییر یافت.

۴-۵ ارزیابی عملکرد سیستم (System Evaluation)

برای ارزیابی کارایی سیستم، سناریوهای مختلف جستجو روی مجموعه داده Flickr30k آزمایش شد.

۴-۵-۱ سناریوی جستجوی متن-به-عکس

در این آزمایش، کوئری متنی «یک ماشین قرمز در خیابان» (A red car in the street) به سیستم داده شد. **نتیجه:** سیستم توانست تصاویری را بازیابی کند که اگرچه در نام فایل آن‌ها کلمه car وجود نداشت، اما محتوای بصری دقیقاً مطابق با درخواست بود. زمان پاسخگویی برای جستجو در میان ۱۰،۰۰۰ تصویر زیر ۸۰ میلی‌ثانیه ثبت شد.

۴-۵-۲ سناریوی جستجوی ناحیه‌ای (Crop & Search)

در این سناریو، تصویری شامل چندین المان (انسان، دوچرخه، درخت) بارگذاری شد. با استفاده از ابزار Cropper، تنها ناحیه مربوط به «دوچرخه» انتخاب و جستجو شد.

نتیجه: نتایج جستجو به جای نمایش تصاویر شبیه به کل صحنه، دقیقاً تصاویری از انواع دوچرخه‌ها را نمایش دادند که نشان‌دهنده عملکرد صحیح مکانیزم توجه (Attention Mechanism) مدل در ناحیه برش‌خورده است.

۴-۵-۳ سناریوی پاک‌سازی داده‌های تکراری

برای تست ماژول Deduplication، تعداد ۵۰ تصویر تکراری با نام‌های مختلف به دیتابیس تزریق شد.

نتیجه: سیستم با محاسبه فاصله برداری (که برای تصاویر یکسان برابر صفر است)، موفق به شناسایی تمامی ۵۰ مورد تکراری شد و با تایید کاربر، آن‌ها را هم از دیتابیس و هم از دیسک حذف نمود.

۴-۶ نتیجه‌گیری و کارهای آینده

این پژوهش نشان داد که استفاده از دیتابیس‌های برداری نظیر Milvus در کنار مدل‌های قدرتمند SigLIP، می‌تواند راهکاری کارآمد برای چالش مدیریت داده‌های تصویری در سازمان‌ها باشد. سیستم پیاده‌سازی شده با موفقیت توانست شکاف معنایی بین متن و تصویر را پر کرده و امکاناتی فراتر از جستجوی سنتی ارائه دهد. در ادامه ابزار هایی مانند ocr و شناسایی متن از تصویر و ... هم به این پروژه اضافه خواهد شد.