پلتفرم هوشمند راوی (Ravi)

معماری فنی پیشرفته: مچینگ گروهی و اتوماسیون بات

- معماری ربات تلگرام (FSM, Middleware, Webhook)
- ارکستراسیون دقیق داده‌ها با n8n
- تکنولوژی الگوریتم خوشه‌بندی و مچینگ

۱. استراتژی محصول: مدل هیبریدی راوی

هسته مرکزی:

• ترکیب وب‌سایت (رزرو و پروفایل) + تلگرام (تعامل زنده).

جریان کاربر (User Flow):

• ورود از سایت -> نرمال‌سازی دیتا -> دعوت به گروه تلگرام.

• تست تعویق‌یافته (Deferred Testing) برای کاهش اصطکاک ورود.

۲. توسعه ربات راوی: زبان‌ها و ابزارها

زبان‌ها و فریم‌ورک‌ها:

• گزینه اول (پیشنهادی): Python + Aiogram (به دلیل قدرت بالا و سینک بودن با هسته AI).

• گزینه دوم: Node.js + Telegraf (سازگاری بالا با معماری میکروسرویس).

نقشه راه فنی (Technical Roadmap):

۱. معماری ماشین حالت (FSM) برای مدیریت حافظه.

۲. پیاده‌سازی Webhook برای سرعت Real-time.

۳. طراحی لایه امنیتی Middleware.

۴. اتصال به Redis برای کشینگ و مدیریت Session.

۳. معماری FSM و مدیریت حافظه (Redis)

۱. تعریف معماری FSM (ماشین حالت):

- مشکل: ربات‌ها به‌طور پیش‌فرض Stateless هستند (حافظه ندارند).
- راهکار: تعریف وضعیت (State) برای هر کاربر جهت تفسیر صحیح پیام.
- مثال در راوی: وضعیت `5WAITING_FOR_BIG` (انتظار برای تست) در مقابل `IN_GROUP_CHAT`.

۲. اتصال به Redis (مدیریت Session):

- سرعت: خواندن/نوشتن زیر ۵ میلی‌ثانیه (روی RAM) برای مدیریت هزاران کاربر همزمان.
- کاربردها: ذخیره State لحظه‌ای، مدیریت توکن‌های موقت، و سیستم اخطار (Strike System) با قابلیت TTL.

۴. ارتباط آنی و امنیت (Webhook & Middleware)

۳. پیاده‌سازی Webhook (دریافت لحظه‌ای):

• تفاوت با Polling: حذف تاخیر و کاهش فشار روی سرور (Push vs Pull).

• مزیت برای راوی: سازگاری عالی با n8n (اجرای ورک‌فلو دقیقاً در لحظه دریافت پیام).

۴. طراحی Middleware (لایه امنیتی):

• نقش: «نگهبان دروازه» قبل از رسیدن پیام به لاجیک اصلی.

• وظایف:

- فیلتر کردن اسپم (Rate Limiting).

- بررسی نوع پیام (حذف استیکرها یا فورواردها اگر مجاز نباشند).

- ایمن‌سازی ورودی‌ها (Sanitization) قبل از پردازش.

۵. چالش همزمانی (Race Condition)

چالش: کلیک همزمان روی دکمه عضویت دو گروه.

راهکار: قفل توزیع‌شده (Distributed Lock) با Redis.

```
async def join_group(user_id, group_id):
# ایجاد قفل با کلید منحصر به فرد کاربر
lock = redis.lock(f'user:{user_id}', timeout=(5
if not lock.acquire(blocking=False):
return 'Please wait'...
try:
# بررسی و کسر اعتبار در محیط ایزوله
if user.credits > :0
user.credits -= 1
await telegram.add_user(group_id, user_id)
finally:
lock.release()
```

۶. پایپ‌لاین داده‌ها در n8n (هسته هوشمند)

مدیریت جریان داده بین تلگرام و هسته هوشمند راوی:

۱. دریافت و نرمال‌سازی (Ingestion):

- گام ۱ (Webhook Trigger): دریافت Payload JSON خام از تلگرام.
- گام ۲ (Data Normalization): تمیزسازی متن و تبدیل به فرمت استاندارد DB.

۲. پردازش و هوش مصنوعی (Processing):

- گام ۳ (Routing): تفکیک هوشمند مسیر پردازش متن (NLP) از تصویر (MinIO).
- گام ۴ (AI Enrichment): اتصال به LLM جهت استخراج Sentiment و کلمات کلیدی.

۳. ذخیره‌سازی و اقدام (Storage & Action):

- گام ۵ (Vector Store): تبدیل ویژگی‌ها به وکتور و ذخیره در Postgres (pgvector).
- گام ۶ (Action): ارسال پاسخ نهایی یا انجام عملیات مدیریتی.

۷. تکنولوژی الگوریتم مچینگ (Clustering)

ابزارهای محاسباتی:

• زبان اصلی: Python (کتابخانه‌های Scikit-learn, Pandas, NumPy).

منطق ریاضی:

• Cosine Similarity: محاسبه زاویه بین «بردار کاربر» و «بردار ایونت».

• هرچه مقدار به ۱ نزدیک‌تر باشد، سازگاری روانی و رفتاری بیشتر است.

روش خوشه‌بندی:

• استفاده از K-Means Clustering برای گروه‌بندی ۴ تا ۶ نفره که مرکزیت آن‌ها (Centroid) نزدیک به ویژگی‌های ایونت باشد.

۸. پیچیدگی‌های پیاده‌سازی مچینگ

۱. مشکل شروع سرد(Cold Start):

•چالش: کاربر جدیدی که هیچ دیتایی ندارد چگونه مچ شود؟

• راهکار: استفاده از دیتای دموگرافیک (سن، شغل) به عنوان وزن اولیه تا زمان تکمیل تست.

۲. بهینه‌سازی چندگانه (Multi-Objective) :

•ما فقط به دنبال شباهت نیستیم؛ باید تنوع (Diversity) را هم حفظ کنیم.

• چالش کدنویسی: افزودن قید (Constraint) به الگوریتم K-Meansکه مثلاً در هر گروه حداقل ۲ جنسیت مخالف باشند.

۳. ابعاد بالا (High Dimensionality):

•مدیریت بردارهای ۱۵۳۶ بعدی (OpenAI Embeddings) و کاهش ابعاد برای افزایش سرعت پردازش.

الگوریتم مچینگ: چگونه گروه می‌سازیم؟

فرمول ریاضی: شباهت کسینوسی (Cosine Similarity)

- هدف: یافتن افرادی که جهت بردارهایشان در فضای ریاضی به هم نزدیک است.
- فرمول: محاسبه زاویه بین بردار شخص A و شخص B.

فرآیند کلاسترینگ (Clustering Pipeline):

1. واکشی تمام کاربران در انتظار (Pool).
2. نرمال‌سازی داده‌ها (Data Normalization).
3. اجرای الگوریتم K-Means یا DBSCAN:
   - این الگوریتم نقاط (کاربران) را در فضای چندبعدی پخش می‌کند.
   - دایره‌هایی (Cluster) می‌کشد تا گروه‌های ۴ نفره با کمترین فاصله از هم تشکیل شوند.
4. بررسی قیود سخت (Hard Constraints): مثل بازه سنی یا جنسیت.

۹. معماری زیرساخت (Infrastructure)

Database Layer:

- PostgreSQL + pgvector: دیتابیس اصلی و وکتوری.
- Redis: کشینگ پرسرعت و مدیریت صف (Message Broker).

Server Side:

- VPS با سیستم عامل Ubuntu 22.04.
- Nginx: به عنوان Reverse Proxy و مدیریت SSL.
- Docker: برای کانتینرایز کردن سرویس‌ها (n8n, Bot, DB).

۱۰. چرخه یادگیری (Feedback Loop)

مکانیزم اصلاح وزن‌ها:

- جمع‌آوری امتیازات کاربر نسبت به هم‌گروهی‌ها و ایونت.
- اصلاح بردار کاربر در صورت نارضایتی (دور کردن از خوشه‌های مشابه).

دیتای ضمنی (Implicit Data):

- تحلیل سرعت پاسخگویی و تعاملات در گروه برای بهبود مچینگ‌های آینده.

۱۱. جمع‌بندی فنی پروژه راوی

- راوی یک پلتفرم داده‌محور است که فراتر از یک پیام‌رسان عمل می‌کند.
- ترکیب معماری FSM و Redis، تعاملی سریع و بدون باگ را تضمین می‌کند.
- استفاده از Webhook و Middleware امنیت و مقیاس‌پذیری را فراهم کرده است.
- پایپ‌لاین ۶ مرحله‌ای n8n، داده‌های خام را به دانش قابل استفاده تبدیل می‌کند.
- قلب سیستم، الگوریتم خوشه‌بندی پایتونی است که با فیدبک کاربران هوشمندتر می‌شود.

پایان