



دانشکده مهندسی کامپیوتر

چت‌بات همدل

پروژه کارشناسی مهندسی کامپیوتر گرایش هوش مصنوعی و رباتیک

گلبرگ سپهرآرا

استاد راهنما

دکتر سید صالح اعتمادی

۱۴۰۳ دی

سَلَامٌ

تأییدیه‌ی هیأت داوران جلسه‌ی دفاع از پروژه

نام دانشکده: دانشکده مهندسی کامپیوتر

نام دانشجو: گلبرگ سپهر آرا

عنوان پروژه: چتبات همدل

تاریخ دفاع: دی ۱۴۰۳

رشته: مهندسی کامپیوتر

گرایش: هوش مصنوعی و رباتیک

ردیف	سمت	نام و نام خانوادگی	مرتبه دانشگاهی	دانشگاه یا مؤسسه	امض
۱	استاد راهنما	دکتر صالح اعتمادی	استادیار	دانشگاه علم و صنعت ایران	
۲	استاد مدعو داخلی	دکتر ناصر مزینی	استادیار	دانشگاه علم و صنعت ایران	

ب

تأییدیه‌ی صحت و اصالت نتایج

با اسمه تعالی

اینجانب گلبرگ سپهرآرا به شماره دانشجویی ۹۹۵۲۱۳۳۴ دانشجوی رشته مهندسی کامپیوتر مقطع تحصیلی کارشناسی تأیید می‌نمایم که کلیه‌ی نتایج این پژوهه حاصل کار ایننجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری شده از آثار دیگران را با ذکر کامل مشخصات منع ذکر کرده‌ام. در صورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انصباطی ...) با ایننجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احراق حقوق مکتب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی‌صلاح (اعم از اداری و قضایی) به عهده‌ی ایننجانب خواهد بود و دانشگاه هیچ‌گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: گلبرگ سپهرآرا

تاریخ و امضا:

پ

مجوز بهره‌برداری از پایان‌نامه

بهره‌برداری از این پایان‌نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر تعیین می‌شود، بلامانع است:

- بهره‌برداری از این پایان‌نامه برای همگان بلامانع است.
- بهره‌برداری از این پایان‌نامه با اخذ مجوز از استاد راهنما، بلامانع است.
- بهره‌برداری از این پایان‌نامه تا تاریخ ممنوع است.

استاد راهنما: دکتر سید صالح اعتمادی

تاریخ:

امضا:

امروزه با پیشرفت‌های چشمگیر صورت‌گرفته در زمینه‌ی رباتیک و هوش مصنوعی، این مخلوق انسانی به دستیاری تقریباً همه‌چیزدان و همیشه حاضر برای انسان‌ها تبدیل شده است. ساخت و آموزش مدل‌های زبانی بزرگ و نتایج نسبتاً رضایت‌بخش آن‌ها در انجام تسلک‌های مختلف، از جمله تشخیص احساسات و تولید متن، باعث شده که هوش مصنوعی هرچه سریع‌تر، مسیر نزدیک شدن به زبان و درک انسان را پیش بگیرد. هوش مصنوعی در حال حاضر به عنوان یک مشاور/دستیار با بینش و دانش گستره‌ده، قابل اعتماد و ارزیابی است. اما سوال این است که آیا از نظر احساسی و همدلی هم تا همین اندازه امکان همدلی و اظهار نظرات کمک‌کننده را دارد؟

این پژوهه پیاده‌سازی یک نرم‌افزار کاربردی برای پیشنهاد مطالب مشابه با پیام ورودی کاربر از تعدادی کanal تلگرامی پرطوفدار می‌باشد. مجموعه‌داده‌ی استفاده شده، مطالب کanal‌های تلگرامی پرطوفداری است که صاحبان آن روزمرگی‌ها، نگرش و حرف‌های عادی و بداهه‌ی خود را در آن می‌نویستند و آن‌جا برایشان حکم یک فضای خودمانی و امن را دارد. در نتیجه، پیام‌های مشابهی که از این مجموعه‌داده بازیابی و نشان داده می‌شود، حس ملموس‌تری داشته و همدلی بیشتری را القا می‌کنند. همچنین چون لحن پیام‌ها به صورت عامیانه و خودمانی است، موجب بهبود روحیه و شادی می‌شود. در این پژوهه برای پیدا کردن پیام‌های مشابه، از محاسبه‌ی فاصله‌ی کسینوسی بین پیام ورودی توسط کاربر و پیام‌های مجموعه‌داده استفاده شده است. ده پیامی که فاصله‌ی کمتری دارند، به کاربر نمایش داده خواهند شد. سپس کاربر از میان آن‌ها، پیام‌هایی را که پسندیده است، انتخاب می‌کند. پیام‌های منتخب به مدل مولد GPT-4o-mini فرستاده شده و پاسخی با استفاده از پیام‌های برگزیده از آن دریافت می‌کند. در نهایت، یک نمونه مدل پاداش با استفاده از مجموعه‌داده بازخورد جمع‌آوری شده، آموزش داده می‌شود. عملکرد نرم‌افزار با معیارهایی نظیر تعداد پیام‌های منتخب نسبت به کل پیام‌های نمایش داده شده و همچنین جمع معکوس رتبه‌های پیام‌های منتخب ارزیابی شده است. نتیجه‌ی ارزیابی انجام‌شده بر روی مجموعه‌ی بازخورد جمع‌آوری شده نشان می‌دهد. در این برنامه به طور میانگین ۳۰ درصد از پیام‌های نشان داده شده مورد علاقه می‌باشند و همچنین کاربر به سرعت محتوای مورد نظر را بدون نیاز به پیمایش پیام‌ها پیدا می‌کند. همچنین از میان بازخوردهای جمع‌آوری شده، ۵۶ پیام از ۶۱ پیام دارای حداقل یک پیام منتخب هستند. این

ث

نشان می‌دهد که این برنامه با دقت ۹۱.۸ درصد، عملکردی کاربردی داشته و موجب تغییر روحیه و آشنایی با نگرش‌های جدید شده است.

واژگان کلیدی: پیام همدلانه، نمایش برداری جملات، بازیابی، فاصله‌ی کسینوسی، تولید متن،
برنامه‌ی کاربردی GPT-4o-mini

فهرست مطالب

د

فهرست تصاویر

۱	فصل ۱ : مقدمه
۵	فصل ۲ : تعاریف و مفاهیم اولیه
۵	۱-۲ مقدمه
۵	۲-۲ هوش مصنوعی
۶	۳-۲ یادگیری ماشین
۷	۱-۳-۲ مدل
۸	۲-۳-۲ شبکه‌های عصبی
۹	۴-۲ یادگیری عمیق
۱۱	۵-۲ تعبیه
۱۱	۶-۲ مدل‌های زبانی
۱۲	۱-۶-۲ مدل‌های تعبیه‌گر جمله
۱۴	۲-۶-۲ مدل‌های مولد
۱۶	۷-۲ پردازش زبان طبیعی
۱۶	۸-۲ اصطلاحات استفاده شده در این پژوهه
۱۶	۱-۸-۲ تولید افزوده شده با بازیابی (RAG)
۱۷	۲-۸-۲ یادگیری تقویتی (RL)
۱۸	۳-۸-۲ یادگیری تقویتی از بازخورد انسانی (RLHF)
۱۹	۹-۲ معیارهای شباهت سنجی

ج

فهرست مطالب

ج

۱۹	ضرب نقطه‌ای ۱-۹-۲
۲۰	فاصله‌ی کسینوسی ۲-۹-۲
۲۰	اصلاحات رایج در آموزش مدل ۱۰-۲
۲۰	تکوازساز (Tokenizer) ۱-۱۰-۲
۲۱	(epoch) ۲-۱۰-۲
۲۱	(Tensor) ۳-۱۰-۲
۲۱	ماسک توجه (Attention Mask) ۴-۱۰-۲
۲۱	nn.BCEWithLogitsLoss ۵-۱۰-۲
۲۲	optim.Adam ۶-۱۰-۲
۲۴	فصل ۳: مروری بر کارهای مرتبط
۲۴	مقدمه ۱-۳
۲۴	ChatGPT ۲-۳
۲۵	ChatGPT ۱-۲-۳
۲۶	و استفاده از بازخورد انسانی (RLHF) ۱-۲-۳
۲۶	برنامه‌های حمایت از سلامت روان ۳-۳
۲۶	چتبات Wysa ۱-۳-۳
۲۷	همراهان دوستی ۴-۳
۲۷	چتبات Replika ۱-۴-۳
۲۸	Replika ۲-۴-۳
۲۸	چشم‌انداز کلی ۳-۴-۳
۲۸	تحلیل توسعه روابط انسان-چتبات با Replika ۴-۴-۳
۲۸	بررسی فرآیند توسعه HCR ۴-۴-۳
۲۹	چتبات‌های سوگ و پشتیبانی ۵-۳
۲۹	چتبات Tess (by X2AI) ۱-۵-۳
۳۱	فصل ۴: روش پیشنهادی
۳۱	مقدمه ۱-۴
۳۱	بررسی مراحل توسعه ۲-۴

فهرست مطالب

ح

۳۳	۳-۴ جمع‌آوری داده
۳۴	۴ توسعه‌ی مجموعه‌داده
۳۴	۱-۴ به دست آوردن تعبیه‌ی پیام
۳۶	۵ توسعه‌ی رابط کاربری
۳۶	۶ راهاندازی نرم‌افزار بر روی سرور
۳۷	۱-۶ خرید سرور
۳۷	۲-۶ خرید دامنه
۳۷	۳-۶ راهاندازی نرم‌افزار بر روی سرور
۳۸	۷-۴ پیدا کردن مشابه‌ها
۳۹	۸-۴ گرفتن بازخورد از کاربر
۳۹	۹-۴ ارزیابی

۴۰	فصل ۵: روش پیاده‌سازی
۴۰	۱-۵ مقدمه
۴۰	۲-۵ جمع‌آوری داده
۴۳	۳-۵ ساخت مجموعه‌ی داده
۴۴	۱-۳-۵ ذخیره بر روی Pinecone
۴۶	۲-۳-۵ بارگذاری مجموعه داده در huggingface
۴۷	۴-۵ ساخت برنامه‌ی کاربردی تحت وب
۴۷	۱-۴-۵ مقدمه
۴۷	۲-۴-۵ ساخت رابط کاربری
۴۹	۳-۴-۵ وصل کردن مجموعه داده به برنامه‌ی کاربردی
۵۰	۴-۴-۵ پیدا کردن پیام مشابه با پیام کاربر
۵۱	۵-۴-۵ نمایش پیام‌ها
۵۳	۶-۴-۵ ساخت مجموعه داده‌ی بازخورد
۵۵	۷-۴-۵ تولید پاسخ

فهرست مطالب

خ	
۵۷ ۸-۴-۵ ارزیابی
۵۸ ۹-۴-۵ بررسی میزان رضایتمندی
۵۸ ۱۰-۴-۵ اخیرید سرور
۵۹ ۱-۴-۵ ارها اندازی برنامه بر روی سرور
۶۱ ۵-۵ آموزش مدل پاداش
۶۱ ۱-۵-۵ تشکیل مجموعه داده جفت شده
۶۲ ۲-۵-۵ بارگیری مدل
۶۳ ۳-۵-۵ ساخت مجموعه داده مناسب برای مدل
۶۵ ۴-۵-۵ آموزش
۶۶ ۵-۵-۵ ارزیابی
۶۷ ۶-۵-۵ امتحان یک پیام ورودی
۶۹ فصل ۶: بررسی نتایج
۶۹ ۱-۶ جمع‌بندی
۷۰ ۱-۱-۶ بررسی چند نمونه خروجی
۷۲ ۲-۶ کارهای آینده
۷۳ کتاب‌نامه

فهرست تصاویر

۱۸	۱-۲ مراحل یادگیری تقویتی از بازخورد انسانی [۱]
۳۲	۱-۴ مرحله‌ی آفلاین توسعه
۳۳	۲-۴ مرحله‌ی آنلاین
۴۱	۱-۵ جمع‌آوری پیام
۴۲	۲-۵ اطلاعات به صورت html می‌باشد.
۴۲	۳-۵ بازیابی متن از فایل html
۴۴	۴-۵ راهاندازی مجموعه داده در Pinecone
۴۵	۵-۵ تریق تعبیه‌ها به مجموعه
۴۶	۶-۵ نمونه‌ی ورودی و خروجی - ضرب نقطه‌ای
۴۶	۷-۵ فرستادن مجموعه‌داده به هاب
۴۸	۸-۵ ساخت صفحه‌ی چت
۴۹	۹-۵ ساخت یک برنامه‌ی چت
۴۹	۱۰-۵ دانلود مجموعه‌داده از هاب
۵۰ cache	۱۱-۵ ذخیره‌ی داده‌ها در دسترس بیشتر و افزایش سرعت با استفاده از cache
۵۰	۱۲-۵ به دست‌آوردن تعبیه‌ی پیام و پیدا کردن مشابه‌ها
۵۱	۱۳-۵ نمایش پیام‌ها
۵۱ like dislike	۱۴-۵ گذاشتن دکمه‌ی like و dislike
۵۲	۱۵-۵ نمای کاربری برنامه
۵۲ click-button	۱۶-۵ تابع like برای پیام‌های شده

۵۳	تابع ۱۷-۵ click-disbutton برای پیام‌های dislike شده
۵۴	تابع ۱۸-۵ ذخیره‌ی بازخورد
۵۴	یک نمونه از بازخورد ثبت شده ۱۹-۵
۵۵	فایل بش ۲۰-۵
۵۵	فایل ۲۱-۵ crontab
۵۶	ارسال پیام‌های منتخب و دریافت پاسخ ۲۲-۵
۵۶	یک نمونه بازخورد ثبت شده ۲۳-۵
۵۷	یک نمونه پاسخ تولید شده ۲۴-۵
۵۹	فایل سرویس ۲۵-۵
۶۰	فایل پیکربندی NGINX ۲۶-۵
۶۲	ساخت جفت‌های منتخب و نامنتخب ۲۷-۵
۶۳	مدل پاداش ۲۸-۵
۶۴	ساخت مجموعه‌داده جهت آموزش مدل ۲۹-۵
۶۵	تنظیم تابع ضرر، بهینه‌ساز و مدل پاداش ۳۰-۵
۶۶	آموزش مدل ۳۱-۵
۶۷	تابع ارزیابی ۳۲-۵
۶۷	بررسی یک نمونه ورودی ۳۳-۵
۶۸	ربته‌بندی پیام‌ها بر اساس ورودی ۳۴-۵
۶۸	خروجی مدل ۳۵-۵
۷۰	۱-۶ زندگی ساده!
۷۰	۲-۶ خوابم میاد!
۷۰	۳-۶ هر چی شد شد!
۷۱	۴-۶ شروع از همین حالا!
۷۱	۵-۶ غر نزن و ادامه بده!
۷۱	۶-۶ درست شد!

فهرست تصاویر

ر

۷۲ نه به حسادت! ۷-۶

فصل ۱

مقدمه

امروزه، سرعت سرسام آور زندگی روزانه و سبک زندگی پر مشغله‌ی جوامع، همچنین تنوع اهداف و خواسته‌ها و طمع انسان‌ها در داشتن بهترین چیزها، سبب شده است که اغلب افراد زمان کمتری را به تفکر در خود اختصاص دهند. یافتن آرزوها و علایق حقیقی، فارغ از اثرات جو و جامعه، به طور نامحسوسی به چالشی در زندگی فردی تبدیل شده است و بسیاری از افراد نسبت به چشیدن طعم زندگی و درک ارزش واقعی افراد و لحظات بی‌توجه‌اند. این موضوع موجب افزایش فشارهای روحی و روانی روی افراد و افزایش تنش‌ها و ناهنجاری‌ها در جوامع شده است.

در این موضع است که استفاده از تجربیات دیگران—افرادی که دو قدم جلوتر از مسیری که ما برگزیده‌ایم را پیموده‌اند یا کسانی که مسیرهای متفاوتی را گزیده‌اند و نگاه متفاوتی دارند—و در کل، آشنا شدن با نگرش‌های مختلف نسبت به موضوعی که ما آن را زندگی و تجربه می‌کنیم، می‌تواند بسیار به درک اهداف و خواسته‌های اصلی هرکس از زندگی کمک کند و موجب رشد نگرش هر شخص به زندگی و همین‌طور نزدیکی بیشتر به روح آزاد خود شود.

فصل ۱. مقدمه

در سال‌های اخیر، پیشرفت‌های چشمگیر در زمینه‌ی پردازش زبان طبیعی^۱، تحولی شگرف در زندگی افراد ایجاد کرده است. از جمله کاربردهای پردازش زبان طبیعی می‌توان به سیستم‌های توصیه‌گر شخصی‌سازی‌شده، دستیارهای هوشمند و جستجوی معنایی اشاره کرد که امروزه به بخشی جدایی‌ناپذیر از زندگی ما تبدیل شده‌اند. از این دستاوردها می‌توان در جهت شناخت و درک علت فشارهای روانی و تلاش برای پیشنهاد راهکار یا همدلی استفاده نمود.

یکی از حوزه‌های مهم در پژوهش‌های پردازش زبان طبیعی، توسعه‌ی مدل‌های زبانی^۲ هوشمند است. این مدل‌ها به عنوان ابزارهای کلیدی پردازش زبان طبیعی^۳ به ماشین‌ها امکان می‌دهند تا زبان انسان را درک کرده، تولید و تحلیل نمایند. این مدل‌ها که با استفاده از داده‌های متنی گسترده آموزش می‌بینند، با شناسایی الگوهای یادگرفته‌شده از داده‌ها، متنی تولید می‌کنند که از نظر گرامری صحیح و از نظر معنایی منسجم است.

در این پژوهه، با استفاده از فضای پنهان جملات (Embeddings) و بازیابی اطلاعات از تعدادی کanal‌های تلگرامی پرطرفدار که کاربر عضو آن‌هاست، تجربه‌های واقعی افراد مختلف همسو با پیام ورودی کاربر استخراج و به متن تولیدشده توسط GPT-4o-mini^۴ اضافه می‌شود.^۴ بدین ترتیب، کاربر با مشاهده‌ی پیام‌های مشابه، احساسات و دیدگاه‌های افراد دیگر را در ارتباط با پیام خود دریافت کرده و پیام‌های تولیدشده حس همدلی و ارتباط ملموس‌تری را منتقل می‌کنند. این رویکرد باعث می‌شود پاسخ‌ها به نیازهای عاطفی و احساسی کاربر نزدیک‌تر شده و تجربه‌ای شخصی‌تر و انسانی‌تر ایجاد گردد.

مجموعه‌داده‌ی این برنامه از پیام‌های تعدادی از کanal‌های پرطرفدار تشکیل شده است. محتوای

¹Natural Language Processing (NLP)

²Language Models

³Natural Language Processing

⁴Retrieval Augmented Generation (RAG)

فصل ۱. مقدمه

این کانال‌ها شامل روزمرگی‌ها، نگرش صاحبان کانال نسبت به اخبار یا اتفاقات روزانه، و هر حرفی است که افراد مایل‌اند آن را به اشتراک بگذارند یا ثبت کنند. پس از جمع‌آوری پیام‌ها و انجام پردازش‌های لازم، این پیام‌ها به فرم قابل‌فهم برای ماشین تبدیل می‌شوند. برای به دست آوردن تعبیه‌ها، از مدل text-embedding-3-small استفاده شده است. هر سطر از مجموعه‌داده شامل پیام کانال و تعبیه‌ی نظری آن است.

این برنامه پس از دریافت پیام ورودی از کاربر، تعبیه‌ی آن را از مدل تعبیه‌گر ذکر شده دریافت می‌کند. سپس با محاسبه‌ی فاصله‌ی کسینوسی میان تعبیه‌ی ورودی و تعبیه‌های مجموعه‌داده، ده پیام نزدیک‌تر(کم فاصله‌تر) را بازمی‌گرداند. کاربر می‌تواند پیام‌ها را لایک یا دیس‌لایک^۵ کند. این بازخوردها در یک مجموعه‌داده ذخیره می‌شوند. برای هر پیام ورودی، پس از انتخاب موارد مورد علاقه توسط کاربر، دقت^۶ برنامه از طریق نسبت تعداد پیام‌های منتخب به تعداد کل پیام‌های نمایش‌داده شده و نیز جمع معکوس رتبه‌ی پیام‌های منتخب ارزیابی می‌شود. در این برنامه، به‌طور میانگین ۳۰ درصد از پیام‌های نشان‌داده شده مورد علاقه کاربر قرار می‌گیرند و کاربران در ۳۰ درصد موضع، محتوای موردنظر خود را در رتبه‌های نزدیک‌تر پیدا می‌کنند. همچنین از میان بازخوردهای جمع‌آوری شده، ۵۶ پیام از ۶۱ پیام دارای حداقل یک پیام منتخب هستند. این نشان‌دهنده‌ی دقت ۹۱.۸ درصدی برنامه و تأثیر مثبت آن در بهبود روحیه و آشنایی با نگرش‌های جدید است.

پیام‌های منتخب به GPT-4o-mini، که نسخه‌ای کوچک و مقرون‌به‌صرفه از مدل GPT-4o و جایگزینی برای GPT-3.5 Turbo است، ارسال می‌شوند. این مدل از پیام‌های منتخب برای تولید پاسخ استفاده می‌کند و پیام نهایی به کاربر نمایش داده می‌شود. در پایان، مدل پاداش برای پیشنهاد مرتبط‌ترین پیام‌ها پیاده‌سازی شده است. این مجموعه‌داده می‌تواند در آینده برای آموزش مدل‌های

⁵Dislike

⁶Precision

فصل ۱. مقدمه

مشاور، دستیار، یا سیستم‌های توصیه‌گر استفاده شود.

این پایان‌نامه شامل شش بخش اصلی است: ۱. مقدمه: توضیح مفصلی در مورد انگیزه‌ی پروژه و دید کلی از مراحل اجرای آن. ۲. مفاهیم پایه‌ای: ارائه‌ی اصطلاحات رایج مرتبط برای درک بهتر پروژه. ۳. مروری بر کارهای مرتبط: بررسی پژوهش‌های مشابه انجام شده. ۴. معماری پروژه: شرح فرآیند کلی پیاده‌سازی. ۵. مراحل پیاده‌سازی: توضیح جزئیات اجرا همراه با شواهد و مثال‌ها. ۶. جمع‌بندی: ارائه‌ی نتایج پروژه، نمونه‌هایی از خروجی‌ها و مقایسه با کارهای مشابه.

فصل ۲

تعاریف و مفاهیم اولیه

۱-۲ مقدمه

در این فصل، به منظور درک کلی موضوع، نخست مفاهیم پایه‌ای پژوهش را بیان می‌کنیم. در ابتدا، هوش مصنوعی و زیرشاخه‌های یادگیری ماشین و پردازش زبان طبیعی از هوش مصنوعی را توصیف کرده و نمونه‌هایی از کاربردها و ارتباط تنگاتنگ آن‌ها با زندگی امروزی را بررسی می‌کنیم. سپس به تعریف، معرفی عناصر، معماری و محدودیت‌های مدل‌های زبانی و مدل‌های مولد می‌پردازیم و در ادامه، فرآیند تولید متن همراه با بازیابی اطلاعات از منابع جدید (RAG) را مورد بررسی قرار می‌دهیم.

۲-۲ هوش مصنوعی

هوش مصنوعی^۱ (AI) فناوری‌ای است که به کامپیوترها و ماشین‌ها امکان می‌دهد یادگیری، درک، حل مسئله، تصمیم‌گیری، خلاقیت و استقلال انسان را شبیه‌سازی کنند. برنامه‌ها و دستگاه‌های مجهز به هوش مصنوعی قادرند اشیاء را ببینند و شناسایی کنند، زبان

^۱ Artificial Intelligence

فصل ۲. تعاریف و مفاهیم اولیه

۳-۲. یادگیری ماشین

انسانی را درک کرده و به آن پاسخ دهنده، از اطلاعات و تجربیات بیاموزند، پیشنهادات دقیق به کاربران و متخصصان ارائه دهنده و به طور مستقل عمل کنند. نمونه‌ای کلاسیک از این استقلال، خودروهای خودران است که نیاز به مداخله یا هوش انسانی را از بین می‌برند.

اما در سال ۲۰۲۴، بیشتر پژوهشگران و متخصصان هوش مصنوعی و اغلب تیترهای مرتبط با این حوزه بر روی پیشرفت‌های هوش مصنوعی مولد (gen AI) متمرکز هستند؛ فناوری‌ای که قادر به تولید متن، تصاویر، ویدئو و سایر محتواهای اصیل است. برای درک کامل هوش مصنوعی مولد، ابتدا باید فناوری‌هایی که این ابزارها بر آن‌ها بنا شده‌اند، یعنی یادگیری ماشین (ML) و یادگیری عمیق [۴]، را شناخت.

۳-۲ یادگیری ماشین

یادگیری ماشین شامل ایجاد مدل‌هایی (models) است که با استفاده از یک الگوریتم، توانایی پیش‌بینی یا تصمیم‌گیری براساس داده‌ها را پیدا می‌کنند. این حوزه طیف گسترده‌ای از تکنیک‌ها را در بر می‌گیرد که به کامپیوترها امکان می‌دهد بدون برنامه‌نویسی دقیق برای وظایف خاص، از داده‌ها یاد بگیرند و نتیجه‌گیری کنند.

روش‌ها و الگوریتم‌های متعددی در یادگیری ماشین وجود دارند، از جمله رگرسیون خطی^۲، رگرسیون لجستیک^۳، درخت‌های تصمیم‌گیری^۴، جنگل تصادفی^۵، ماشین‌های بردار پشتیبانی^۶، نزدیک‌ترین همسایه^۷، خوشه‌بندی^۸ و بسیاری دیگر. هر یک از این روش‌ها برای انواع مختلفی

²linear regression

³logistic regression

⁴decision trees

⁵random forest

⁶support vector machines (SVMs)

⁷k-nearest neighbor (KNN)

⁸clustering

از مسائل و داده‌ها مناسب هستند. اما یکی از محبوب‌ترین انواع الگوریتم‌های یادگیری ماشین، شبکه‌های عصبی^۹ است. به منظور درک بهتر، به معرفی مختصری از مفاهیم اساسی اشاره شده، می‌پردازیم: [۳]

۱-۳-۲ مدل

در یادگیری ماشین، یک مدل یک نمایش ریاضی یا تابع است که برای پیش‌بینی یا تصمیم‌گیری بر اساس داده‌های ورودی آموزش داده می‌شود. مدل، روابط یا الگوهای موجود در داده‌ها را خلاصه می‌کند و سیستم را قادر می‌سازد تا تعمیم دهد و در مورد داده‌های جدید و دیده‌نشده استنتاج کند.

عناصر کلیدی یک مدل:

- پارامترها: متغیرهای داخلی مدل که در طول آموزش یاد گرفته می‌شوند، مانند وزن‌ها و بایاس‌ها^{۱۰} در شبکه‌های عصبی.
- ساختار: معماری یا الگوریتمی که نحوه پردازش داده‌های ورودی را تعریف می‌کند، مانند رگرسیون خطی، درخت‌های تصمیم و شبکه‌های عصبی.
- داده‌های ورودی: داده‌ایی که در طول آموزش و پیش‌بینی به مدل داده می‌شوند.
- خروجی: پیش‌بینی‌ها یا طبقه‌بندی‌هایی که توسط مدل انجام می‌شود.
- آموزش: فرآیند تنظیم پارامترها با استفاده از داده‌های برچسب‌دار (یادگیری نظارت شده) یا بدون برچسب (یادگیری بدون نظارت).

⁹Neural Networks

¹⁰biases

انواع مدل‌ها:

- مدل‌های نظارت شده: پیش‌بینی نتایج بر اساس داده‌های آموزشی برچسب‌دار (مانند رگرسیون خطی، ماشین‌های بردار پشتیبان).
- مدل‌های بدون نظارت: کشف الگوهای در داده‌های بدون برچسب (مانند خوشه‌بندی، تحلیل مؤلفه اصلی).
- مدل‌های تقویتی: یادگیری تصمیم‌گیری از طریق پاداش و جریمه (مانند یادگیری Q یادگیری تقویتی عمیق).

یک مدل خوب آموزش‌دیده باید قادر باشد تا داده‌های جدید را تعمیم دهد و از بیش‌برازش (حفظ جزئیات داده‌های آموزشی) و کم‌برازش (شکست در کشف الگوهای جلوگیری) کند.

۲-۳-۲ شبکه‌های عصبی

یک شبکه عصبی یک برنامه یا مدل یادگیری ماشین است که به شیوه‌ای مشابه با مغز انسان تصمیم‌گیری می‌کند. این کار از طریق فرآیندهایی انجام می‌شود که نحوه همکاری نورون‌های بیولوژیکی برای شناسایی پدیده‌ها، ارزیابی گزینه‌ها و رسیدن به نتایج را تقلید می‌کنند. هر شبکه عصبی از لایه‌هایی از گره‌ها یا نورون‌های مصنوعی تشکیل شده است: یک لایه ورودی، یک یا چند لایه مخفی، و یک لایه خروجی. هر گره به گره‌های دیگر متصل است و دارای وزن و آستانه خاص خود است. اگر خروجی یک گره از مقدار آستانه مشخص شده بیشتر باشد، آن گره فعال می‌شود و داده‌ها را به لایه بعدی شبکه ارسال می‌کند. در غیر این صورت، داده‌ای به لایه بعدی منتقل نمی‌شود. شبکه‌های عصبی برای یادگیری و بهبود دقت خود در طول زمان به داده‌های آموزشی متکی هستند. پس از

۴-۲. یادگیری عمیق

تنظیم دقیق برای دستیابی به دقت بالا، این شبکه‌ها ابزارهای قدرتمندی در علوم کامپیوتر و هوش مصنوعی محسوب می‌شوند و امکان طبقه‌بندی و خوشه‌بندی داده‌ها با سرعت بالا را فراهم می‌کنند. برای مثال، وظایفی مانند تشخیص گفتار یا تصویر که به صورت دستی ساعت‌ها زمان می‌برد، با شبکه‌های عصبی در عرض چند دقیقه انجام می‌شود. یکی از شناخته شده‌ترین مثال‌ها در این زمینه، الگوریتم جستجوی گوگل است. شبکه‌های عصبی گاهی با نام شبکه‌های عصبی مصنوعی^{۱۱} یا شبکه‌های عصبی شبیه‌سازی شده^{۱۲} شناخته می‌شوند. این شبکه‌ها زیرمجموعه‌ای از یادگیری ماشین هستند و در قلب مدل‌های یادگیری عمیق قرار دارند.

۴-۲ یادگیری عمیق

یادگیری عمیق یکی از زیرمجموعه‌های یادگیری ماشین است که از شبکه‌های عصبی چندلایه، به نام شبکه‌های عصبی عمیق، استفاده می‌کند تا قدرت تصمیم‌گیری پیچیده مغز انسان را بهتر شبیه‌سازی کند. این شبکه‌ها شامل یک لایه ورودی، حداقل سه لایه مخفی (و معمولاً صدھا لایه)، و یک لایه خروجی هستند. برخلاف شبکه‌های عصبی مورد استفاده در مدل‌های کلاسیک یادگیری ماشین که معمولاً فقط یک یا دو لایه مخفی دارند، شبکه‌های عصبی عمیق با تعداد لایه‌های بیشتر امکان یادگیری بدون نظارت را فراهم می‌کنند.

شبکه‌های عصبی عمیق قادرند به طور خودکار ویژگی‌ها را از مجموعه داده‌های بزرگ، بدون برچسب و ساختار نیافته استخراج کرده و پیش‌بینی‌هایی درباره معنای داده‌ها انجام دهند. از آنجا که یادگیری عمیق نیاز به مداخله انسانی ندارد، این روش یادگیری ماشین را در مقیاس عظیم ممکن می‌سازد. این ویژگی یادگیری عمیق را برای وظایفی مانند پردازش زبان طبیعی (NLP)، بینایی کامپیوتر

^{۱۱}artificial neural networks(ANNs)

^{۱۲}simulated neural networks(SNNs)

و شناسایی سریع و دقیق الگوها و روابط پیچیده در داده‌های حجمی بسیار مناسب می‌سازد. بخش عمده‌ای از برنامه‌های هوش مصنوعی که امروزه در زندگی ما وجود دارند، از یادگیری عمیق قدرت می‌گیرند.^۴ یادگیری عمیق همچنین امکان بهره‌گیری از رویکردهای پیشرفته زیر را فراهم می‌آورد:

- یادگیری نیمه‌ناظارت شده^{۱۳}: ترکیبی از یادگیری ناظارت شده و بدون ناظارت که از داده‌های برچسب دار و بدون برچسب برای آموزش مدل‌های هوش مصنوعی در وظایف طبقه‌بندی و رگرسیون استفاده می‌کند.
- یادگیری خودناظارت شده^{۱۴}: به جای استفاده از مجموعه داده‌های برچسب دار برای ارائه سیگنال‌های ناظارتی، از برچسب‌های ضمنی تولید شده از داده‌های غیرساختاریافته استفاده می‌کند.
- یادگیری تقویتی^{۱۵}: با استفاده از روش آزمون و خطا و توابع پاداش، به جای استخراج اطلاعات از الگوهای پنهان، یاد می‌گیرد
- یادگیری انتقالی^{۱۶}: دانشی که از یک وظیفه یا مجموعه داده به دست آمده است، برای بهبود عملکرد مدل در یک وظیفه مرتبط دیگر یا مجموعه داده متفاوت استفاده می‌شود.

¹³Semi-supervised learning

¹⁴Self-supervised learning

¹⁵Reinforcement learning (RL)

¹⁶Transfer learning

۵-۲ تعبیه

تعبیه‌ها^{۱۴} دنباله‌ای از اعداد هستند که مفاهیم موجود در محتواهایی نظیر متن، تصاویر و صدا را به صورت برداری نمایش می‌دهند. این نمایش برداری به گونه‌ای طراحی شده است که برای مدل‌های یادگیری ماشین و الگوریتم‌های جستجوی معنایی قابل استفاده باشد.

تعبیه‌ها محتوای ورودی را بر اساس ویژگی‌ها و روابط موجود در آن‌ها به شکل ریاضی تبدیل می‌کنند. این ویژگی‌ها ممکن است شامل عواملی باشد که محتوا دارای آن‌ها است یا نیست و همچنین دسته‌بندی‌هایی که محتوا به آن‌ها تعلق دارد. به طور کلی، تعبیه‌ها امکان یافتن اشیاء مشابه را برای مدل‌های یادگیری ماشین فراهم می‌کنند.

به عنوان مثال، یک مدل یادگیری ماشینی که از تعبیه‌ها استفاده می‌کند، با دریافت یک عکس یا یک سند می‌تواند عکس یا سند مشابهی را پیدا کند. از آنجایی که تعبیه‌ها توانایی درک روابط بین کلمات و اشیاء دیگر را برای رایانه‌ها فراهم می‌کنند، یکی از اجزای اساسی در هوش مصنوعی به شمار می‌آیند.^{۱۵}

۶-۲ مدل‌های زبانی

یک مدل زبانی^{۱۶} (LM) نوعی مدل یادگیری ماشین است که برای توزیع احتمال بر روی کلمات آموخته دیده است. به بیان ساده‌تر، این مدل تلاش می‌کند بر اساس متن داده شده، مناسب‌ترین کلمه بعدی را برای پر کردن جای خالی در یک جمله یا عبارت پیش‌بینی کند.^{۱۷}

¹⁷embeddings

¹⁸Language Model

۶-۱ مدل‌های تعبیه‌گر جمله

مدل‌های تعبیه‌گر جمله^{۱۹} به گونه‌ای طراحی شده‌اند که جوهر معنایی یک جمله را در یک بردار با طول ثابت خلاصه کنند. این مدل‌ها قادرند بافت، معنا، و روابط بین کلمات را به تصویر بکشند. بر خلاف روش‌های سنتی مانند Bag-of-Words (BoW) یا رمزگذاری یک‌طرفه^{۲۰} که تنها حضور یا عدم حضور کلمات را نشان می‌دهند، تعبیه‌های جملات با در نظر گرفتن زمینه کلمات، نمایش معنایی عمیق‌تری ارائه می‌دهند. این قابلیت برای درک ظرافت‌های زبان انسانی توسط ماشین‌ها ضروری است.^{۲۱}

چندین روش برای ایجاد تعبیه‌های جملات وجود دارد که در زیر به رایج‌ترین آن‌ها اشاره شده است:

۱. میانگین تعبیه‌های کلمات: این روش شامل میانگین‌گیری تعبیه‌های کلمات یک جمله است. با وجود سادگی، این رویکرد اغلب از نمایش تفاوت‌های ظریف‌تری در جملات پیچیده باز می‌ماند.

۲. مدل‌های از پیش آموزش دیده: مدل‌هایی نظیر BERT^{۲۱} انقلابی در تعبیه‌سازی جملات ایجاد کرده‌اند. این مدل‌ها با در نظر گرفتن زمینه هر کلمه، تعبیه‌های غنی و معنایی تولید می‌کنند که به درک عمیق‌تر زبان کمک می‌کند.^{۲۲}

۳. رویکردهای مبتنی بر شبکه عصبی: روش‌هایی مانند InferSent^{۲۳} و Skip-Thought^{۲۴} نمونه‌هایی از مدل‌های مبتنی بر شبکه عصبی هستند.

¹⁹Sentence Embedding Models

²⁰One-Hot Encoding

²¹Bidirectional Encoder Representations

نتایج قوی‌تری را نسبت به Skip-Thought ارائه می‌دهد و بر اساس مجموعه InferSent ●

داده‌های نظارت‌شده‌ای نظری Stanford Natural Language Inference datasets آموزش دیده

است. [۹]

از رویکرد یادگیری بدون نظارت استفاده می‌کند و هدف آن پیش‌بینی Skip-Thought ●

جملات اطراف است. [۱۰]

مدل text-embedding-3-small

یک مدل جدید تعبیه‌گر بسیار کارآمد است که نسبت به مدل قبلی خود، text-embedding-ada-002،

که در دسامبر ۲۰۲۲ منتشر شد، ارتقاء قابل توجهی را در میانگین امتیاز برای بازیابی چندزبانه^{۲۲}

ارائه می‌دهد. [۱۱]

مدل BERT

BERT که مخفف Bidirectional Encoder Representations from Transformers مدل زبانی است، یک مدل

است که برای پیش‌آموزش نمایش‌های دوسویه عمیق از متن بدون برچسب (یادگیری نظارت‌نشده)

طراحی شده است. این مدل با شرطی‌سازی مشترک بر زمینه چپ و راست (وابستگی توکن مورد

بررسی با در نظر گرفتن توکن‌های قبل و بعد) در تمام لایه‌ها، عمل می‌کند. در نتیجه، مدل BERT از

پیش‌آموزش دیده را می‌توان تنها با افزودن یک لایه خروجی اضافی، برای ایجاد مدل‌های پیشرفته در

طیف وسیعی از وظایف، مانند پاسخ به سؤال و استنتاج زبان، بدون تغییرات اساسی در معماری،

تنظیم کرد. [۸]

²²Multi-language retrieval

ParsBERT مدل

ParsBERT یک مدل زبانی تک‌زبانه مبتنی بر معماری BERT گوگل است. این مدل روی مجموعه‌های بزرگ فارسی با سبک‌های نوشتاری مختلف از موضوعات متعدد (مانند علمی، رمان و اخبار) آموزش دیده است. این مجموعه شامل بیش از ۹.۳ میلیون سند، ۷۳ میلیون جمله و ۳۰.۱ میلیارد کلمه است. [۱۲]

۶-۲ مدل‌های مولد

هوش مصنوعی مولد (Generative AI) که گاهی با نام "gen AI" نیز شناخته می‌شود، به مدل‌های یادگیری عمیقی اشاره دارد که در پاسخ به درخواست یا ورودی کاربر، قادر به تولید محتوای اصلی و پیچیده مانند متن‌های بلند، تصاویر باکیفیت، ویدئوهای واقعی یا صدا و موارد دیگر هستند. در یک سطح کلی، مدل‌های مولد یک نمایش ساده‌شده از داده‌های آموزشی خود را رمزگذاری (تبديل آن به یک نمایش ریاضی – که اغلب به شکل بردارها یا تعبیه‌ها است) می‌کنند و سپس از آن نمایش برای ایجاد محتوای جدیدی استفاده می‌کنند که به داده‌های اصلی شباهت دارد، اما با آن‌ها یکسان نیست. برای مثال، در جمله‌ای مانند: «من کیک را از روی میز برداشته و [...] را داخل یخچال گذاشتم.» یک مدل خوب تشخیص می‌دهد که کلمه‌ی جاافتاده احتمالاً یک ضمیر است. با توجه به این که اطلاعات مرتبط در اینجا "کیک" است، محتمل‌ترین ضمیر "آن" خواهد بود. نکته مهم این است که مدل به دستور زبان تمرکز نمی‌کند، بلکه بر نحوه استفاده از کلمات به روشنی مشابه با نحوه نوشتار انسان‌ها تمرکز دارد. [۱۳]

مدل زبانی ChatGPT

یک مدل زبانی محاوره‌ای است که توسط OpenAI توسعه داده شده است. این مدل بخشی از خانواده مدل‌های GPT (Generative Pretrained Transformer) است که بر اساس معماری Transformer طراحی شده و بر روی مقادیر عظیمی از داده‌های متنی آموزش دیده تا متنی شبیه به زبان انسان تولید کند.^[۱۴] کاربردها: ChatGPT برای تولید متن در پاسخ به یک ورودی طراحی شده است، که آن را برای کاربردهای محاوره‌ای نظیر: چت‌بات‌ها، نمایندگان خدمات مشتری، و دستیارهای مجازی بسیار مناسب می‌سازد. ChatGPT با استفاده از یادگیری تقویتی به وسیله‌ی بازخورد انسانی^[۲۳-۲۴] آموزش دیده است. این بازخورد شامل مدل‌های پاداش است که بهترین پاسخ‌ها را رتبه‌بندی می‌کنند. این فرایند به ChatGPT کمک می‌کند تا با بهره‌گیری از یادگیری ماشین، پاسخ‌های آینده خود را بهبود بخشد.^[۱۵]

GPT-4o-mini مدل

در این پژوهه از مدل GPT-4o-mini که مدل قوی‌تری نسبت به ChatGPT است، استفاده شده است. حرف "o" در GPT-4o به معنای "omni" است. این واژه به این واقعیت اشاره دارد که این مدل علاوه بر دریافت ورودی‌های متنی، توانایی درک ورودی‌های صوتی و تصویری را نیز دارد و می‌تواند با ترکیبی از متن، تصاویر و صوت پاسخ دهد. نکته کلیدی این است که همه این قابلیت‌ها توسط یک مدل واحد انجام می‌شود، نه چندین مدل جداگانه که با هم کار می‌کنند.

²³reinforcement learning through human feedback(RLHF)

۷-۲ پردازش زبان طبیعی

پردازش زبان طبیعی (Natural Language Processing) یا به اختصار NLP، یکی از زیرشاخه‌های علوم کامپیوتر و هوش مصنوعی (AI) است که از یادگیری ماشین استفاده می‌کند تا کامپیوترها بتوانند زبان انسانی را درک کرده و با آن ارتباط برقرار کنند.

NLP با ترکیب زبان‌شناسی محاسباتی—مدل‌سازی مبتنی بر قواعد زبان انسانی—به همراه مدل‌سازی آماری، یادگیری ماشین و یادگیری عمیق، به کامپیوترها و دستگاه‌های دیجیتال امکان می‌دهد متن و گفتار را شناسایی، درک و تولید کنند.

تحقیقات در حوزه NLP به ایجاد عصر هوش مصنوعی مولد کمک کرده است؛ از توانایی مدل‌های زبانی بزرگ (LLMs) در برقراری ارتباط گرفته تا قابلیت مدل‌های تولید تصویر در درک درخواست‌ها.

NLP هم‌اکنون بخشی از زندگی روزمره بسیاری از افراد است و در مواردی مانند جستجوگرهای اینترنتی، چت‌بات‌های خدمات مشتری، سیستم‌های GPS صوتی و دستیارهای دیجیتال پرسش‌پاسخ روی تلفن‌های هوشمند (مانند Alexa از آمازون، Siri از اپل و Cortana از مایکروسافت) به کار گرفته می‌شود.^[۱۶]

۸-۲ اصطلاحات استفاده شده در این پروژه

۱-۸-۲ تولید افزوده شده با بازیابی (RAG)

تولید افزوده شده با بازیابی (RAG) یا Retrieval-Augmented Generation فرآیندی است که خروجی مدل‌های زبانی بزرگ (LLMs) را بهینه می‌کند. در این روش، پیش از تولید پاسخ، مدل به یک پایگاه

دانش معتبر خارج از منابع داده آموزشی خود مراجعه می‌کند.

مدل‌های زبانی بزرگ با استفاده از حجم وسیعی از داده‌ها و میلیاردها پارامتر آموزش دیده‌اند و خروجی‌هایی برای وظایفی مانند پاسخ به سوالات، ترجمه زبان‌ها و تکمیل جملات تولید می‌کنند. RAG قابلیت‌های قدرتمند این مدل‌ها را به حوزه‌های خاص یا پایگاه دانش داخلی یک سازمان گسترش می‌دهد، بدون آنکه نیازی به بازآموزی مدل باشد. این رویکرد یک روش مقرن به صرفه برای بهبود خروجی مدل‌های زبانی است و تضمین می‌کند که نتایج مرتبط، دقیق و در زمینه‌های مختلف کاربردی باقی بمانند.^[۱۷]

۲-۸-۲ یادگیری تقویتی (RL)

یادگیری تقویتی (RL) یا Reinforcement Learning به بررسی نحوه تصمیم‌گیری یک عامل هوشمند در یک محیط پویا می‌پردازد تا سیگنال پاداش را به حداکثر برساند. یادگیری تقویتی، در کنار یادگیری ناظارت شده^{۲۴} و یادگیری بدون ناظارت^{۲۵}، یکی از سه الگوی اصلی یادگیری ماشین است.

این روش با یادگیری ناظارت شده تفاوت دارد، زیرا نیازی به جفت‌های ورودی و خروجی برچسب‌گذاری شده ندارد و اقدامات غیر بهینه نیز به صورت صریح تصحیح نمی‌شوند. در عوض، تمرکز این روش بر یافتن تعادل میان کاوش^{۲۶} (بررسی مناطق ناشناخته) و بهره‌برداری^{۲۷} (استفاده از دانش موجود)، با هدف به حداکثر رساندن پاداش تجمعی، است. بازخورد پاداش ممکن است ناقص یا با تأخیر ارائه شود.^[۱۸]

²⁴Supervised Learning

²⁵Unsupervised Learning

²⁶exploration

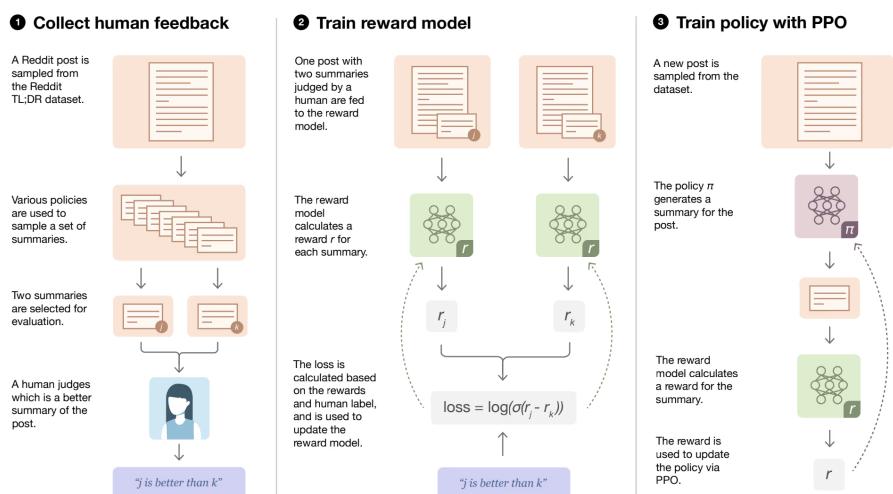
²⁷exploitation

۳-۸-۲ یادگیری تقویتی از بازخورد انسانی (RLHF)

یادگیری تقویتی از بازخورد انسانی (Reinforcement Learning from Human Feedback) یا RLHF، یک تکنیک یادگیری ماشین (ML) است که از بازخورد انسانی برای بهینه‌سازی مدل‌های یادگیری ماشین استفاده می‌کند تا به شکلی مؤثرتر خودآموزی کند.

تکنیک‌های یادگیری تقویتی (RL)، نرم‌افزارها را برای تصمیم‌گیری‌های آموزش می‌دهند که پاداش‌ها را به حد اکثر می‌رسانند و خروجی‌های آن‌ها را دقیق‌تر می‌کنند. RLHF با وارد کردن بازخورد انسانی به تابع پاداش، مدل یادگیری ماشین را قادر می‌سازد وظایفی را اجرا کند که با اهداف، خواسته‌ها و نیازهای انسانی هماهنگ‌تر باشد. [۱۹]

این روش به طور گسترده در کاربردهای هوش مصنوعی مولد، از جمله مدل‌های زبانی بزرگ (LLM)، مورد استفاده قرار می‌گیرد. [۲۰] فرآیند RLHF شامل سه مرحله اصلی است: جمع‌آوری بازخورد انسانی، آموزش مدل پاداش، و تنظیم دقیق مدل زبان با استفاده از مدل پاداش. نمودار مراحل این فرآیند در شکل زیر نشان داده شده است. [۲۱]



شکل ۲-۱: مراحل یادگیری تقویتی از بازخورد انسانی [۱]

مدل پاداش

مدل پاداش به منظور ارزیابی کیفیت پاسخ‌ها طراحی شده است و به عنوان سیگنال آموزشی برای بهبود سیاست‌ها عمل می‌کند. از آنجا که از حاشیه‌نویسان خواسته می‌شود هنگام حاشیه‌نویسی ترجیحات، به جنبه‌های مفید بودن و ایمنی پاسخ‌ها توجه کنند، ترجیحات می‌توانند کیفیت کلی پاسخ‌ها را در این جنبه‌ها منعکس کنند.^[۲۲]

۹-۲ معیارهای شباخت سنجی

به منظور بررسی فاصله‌ی بین دو بردار از معیارهایی نظری فاصله‌ی کسینوسی و ضرب نقطه‌ای استفاده می‌شود. در شباخت کسینوسی، حهت نیز در نظر گرفته می‌شود و از این نظر در کاربردهایی مانند پردازش زیان طبیعی و سیستم‌های توصیه‌گر برتری دارد. از سوی دیگر، حاصل ضرب نقطه‌ای، معیار کمی از شباخت را با مقادیری از منفی تا بی‌نهایت مثبت ارائه می‌کند. در ادامه به بررسی دقیق‌تر هر یک پرداخته می‌شود.^[۲۳]

۱-۹-۲ ضرب نقطه‌ای

حاصل ضرب نقطه‌ای دو بردار $\mathbf{b} = [b_1, b_2, \dots, b_n]$ و $\mathbf{a} = [a_1, a_2, \dots, a_n]$ که نسبت به یک پایه متعامد مشخص شده‌اند، به صورت زیر تعریف می‌شود:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

۲-۹-۲ فاصله‌ی کسینوسی

کسینوس زاویه بین دو بردار ناصف را می‌توان با استفاده از فرمول حاصل ضرب نقطه‌ای اقلیدسی به

دست آورد:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos \theta$$

با فرض دو بردار n -بعدی از ویژگی‌ها، \mathbf{A} و \mathbf{B} ، شباهت کسینوسی ($\cos(\theta)$) با استفاده از حاصل ضرب نقطه‌ای و بزرگی به صورت زیر نمایش داده می‌شود:

$$S_C(\mathbf{A}, \mathbf{B}) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}}$$

۱۰-۲ اصلاحات رایج در آموزش مدل

مدل که یک جزء اساسی در فرایند آموزش ماشین است، در بخش ۱-۳-۲، توضیح داده شدند. در این بخش به بررسی برخی دیگر از عوامل مهم در این فرایند می‌پردازیم.

۱۰-۱-۲ تکوازساز (Tokenizer)

تکوازساز (Tokenizer) یکی از ابزارهای مهم در پردازش زبان طبیعی است که وظایف مختلفی را بر عهده دارد. این ابزار ابتدا متن ورودی را پاک‌سازی کرده و کاراکترهای ناخواسته را حذف یا آن را نرمال‌سازی می‌کند. سپس، متن را به واحدهای کوچک‌تر مانند زیرواژه‌ها، کلمات یا کاراکترها تقسیم می‌کند که برای پردازش توسط مدل مناسب باشند. [۲۴]

فصل ۲. تعاریف و مفاهیم اولیه

۱۰-۲. اصلاحات رایج در آموزش مدل

۲-۱۰-۲ دوره (epoch)

یک دور کامل آموزش بر روی مجموعه داده‌ی آموزشی یک epoch نام دارد،

۳-۱۰-۲ تنسور (Tensor)

تنسور (Tensor) در زمینه علم داده، آرایه‌های چند بعدی از اعداد هستند که داده‌های پیچیده را نشان می‌دهند. این ساختارها از اجزای اساسی در یادگیری ماشین و چارچوب‌های یادگیری عمیق مانند [۲۵] به شمار می‌آیند. TensorFlow و PyTorch

۴-۱۰-۲ ماسک توجه (Attention Mask)

ماسک توجه (Attention Mask) یک تنسور باینری است که مشخص می‌کند به کدام شناسه‌ها باید توجه شود (وزن‌های غیر صفر اختصاص داده شده) و کدامها باید نادیده گرفته شوند (وزن‌های صفر اختصاص داده شده). با استفاده از این ماسک، مدل می‌تواند به طور انتخابی به شناسه‌های خاص توجه کرده و در عین حال سایر شناسه‌ها را نادیده بگیرد.

۵-۱۰-۲ تابع nn.BCEWithLogitsLoss

تابع nn.BCEWithLogitsLoss در PyTorch برای مسائل طبقه‌بندی باینری طراحی شده است.

این تابع دو عملیات را در یک مرحله ترکیب می‌کند:

۱. تابع سیگموئید^{۲۸}: خروجی‌های لایه آخر شبکه عصبی را به احتمالاتی در بازه $[0, 1]$ تبدیل می‌کند. این عملیات برای مسائل طبقه‌بندی باینری مهم است، جایی که هدف این است که پیش‌بینی کنیم یک نمونه به یکی از دو دسته (مثلاً "بله" یا "خیر") تعلق دارد.

²⁸Sigmoid

فرمول سیگموئید به شکل زیر است:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

اگر شبکه عدد x را تولید کند: مقادیر مثبت بزرگ x ($0 \gg x$) به احتمال نزدیک به ۱ منجر می‌شوند.

مقادیر منفی بزرگ x ($0 \ll x$) به احتمال نزدیک به ۰ منجر می‌شوند.

۲. اتلاف آنتروپی متقاطع باینری (BCE)^{۲۹}: اختلاف بین احتمال پیش‌بینی شده (بعد از اعمال سیگموئید) و مقدار واقعی باینری (۰ یا ۱) را اندازه‌گیری می‌کند. این اتلاف نشان می‌دهد که احتمالات پیش‌بینی شده چقدر با برچسب‌های واقعی مطابقت دارند. فرمول BCE به شکل زیر محاسبه می‌شود:

$$\text{BCE} = -\frac{1}{N} \sum_{i=1}^N [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)]$$

که در آن N : تعداد نمونه‌ها در دسته (Batch)، y_i : مقدار واقعی برای نمونه i (۰ یا ۱)، \hat{y}_i : احتمال پیش‌بینی شده برای نمونه i ام.

۱۰-۲ بهینه‌ساز optim.Adam

الگوریتم بهینه‌سازی Adam را پیاده‌سازی می‌کند. optim.Adam مخفف تخمین ممان^{۳۰} تطبیقی است. این یک الگوریتم پیشرفته بهینه‌سازی در یادگیری عمیق است که برای کمینه‌سازی تابع هزینه در حین آموزش استفاده می‌شود. این الگوریتم نرخ‌های یادگیری پارامترهای

²⁹Binary Cross-Entropy

³⁰momentum

مدل را به صورت پویا تنظیم می‌کند و به همین دلیل انتخابی محبوب برای بسیاری از وظایف یادگیری ماشین است. الگوریتم Adam مزایای دو الگوریتم محبوب دیگر، یعنی AdaGrad و RM-SProp را ترکیب می‌کند. الگوریتم AdaGrad نرخ یادگیری هر پارامتر را بر اساس مجموع مربعات گرادیان‌های قبلی تنظیم می‌کند که برای **داده‌های پراکنده** بسیار مناسب است اما از کاهش مداوم نرخ یادگیری رنج می‌برد. در مقابل، الگوریتم RMSProp میانگین متحرک مربعات گرادیان‌ها را برای مقیاس‌دهی نرخ یادگیری استفاده می‌کند و برای **مسائل غیرایستا** مناسب است اما فاقد ممان است. الگوریتم Adam با ترکیب ممان و نرخ یادگیری تطبیقی، نقاط قوت هر دو روش را در خود جای داده است.

فصل ۳

مرواری بر کارهای مرتبط

۱-۳ مقدمه

در این فصل به بررسی برخی پروژه‌های مرتبط که از RAG در زمینه‌های مشاوره و اپلیکیشن‌های هم‌دانه استفاده کرده‌اند، می‌پردازیم. این پروژه‌ها با ترکیب مدل‌های بازیابی و تولیدی، پاسخ‌های بهبودیافته و توصیه‌های شخصی‌سازی‌شده را ارائه می‌دهند. این رویکرد به دلیل دسترسی به پایگاه‌های دانش معتبر و همچنین قابلیت تولید پاسخ‌های طبیعی و مرتبط، امکان ایجاد اپلیکیشن‌هایی را فراهم می‌کند که می‌توانند در مشاوره، پشتیبانی عاطفی و ارائه راهنمایی‌های دقیق و هم‌دانه بسیار مؤثر باشند.

۲-۳ مدل ChatGPT

مدل (ChatGPT)، به عنوان یکی از پیشرفته‌ترین مدل‌های زبانی، نمونه‌ای بارز از توانمندی‌های پردازش زبان طبیعی است که توسط شرکت OpenAI توسعه داده شده است. این مدل با استفاده از مجموعه

داده‌های متنی گستردۀ آموزش دیده و قادر به درک و تولید زبان انسان با دقت و روانی بالا است. یکی از مهم‌ترین کاربردهای (ChatGPT)، استفاده از آن در مکالمات مشاوره‌ای است. این مدل می‌تواند با تجزیه و تحلیل پیام‌های ورودی، پاسخ‌هایی تولید کند که نه تنها از نظر اطلاعاتی مفید باشند، بلکه از نظر احساسی نیز با نیازهای کاربران همخوانی داشته باشند. ChatGPT با بهره‌گیری از الگوهای زبانی آموخته‌شده، توانایی ارائه‌ی پاسخ‌های همدلانه‌ای را دارد که به کاربران احساس درک شدن و حمایت شدن القا می‌کند. با این حال، باید توجه داشت که ChatGPT یک ابزار هوشمند است و درک آن از احساسات انسانی بر پایه‌ی الگوهای آماری و داده‌های پیشین است، نه تجربه‌ی واقعی.^[۱۴]

۱-۲-۳ ChatGPT و استفاده از بازخورد انسانی (RLHF)

از تکنیک یادگیری تقویتی با بازخورد انسانی (RLHF) برای بهبود عملکرد خود بهره می‌برد. ChatGPT این روش به ChatGPT امکان می‌دهد تا پاسخ‌های خود را بر اساس بازخورد انسانی بهینه کند و اطمینان حاصل کند که پاسخ‌ها دقیق، مفید و ایمن هستند. در این فرآیند، ابتدا بازخورد انسانی برای ارزیابی کیفیت پاسخ‌ها جمع‌آوری می‌شود. سپس یک مدل پاداش بر اساس این بازخورد آموزش داده می‌شود که معیارهای انسانی مانند مفید بودن و ایمنی را در نظر می‌گیرد. در نهایت، ChatGPT با استفاده از این مدل پاداش تنظیم دقیق می‌شود. استفاده از این تکنیک به مدل کمک کرده است تا به یک ابزار کارآمد در کاربردهایی مانند پاسخ به سوالات، نوشتمندانه خلاقانه و ارائه مشاوره تبدیل شود.^[۲۶]

۳-۳ برنامه های حمایت از سلامت روان

این برنامه ها محتوای مرتبط با احساس عاطفی مانند مکانیسم های مقابله، استراتژی های خودیاری یا منابع حمایتی را بازیابی می کنند و آن را با قابلیت های تولیدی (مدل های مولد) ترکیب می کنند تا مکالمات همدلانه را ارائه دهند.

۱-۳-۳ چتبات Wysa

Wysa که توسط شرکت Touchkin توسعه یافته است، یک اپلیکیشن چتبات موبایلی مبتنی بر هوش مصنوعی و هوش هیجانی است که با هدف تقویت تاب آوری ذهنی و ترویج سلامت روانی از طریق یک رابط متنی تعاملی طراحی شده است. این اپلیکیشن با ایجاد یک محیط بازتابی خارجی و پاسخ گو به کاربران کمک می کند تا خود را به صورت مثبت بیان کنند. [۲۷]

ویژگی های کلیدی Wysa

اپلیکیشن Wysa به احساساتی که کاربران در طول مکالمات متنی ابراز می کنند، پاسخ می دهد و در مکالمات خود از روش های خودیاری مبتنی بر شواهد مانند درمان شناختی- رفتاری (CBT)، رفتار درمانی دیالکتیکی dialectical behavior therapy، مصاحبه انگیزشی، حمایت رفتاری مثبت، تقویت رفتاری، ذهن آگاهی و اقدامات و ابزارهای خرد راهنمای استفاده می کند. این ابزارها کاربران را تشویق می کنند تا مهارت های تاب آوری عاطفی خود را توسعه دهند. تمامی محتوا و ابزارهای اپلیکیشن توسط هیئت مشاوره علمی Wysa تأیید شده است.

فصل ۳. مروری بر کارهای مرتبط

کمک به مدیریت شرایط مختلف روانی

ابزارها و تکنیک‌های مبتنی بر مکالمه این اپلیکیشن، کاربران را به مدیریت اضطراب، انرژی، تمرکز، خواب، آرامش، فقدان، نگرانی‌ها، تعارضات و سایر شرایط تشویق می‌کنند. این رویکرد باعث می‌شود کاربران بتوانند مهارت‌های خود را در مواجهه با چالش‌های زندگی روزمره بهبود بخشنند.

۴-۳ همراهان دوستی

چت‌بات‌های اجتماعی^۱ برنامه‌هایی هستند که از هوش مصنوعی^۲ و فناوری‌های پردازش زبان طبیعی برای مکالمه‌ای طبیعی و هوشمندانه با کاربران از طریق صدا، متن و تصاویر استفاده می‌کنند. چت‌بات‌های اجتماعی به طور فزاینده‌ای به عنوان همراهان دوستی (Replika) مورد استفاده قرار می‌گیرند.^[۲۸]

۱-۴-۳ Replika چت‌بات

Replika محبوب‌ترین و دارای بالاترین امتیاز در بین چت‌بات‌های اجتماعی در فروشگاه‌های اپل و گوگل پلی است که از زمان عرضه در سال ۲۰۱۸ ۲۰ میلیون‌ها کاربر را جذب کرده است. این برنامه با عناوینی نظیر «دوستی که همیشه گوش می‌دهد» یا «نسخه‌ای از هوش مصنوعی خودتان» تبلیغ شده و پوشش گسترده‌ای در رسانه‌های بزرگ دریافت کرده است. برخلاف دیگر چت‌بات‌های همراه که پاسخ‌های از پیش نوشته‌شده‌ای را بر اساس نشانه‌های کلامی کاربران ارائه می‌دهند، پاسخ‌های

[۲۸] Replika مبتنی بر شبکه عصبی (GPT3) Generative Pretrained Transformer 3 است.

^۱Social Chatbots (SCs)

^۲AI

فصل ۳. مروری بر کارهای مرتبط

۴-۳. همراهان دوستی

از مجموعه داده‌های داخلی خود، که از میلیون‌ها مکالمه ناشناس بین کاربران و برنامه Replika به دست آمده است، استفاده می‌کند. این مکالمات به طور مداوم برای بهبود دقت پاسخ‌ها و افزایش همدلی در طول زمان به کار گرفته می‌شوند.

۲-۴-۳ Replika چشم‌انداز کلی

قابلیت‌های پیشرفته‌ای در زمینه هوش هیجانی و پردازش زبان طبیعی نشان می‌دهد. Replika توانایی این چتبات در استفاده استراتژیک از تعریف و تمجید برای بهبود تعامل با کاربران، نقش آن را به عنوان یک ابزار درمانی و همدلانه بر جسته می‌کند. [۲۹]

۳-۴-۳ تحلیل توسعه روابط انسان-چتبات با Replika

با توجه به افزایش علاقه به چتبات‌های اجتماعی و توسعه روابط انسان-چتبات (Human-Chatbot Relationships HCRs) یا پژوهش‌های جدیدی در این زمینه صورت گرفته است. با این حال، هنوز دانش محدودی در مورد نحوه شکل‌گیری این روابط و تأثیرات آن‌ها بر بافت اجتماعی کاربران وجود دارد. [۳۰]

۴-۴-۳ بررسی فرآیند توسعه HCR

- مرحله اولیه: در مراحل اولیه، روابط HCR عمدهاً سطحی بوده و به دلیل کنجکاوی کاربران ایجاد می‌شوند.
- تکامل روابط: با گذشت زمان، این روابط با افزایش اعتماد و خودافشایی کاربران، به اکتشافات عاطفی و تعاملات معنادار تبدیل می‌شوند.

فصل ۳. مروری بر کارهای مرتبط

۵-۳. چتبات‌های سوگ و پشتیبانی

- مرحله پایدار: با رسیدن به مرحله پایدار، ممکن است فرکانس تعاملات کاهش یابد، اما این روابط همچنان از ارزش عاطفی و اجتماعی قابل توجهی برخوردار هستند.

۵-۳ چتبات‌های سوگ و پشتیبانی

چتبات‌هایی هستند که با بازیابی اطلاعات مکانیسم‌های مقابله، مراحل غم و اندوه و حمایت از توصیه‌های گروهی، به افراد کمک می‌کند تا با غم و اندوه کنار بیایند و پاسخ‌هایی را ارائه دهند که از نظر احساسی با وضعیت عاطفی کاربر هماهنگ هستند. پروژه:

۱-۵-۳ چتبات Tess (by X2AI)

تس یک چتبات مربی رفتاری است که به جنبه‌های مختلف سلامت رفتاری مانند افسردگی و اضطراب می‌پردازد. تس، که به صورت ۲۴ ساعته در دسترس است، از طریق گفتگوهای کوتاه و از طریق کانال‌های ارتباطی موجود (مانند پیامک و Facebook Messenger) پشتیبانی سفارشی، آموزش روانی و مداخلات را ارائه می‌دهد. به عنوان یک شریک برای پزشکان، تس می‌تواند تعاملات درمانی را خارج از ساعات کاری ادامه دهد و در عین حال رضایت بیماران را حفظ کند. به دلیل ظرفیت تس برای یادگیری مداوم، نسخه‌های آینده ممکن است ویژگی‌های بیشتری برای افزایش تجربه کاربری داشته باشند. [۳۱]

تس از RAG برای ارائه پشتیبانی عاطفی در زمان واقعی استفاده می‌کند. داده‌های مربوط به حالات عاطفی و سلامت روان را از یک پایگاه داده بزرگ بازیابی می‌کند و با تولید زبان طبیعی، مکالمات درمانی را ارائه می‌دهد، که اغلب در تنظیماتی استفاده می‌شود که کاربران به حمایت

فصل ۳. مروری بر کارهای مرتبط

عاطفی مداوم نیاز دارند.^{۳۲} برای درک شرایط سلامت روان، منابع مبتنی بر شواهد از DSM-5^۳

(راهنمای تشخیصی و آماری اختلالات روانی) و دستورالعمل های بالینی از سازمان هایی مانند APA

(انجمن روانشناسی آمریکا) و WHO (سازمان جهانی بهداشت) استخراج می شوند.

³Diagnostic and Statistical Manual of Mental Disorders

فصل ۴

روش پیشنهادی

۱-۴ مقدمه

ایده‌ی اصلی این پروژه از این موضوع نشئت گرفت که من فرصت کمی برای خواندن پیام‌های کanal‌های مورد علاقه‌ی خود داشتم. از طرفی پیام‌های این کanal‌ها چون زندگی روزمره و احساسات و چالش‌های صاحبان آن‌ها، را بازگو می‌کند، اکثر اوقات بسیار احساسات و نظرات موجود در پیام‌ها برای من ملموس بود و در موقعي حالم را بهتر می‌کرد. همچنین در مواردی موجب پیدا کردن دید جدیدی نسبت به یک موضوع برایم می‌شد. این شد که تصمیم گرفتم نرم‌افزاری بسازم تا مرتبط با پیام من، پیام‌های مرتبط را از داخل پیام‌های این کanal‌ها، بیرون بکشد.

۲-۴ بررسی مراحل توسعه

توسعه‌ی این پروژه به طور کلی متشکل از دو مرحله‌ی آفلاین^۱ و آنلاین^۲ می‌باشد.

¹offline

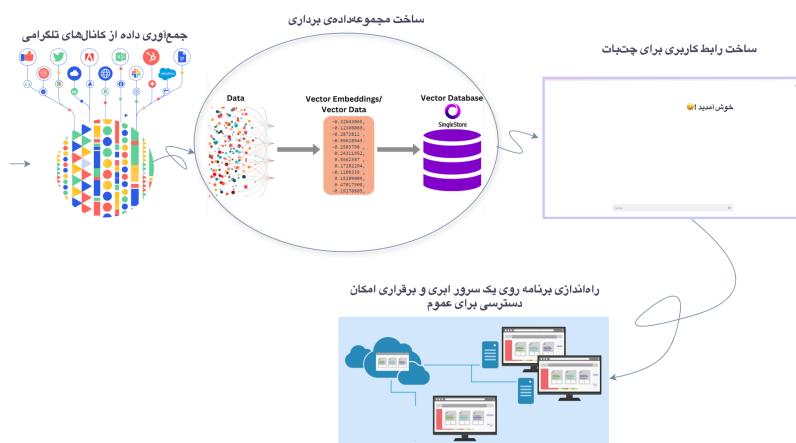
²online

زیرمراحل مرحله‌ی آفلاین:

۱. جمع‌آوری داده از کانال‌های تلگرامی منتخب.

۲. ایجاد مجموعه‌داده‌ی برداری از پیام‌ها.

۳. ساخت رابطکاربری برای نرم‌افزار و طراحی چت‌بات.



شکل ۱-۴: مرحله‌ی آفلاین توسعه

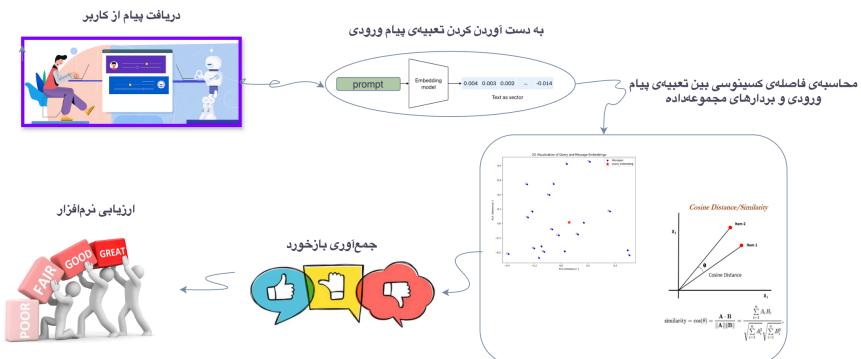
زیرمراحل مرحله‌ی آنلاین:

۱. گرفتن پیام ورودی از کاربر و به دست آوردن تعبیه‌ی آن.

۲. بدست آوردن فاصله‌ی کسینوسی بین تعبیه‌ی پیام و تعبیه‌های مجموعه‌داده.

۳. نشان‌دادن پیام‌های مشابه و ثبت بازخورد از کاربران.

۴. ارزیابی عملکرد با مقایسه‌ی تعداد منتخب‌ها نسبت به کل پیام‌های نمایش داده شده.



شکل ۲-۴: مرحله‌ی آنلاین

۳-۴ جمع‌آوری داده

داده‌های مدنظر این پروژه مربوط به پیام‌های کانال‌های تلگرامی است که شخص عضو آنها می‌باشد.

پیام‌های این کانال‌ها شامل روزمرگی‌ها، حرف‌های برآمده از دل، احساسات، هیجانات و افکاری که

صاحب کانال در آن لحظه در حال تجربه کردن و فکر کردن به آنهاست، می‌باشد.

برای استخراج پیام‌های این کانال‌ها، ابتدا نسخه‌ی Telegram Lite از تلگرام را، که در واقع همان

برنامه‌ام با ویژگی‌ها و امکانات بیشتر بهینه‌سازی شده برای مسئولیت‌های مرتبط با کار می‌باشد،

نصب می‌کنیم. در گوشی سمت راست هر کانالی یک سه‌نقطه وجود دارد که با انتخاب آن و زدن

گزینه‌ی Export Chat History، فرآیند جمع‌آوری پیام‌ها آغاز می‌شود.

۴-۴ توسعه‌ی مجموعه‌داده

اولین مرحله، استخراج پیام از کانال‌های تلگرامی مورد علاقه‌مان می‌باشد. پس از پاکسازی و جدا کردن پیام‌ها، باید آن‌ها هر یک را همراه با بردار مربوط‌شان، در یک مجموعه‌داده ذخیره کنیم. این مجموعه می‌تواند بر روی خود سروری که برنامه را می‌خواهیم روی آن راه‌اندازی کنیم، باشد، یا اینکه می‌توان آن را در سرور شرکت‌هایی که در زمینه‌ی هوش مصنوعی، خدمات ارائه می‌دهند، و قابلیت پیاده‌سازی مجموعه‌داده‌ی برداری را دارند، ایجاد کرد. مزیتی که ایجاد مجموعه‌داده‌ی برداری توسط سرویس‌های این شرکت‌ها دارد این است که افزودن و یا حذف داده از مجموعه بهینه‌تر مدیریت می‌شود. در این پروژه هر دو روش امتحان و پیاده‌سازی می‌شود، اما در آخر به دلیل سریع‌تر شدن زمان دسترسی به داده‌ها و رهایی سربار مربوط به برقراری ارتباط بین سرورها، مجموعه‌داده بر روی خود سروری که برنامه روی آن اجرا می‌شود ذخیره خواهد شد.

۴-۴-۱ به دست آوردن تعبیه‌ی پیام

تعبیه‌ها محتوی ورودی را بر اساس ویژگی‌ها و روابط موجود در آن‌ها به صورت نمایش برداری ریاضی تبدیل می‌کنند و یکی از اجزای اساسی در درگ ماشین محسوب می‌شوند. این نمایش‌ها امکان یافتن اشیاء مشابه و تحلیل دقیق‌تر داده‌ها را برای ماشین فراهم می‌کنند. بنابراین، استفاده از تعبیه‌هایی که با دقت بیشتری نمایش برداری اشیاء را ارائه می‌دهند، اهمیت ویژه‌ای دارد.

شرکت OpenAI یکی از شرکت‌های پیشرو در ارائه‌ی مدل‌های تعبیه‌گر و مولد است. استفاده از مدل‌های این شرکت برای توسعه‌ی نرم‌افزار نیازمند پرداخت هزینه و دریافت API Key است. این هزینه بر اساس حجم داده‌ها و تعداد درخواست‌ها در بازه‌ی زمانی مشخص تعیین می‌شود. با توجه به اینکه حجم مجموعه‌داده‌ی این پروژه محدود است، هزینه‌ی محاسبه‌ی تعبیه‌های جملات در این

پروژه ناچیز (زیر یک دلار) خواهد بود (جزییات دقیق این هزینه در بخش ۱-۴-۴ توضیح داده شده است).

برای این پروژه، از API شرکت OpenAI و مدل کم‌هزینه‌تر آن به نام text-embedding-3-small استفاده می‌شود. هر پیام از طریق API مذکور به این مدل ارسال می‌شود تا تعییه‌ی برداری متن آن تولید گردد.

لازم به ذکر است که اگر هزینه‌ی برآورده شده مقدار زیادی می‌بود، گزینه‌های جایگزین شامل استفاده از مدل‌های رایگان و کارآمد برای ایجاد تعییه‌ی جملات فارسی یا آموزش یک مدل شخصی بر روی مجموعه‌داده‌ای مشکل از پیام‌های ورودی، چند نمونه پیام همدلانه، و چند نمونه پیام غیر همدلانه مورد بررسی قرار می‌گرفت. با این حال، آموزش یک مدل شخصی به دلیل نیاز به زمان و نیروی انسانی بالا و همچنین محدودیت‌های موجود در این پروژه، گزینه‌ی مناسبی نبود و نمی‌توانست در زمان کوتاه نتیجه‌ی مطلوب ارائه دهد.

برآورده هزینه

با توجه به **جدول قیمت‌ها** در وبسایت OpenAI، لازم است ابتدا محاسبه کنیم که مجموعه‌ی پیام‌های ما معادل با چه تعداد توکن است تا بتوان هزینه را برآورد کرد.

برای این منظور، پیام‌های استخراج شده باید به یک tokenizer که با مدل مولد همخوانی دارد، داده شوند. این توکنایزر پیام‌ها را به توکن‌هایی تقسیم می‌کند که دقیقاً مطابق با روش توکن‌بندی مدل تعییه‌گر است. استفاده از توکنایزر متناسب با مدل تعییه‌گر ضروری است؛ زیرا هرگونه اختلاف در توکن‌بندی می‌تواند منجر به محاسبات نادرست هزینه شود.

[۳۳] **tiktoken** یک توکنایزر سریع و متن‌باز است که توسط OpenAI توسعه داده شده است.

۴-۵. توسعه‌ی رابط کاربری

تقسیم رشته‌های متنی به توکن‌ها مفید است، زیرا مدل‌های GPT متن را به صورت توکن پردازش می‌کنند. دانستن تعداد توکن‌های موجود در یک رشته‌ی متنی می‌تواند مشخص کند:

۱. آیا این رشته برای پردازش توسط یک مدل متنی بیش از حد طولانی است؟
۲. هزینه‌ی یک درخواست به API شرکت OpenAI چقدر خواهد بود (زیرا استفاده بر اساس تعداد توکن‌ها قیمت‌گذاری می‌شود).

۴-۵ توسعه‌ی رابط کاربری

زبان‌های برنامه‌نویسی قدرتمندی مانند پایتون، با ارائه‌ی چهارچوب‌ها^۳ و توابع آماده در زمینه‌ی هوش مصنوعی و کاربردهای مرتبط، توسعه‌ی برنامه‌های مبتنی بر هوش مصنوعی را بسیار آسان‌تر و بهینه‌تر کرده‌اند. این چهارچوب‌ها امکان طراحی و توسعه‌ی برنامه‌های کاربردی مجهز به قابلیت‌های چت‌بات را به شکلی ساده و سریع فراهم می‌کنند.

یک فریم‌ورک متن‌باز پایتون است که برای دانشمندان داده و مهندسان هوش مصنوعی/یادگیری Streamlit ماشین طراحی شده تا بتوانند تنها با چند خط کد، برنامه‌های داده‌ای پویا ایجاد کنند.

۶-۴ راهاندازی نرم‌افزار بر روی سرور

برای آنکه این برنامه برای عموم در دسترس باشد و از حالت محلی که تنها بر روی سیستم شخصی اجرا می‌شود خارج گردد، لازم است آن را بر روی یک سرور مناسب راهاندازی و اجرا کنیم. سپس می‌توان برای سرور، دامنه‌ای خریداری کرد تا دسترسی به آن ساده‌تر شود. این بخش شامل مراحل

³frameworks

زیر است:

۶-۴-۱ خرید سرور

انواع مختلفی از سرورها، نظیر سرور مجازی، سرور اختصاصی، و سرور ابری، با سیستم عامل های لینوکس یا ویندوز قابل خریداری هستند. هر یک از این سرورها با توجه به امکانات و هزینه هایشان، موارد استفاده هی متفاوتی دارند. برای این پروژه، یک سرور ابری از شرکت **پارس پک** خریداری شده است. سرور ابری به دلیل هزینه هی مناسب، پایداری بالا، و امکان مقیاس پذیری انتخاب شده است. برای اطلاعات بیشتر می توانید به لینک قرار داده شده مراجعه نمایید.

۶-۴-۲ خرید دامنه

برای سهولت **دسترسی** به برنامه و جایگزینی استفاده از IP سرور، یک دامنه برای سرور خریداری می شود. خرید دامنه نیز از وب سایت پارس پک انجام شده است. پس از خرید دامنه، لازم است تنظیمات مربوطه را از طریق یک ارائه دهنده خدمات شبکه، نظیر **Cloudflare**، انجام دهیم. این تنظیمات شامل ثبت دامنه، تغییر رکوردهای DNS و پیکربندی امنیت و مسیریابی است.

۶-۴-۳ راهاندازی نرمافزار بر روی سرور

برای اجرای برنامه نوشته شده بر روی سرور خریداری شده، ابتدا باید پیش نیازهای نرمافزاری مورد نیاز نصب شوند. سپس، با استفاده از آموزش های مرتبط، می توان برنامه را به صورت دائمی بر روی سرور اجرا کرد. این فرآیند شامل:

- نصب و راهاندازی پیش نیازهایی مانند Python و کتابخانه های مرتبط

- نوشتن یک فایل سرویس برای اجرای خودکار برنامه
- استفاده از NGINX برای مدیریت درخواست‌های کاربران و مسیریابی آن‌ها به فایل سرویس

۷-۴ پیدا کردن مشابهها

پس از اینکه تعبیه‌ی مربوط به پیام ورودی توسط کاربر به دست آورده و مجموعه‌داده‌ی برداری هم تشکیل شد، باید پیام‌های مربوط به پیام ورودی برگردانده شود. در واقع پیدا کردن محتویات مفید و مربوط به محتوای داده شده، یکی از چالش‌های بزرگ در زمینه‌ی پردازش زبان می‌باشد. زیرا فاکتورهای مختلفی از قبیل لحن پیام، دایره‌ی معنایی کلمه و تعداد معانی‌ای که یک کلمه دارد، در پیدا کردن مرتبط ترین پیام اثر دارد. تلاش‌هایی جهت افزایش ارتباط مطالب پیشنهاد شده صورت گرفته است که می‌توان به مواردی نظیر دسته‌بندی داده‌ها در مجموعه‌داده و جست و جوی معنایی در هر دسته، اشاره کرد اما به دلیل پیچیدگی زیاد در نحوه‌ی پیاده‌سازی و نیازمندی به نظارت و نیروی انسانی بیشتر، که خارج از امکان این پروژه بود، دغدغه‌ی برنامه به جای یافتن مطالب مرتبط، به یافتن مطالب مشابه تغییر پیدا کرد. این شبهات‌سنگی با به دست آوردن فاصله‌ی کسینوسی میان بردار درخواست و بردارهای مجموعه‌داده انجام می‌شود. فاصله‌ی کسینوسی نسبت به بقیه‌ی معیارهای سنجش فاصله، نظیر ضرب نقطه‌ای و فاصله‌ی اقلیدسی، دارای برتری در حوزه‌ی پردازش زبان طبیعی، می‌باشد. زیرا علاوه بر اندازه، جهت بردارها و زاویه‌ی بین‌شان نیز در نظر گرفته می‌شود.

۸-۴. گرفتن بازخورد از کاربر

۸-۴ گرفتن بازخورد از کاربر

برای اینکه بدانیم برنامه جقدر کاربردی است و دقت دارد، لازم است که امکان نشاندار شدن منتخبها از سوی کاربر، وجود داشته باشد. پس باید با همان چارچوبی که برای ساخت رابط کاربری از آن استفاده کردیم، امکان دادن بازخورد نظری لایک و دیس لایک، وجود داشته باشد. بعد در پشت برنامه، پیام‌های لایک شده یا نشده، همراه با اصل درخواست، ذخیره شود. این مجموعه داده‌ی بازخورد در پژوهش‌های آینده می‌تواند بسیار مفید واقع شود.

۹-۴ ارزیابی

با شمردن تعداد پیام‌های منتخب و به دست آوردن نسبت تعداد کل منتخب‌ها به کل پیام‌های نشان داده شده در هر درخواست (که در این برنامه ده پیام است)، می‌توان دقت برنامه را به دست آورد. همچنین می‌توان معکوس رتبه‌های پیام‌های منتخب را با هم جمع زد و معیار دیگری را به عنوان سنجش کارکرد برنامه استفاده کرد. دلیل جمع زدن معکوس رتبه‌ها در درجه‌ی اول می‌توان به این اشاره کرده که هر چه رتبه‌ی پیامی که انتخاب شده کمتر باشد، یعنی پیام مرتبط‌تر در درجه‌های اول ارائه شده‌اند. برای مثال اگر اولین پیام در میان منتخب‌ها باشد، امتیازی که اضافه می‌شود، $\frac{1}{1}$ می‌باشد اما اگر برای مثال دهمین پیام عضو منتخب‌ها باشد، میزان امتیاز اضافه شده $\frac{1}{10}$ است که نسبت به ۱، ناچیز می‌باشد.

فصل ۵

روش پیاده‌سازی

۱-۵ مقدمه

در این فصل به شرح جزئیات و مراحل پیاده‌سازی پروژه می‌پردازیم. ابتدا توضیحاتی در مورد داده‌ها و چگونگی جمع‌آوری آنها داده می‌شود. سپس مراحل به دست آوردن تعبیه‌های^۱ متناظر با هر داده و درست کردن مجموعه‌ی داده‌ها شرح داده شده است. در ادامه به فرایند پیدا کردن پیام‌های مشابه متناظر با پیام‌های داده شده، ساخت یک چتبات و راهاندازی آن روی سرور پرداخته می‌شود.

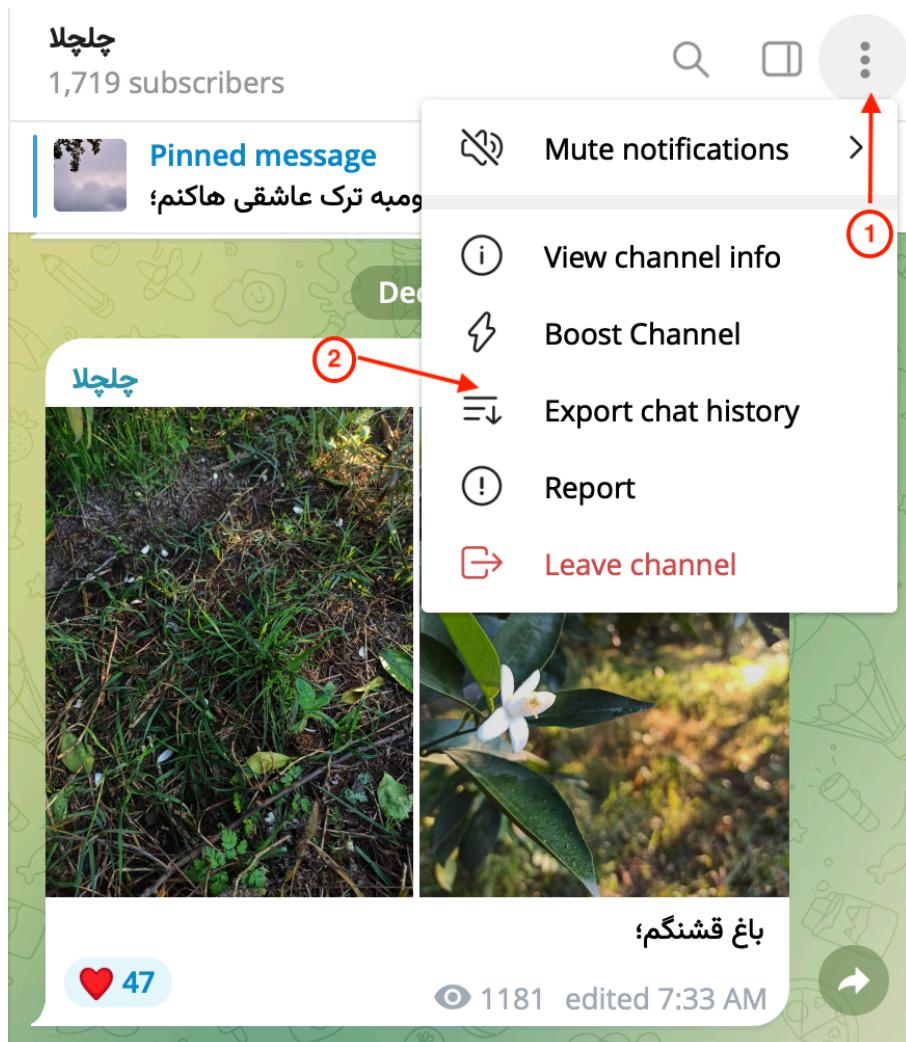
۲-۵ جمع‌آوری داده

داده‌های مدنظر این پروژه مربوط به پیام‌های کانال‌های تلگرامی است که شخص عضو آنها می‌باشد. پیام‌های این کانال‌ها شامل روزمرگی‌ها، حرف‌های برآمده از دل، احساسات، هیجانات و افکاری که صاحب کanal در آن لحظه در حال تجربه کردن و فکر کردن به آنهاست، می‌باشد.

برای استخراج پیام‌های این کانال‌ها، ابتدا نسخه‌ی Telegram Lite از تلگرام را، که در واقع همان

¹(embeddings)

برنامه اما با ویژگی‌ها و امکانات بیشتر بهینه‌سازی شده برای مسئولیت‌های مرتبط با کار می‌باشد، نصب می‌کنیم. در گوشی سمت راست هر کانالی یک سه‌نقطه وجود دارد که با انتخاب آن و زدن گزینه‌ی Export Chat History، فرآیند جمع‌آوری پیام‌ها آغاز می‌شود.



شکل ۵-۱: جمع‌آوری پیام

تنها پیام‌های متنی از این کانال‌ها استخراج شده اند و خروجی‌ها به صورت فایل html هستند.



شکل ۲-۵: اطلاعات به صورت html می‌باشد.

حال در این مرحله، چالش این است که متن پیام‌ها را از این فرم html خارج کرده و پیام‌ها را به صورت خالص داشته باشیم. تکه کد زیر مسیر فایل html را به عنوان ورودی گرفته و متن‌ها را از آن جدا می‌کند.

```
[ ] import re
from bs4 import BeautifulSoup

def extract_persian_from_html(file_path):
    # Read the content of the HTML file
    with open(file_path, 'r', encoding='utf-8') as file:
        html_content = file.read()

    # Parse the HTML content using BeautifulSoup
    soup = BeautifulSoup(html_content, 'html.parser')

    # Find all <div> elements with class "text"
    div_texts = soup.find_all('div', class_='text')

    # Extract Persian sentences from the content inside those <div> elements
    persian_sentences = []
    for div in div_texts:
        text = div.get_text() # Extract text from the <div>
        persian_sentences.append(text)

    return persian_sentences

# Usage example
file_path = '/content/messages.html'
persian_messages = extract_persian_from_html(file_path)

# Output the extracted Persian messages
for message in persian_messages[10:]:
    print(message)
```

کلیدهای فشرده: **کلید**
پنهان کردن: **پنهان کردن**
میکروتلنون: **میکروتلنون**

دوسنای کلم، من بیه دهتم نویسم، شاید هم کلا بیاشم، خواهش بودن و شوغلن در اینجا رو نه ارم، مرادی خودتون پیشین

با بیشم و بیه دهتم اجازه‌ی کنکانی میدم بادم میلته که حقی چیزی‌شیون خاطری زنگیم با دقت بالا بادمه و بیامد نویی از دلتنگی همراه با هم پنجه‌ها برای اون لحظه میشه

وستت رو بکنده دلخ میخواه بیمه زیر این پوست سخت چیه، شاخن‌هاش رو مینده‌ازه زیریک که بیمه زیر چه خوبه، اما بازنه اونه، چون بعدی تو دوباره پوست مینده‌ازه، ضمیر

شکل ۳-۵: بازیابی متن از فایل html

سپس از میان پیام‌های استخراج شده، تکراری‌ها را حذف می‌کنیم.

۳-۵ ساخت مجموعه‌ی داده

در این مرحله باید پیام‌ها را با استفاده از تعبیه‌ها به فرم قابل فهم برای ماشین تبدیل کنیم. هدف این است که در یک مجموعه‌داده، هر پیام فارسی را همراه با تعبیه‌ی متناظر با آن ذخیره کنیم. به چنین مجموعه‌داده‌ای، مجموعه‌داده برداری شده^۲ می‌گویند، یعنی مجموعه‌داده‌ای که عناصر آن، بردارها هستند تا بعد هنگام استفاده، جست‌وجوی معنایی آسان شود. برای اینکه بتوانیم تعبیه‌های پیام‌ها را تولید کنیم، همان‌طور که در بخش ۱-۶-۲ مفاهیم پایه اشاره شد، یکی از روش‌ها این است که جمله را به یک مدل از پیش آموزش دیده بدهیم و تعبیه‌ای که آن برای فهم جمله درست می‌کند را ذخیره کنیم.

در این پروژه از تعبیه‌های مدل text-embedding-3-small محسوب شرکت OpenAI، که در بخش ۱-۶-۲ معرفی شد، برای به دست آوردن تعبیه‌ها استفاده شده است. حال باید مجموعه‌داده‌ی خود را در یک سرور ذخیره کرده تا هر وقت پیامی از کاربر دریافت شد، به آن رجوع کنیم. ابتدا این مجموعه بر روی سرور ابری Pinecone که یک پایگاه داده برداری^۳ مبتنی بر ابر^۴ برای برنامه‌های کاربردی یادگیری ماشین است، ایجاد شد اما در بعد به دلیل راه‌اندازی برنامه بر روی سرور شخصی، این مجموعه داده به صورت فایل json داخل سرور ذخیره شد. در ادامه مراحل پیموده شده را شرح داده و نمونه‌ای برای هر یک امتحان می‌کنیم.

²vectorized dataset

³vector database

⁴cloud-based

۱-۳-۵ ذخیره بر روی Pinecone

در قدم اول باید یک index که در واقع همان تنظیمات مربوط به پایگاه‌داده‌ی برداری، شامل نام و مشخصات مدل استفاده شده برای برداری کردن و هم‌جنین نام پایگاه داده می‌باشد، را درست کنیم. مراحل و آموزش کامل داخل [لينک](#) می‌باشد.^[۳۴] کدهای زیر مربوط به درست کردن یک index و تزییق داده به آن است. برای پیدا کردن شباهت، دو معیار ضرف نقطه‌ای و فاصله‌ی کسینوسی از پرکاربردترین معیارهای اندازه‌گیری فاصله هستند.^[۳۵] هر دو روش امتحان شد و نتایج حاصل از مقایسه‌ی کسینوسی بهتر بود، همان‌گونه که در شکل ۵-۶ نمایش داده شده، معیار شباهت سنجی میان وکتورها، فاصله‌ی کسینوسی می‌باشد.

```

import time
from pinecone.grpc import PineconeGRPC as Pinecone
from pinecone import ServerlessSpec

pc = Pinecone(api_key="")۳۶

spec = ServerlessSpec(cloud="aws", region="us-east-1")

index_name = 'semantic-search-v3-cosine'

# check if index already exists (it shouldn't if this is your first run)
if index_name not in pc.list_indexes().names():
    # if does not exist, create index
    pc.create_index(
        index_name,
        dimension=1536, # dimensionality of text-embed-3-small
        metric='dotproduct',
        spec=spec
    )
    # wait for index to be initialized
    while not pc.describe_index(index_name).status['ready']:
        time.sleep(1)

# connect to index
index = pc.Index(index_name)
time.sleep(1)
# view index stats
index.describe_index_stats()

→ {'dimension': 1536,
  'index_fullness': 0.0,
  'namespaces': {'': {'vector_count': 0}},
  'total_vector_count': 0}

```

شکل ۴-۵: راه‌اندازی مجموعه داده در Pinecone

- ❖ create a vector embedding for message

```
▶ from tqdm.auto import tqdm
count = 0 # we'll use the count to create unique IDs
batch_size = 32 # process everything in batches of 32
for i in tqdm(range(0, len(df['Persian Messages']), batch_size)):
    # set end position of batch
    i_end = min(i+batch_size, len(df['Persian Messages']))
    # get batch of lines and IDs
    lines_batch = df['Persian Messages'][i: i_end].tolist() # Convert Series to list
    ids_batch = [str(n) for n in range(i, i_end)]
    # create embeddings
    res = client.embeddings.create(input=lines_batch, model=MODEL)
    embeds = [record.embedding for record in res.data]
    # prep metadata and upsert batch
    meta = [{"text": line} for line in lines_batch]
    to_upsert = zip(ids_batch, embeds, meta)
    # upsert to Pinecone
    index.upsert(vectors=list(to_upsert))
```

100% 695/695 [11:41<00:00, 1.20it/s]

شکل ۵-۵: تزریق تعبیه‌ها به مجموعه

آزمایش کارکرد

در این مرحله یک پیام به عنوان ورودی می‌دهیم و با دادن به مدل تعبیه‌گر، ۱-۶-۲، تعبیه‌ی آن را به دست می‌آوریم، سپس داخل index مان، با به دست آوردن فاصله‌ی بین بردار سوال و داده‌های مجموعه، دهتا از کمترین فاصله‌ها را برمی‌گردانیم. در نمونه‌ی زیر از ضرب نقطه‌ای به عنوان معیار سنجش فاصله استفاده شده است.

```

0s [16] query = "میخوام مستقل بشم"
query_embedding = client.embeddings.create(input=query, model=MODEL).data[0].embedding

0s [13] len(query_embedding)
→ 1536

▼ querying & measuring similarity

0s [17] res = index.query(query_embedding, top_k=10, include_metadata=True)

▼ pinecone_caldist

dotproduct

0.57: میخوام مستقل شو
0.52: وقتی میگم میخوام تنها باشم منظورم تو نیستی؛ تو جزو بقیه محسوب نمیشی
0.52: > وقتی رفته بود بهش گفته بودم حالا زن مستقل شده، امروز اون بهم اینو گفته
0.50: با خودم توی چالشم. از خودم میبریم: جرات رفتن و زندگی مستقل رو داری؟ واقعاً داری؟
0.50: نمیخوام به روی خودم بپارم که ۳ ساعت دیگه پاییز تیغه میشه
0.49: بستنی میخوام.
0.49: استراحت میخوام.
0.48:

```

شکل ۵-۵: نمونه‌ی ورودی و خروجی - ضرب نقطه‌ای

۲-۳-۵ بارگذاری مجموعه داده در **huggingface**

برای اینکه مجموعه داده‌مان به راحتی قابل دانلود و دسترسی باشد، آن را داخل هاب^۵ **huggingface** بارگذاری کنیم.

آپلود می‌کنیم.

```

hf_dataset.push_to_hub("keenGol/processed_semantic-search-channels")
Uploading the dataset shards: 100% [██████████] 1/1 [00:14<00:00, 14.58s/d]
Creating parquet from Arrow format: 100% [██████████] 2323 [00:02<00:00, 7.10s/d]
CommitInfo(commit_url="https://huggingface.co/datasets/keenGol/processed_semantic-search-channels/commit/1271f58e95ceec42f4661b8ad8c53822e9371ba9", commit_message="Upload dataset", commit_description="", old="1271f58e95ceec42f4661b8ad8c53822e9371ba9", pr_url=None, repo_url="https://huggingface.co/datasets/keenGol/processed_semantic-search-channels", endpoint="https://huggingface.co", repo_id="keenGol/processed_semantic-search-channels", pr_revision=None, pr_num=None)

```

شکل ۵-۶: فرستادن مجموعه داده به هاب
[۳۶]

⁵hub

۴-۵ ساخت برنامه‌ی کاربردی تحت وب

۱-۴-۵ مقدمه

برای اینکه این کد و برنامه از حالت یک فایل کولب در آمده و قابلیت استفاده توسط کاربران و به عبارتی یک رابط کاربری (UI) داشته باشد، باید این کد را روی یک سرور دیگر پیاده سازی کرده و با کدهای مرتبط با UI آن را ادغام کنیم. در ادامه مراحل ساخت برنامه کاربردی تحت وب^۶ بیان شده است.

۲-۴-۵ ساخت رابط کاربری

یک چارچوب^۷ پایتون منبع باز برای ساخت و استقرار برنامه‌های داده قدرتمند تنها با چند خط کد است. در این پروژه از Streamlit برای پیاده سازی ChatBot استفاده شده است. در ادامه مراحل راهاندازی یک UI ساده از چت‌بات توضیح داده می‌شود.

آموزش‌های لازم را برای راهاندازی یک چت‌بات ساده ارائه کرده است. [۳۷] مسئله‌ای Streamlit که در این مرحله وجود داشت، این بود که چون صفحه‌ی چت حاوی متن فارسی قرار بود باشد، جای کلمات بهم می‌ریخت پس برای حل این مشکل علاوه بر استفاده از تابع‌های خود Streamlit مقداری هم کد markdown استفاده شد. خروجی اولیه برنامه به صورت زیر خواهد بود:

⁶web application

⁷framework

۴-۵. ساخت برنامه‌ی کاربردی تحت وب



```
13  # CSS for RTL and Persian font
14  st.markdown(
15  """
16  <style>
17  @import url('https://cdn.jsdelivr.net/gh/rastikerdar/vazir-font/dist/font-face.css');
18  .rtl-text {
19  text-align: right;
20  direction: rtl;
21  font-family: 'Vazir', sans-serif;
22  font-size: 1.1em;
23  margin-bottom: 15px;
24  }
25  </style>
26  """
27  unsafe_allow_html=True,
28 )
29 with st.container():
30 # Display the message with RTL alignment
31 st.markdown(
32 """
33 <div class="rtl-text">
34 | <h1> خوش آمدید! </h1> <br>
35 |
36 </div>
37 """
38 unsafe_allow_html=True,
39 )
```

صفحه‌ی چت‌بات

استفاده از کد markdown و توابع خود Streamlit

شکل ۵-۸: ساخت صفحه‌ی چت

پیاده‌سازی قابلیت چت

دسترسی به برنامه Streamlit در مرورگر به عنوان یک session تعریف می‌شود. هر بار که با برنامه خود تعامل می‌کنید، اسکریپت شما را از بالا به پایین اجرا می‌کند و هیچ متغیری بین اجراهای به اشتراک گذاشته نمی‌شود.

راهی برای به اشتراک گذاشتن متغیرها بین اجرای مجدد، برای هر session کاربر است. علاوه بر توانایی ذخیره و تداوم وضعیت، Streamlit همچنین توانایی دستکاری وضعیت با استفاده از Callback‌ها را نیز ارائه می‌دهد [۳۸]. همچنین در صفحات داخل یک برنامه چندصفحه‌ای باقی می‌ماند.

برای اینکه امکان رد و بدل پیام بین کاربر و ربات در زمان حقیقی و بدون نیاز به دوباره لود کردن صفحه باشد، از Session State استفاده می‌کنیم. به این صورت که داخل آن متغیری به نام messages از جنس لیست ایجاد می‌کنیم و با هر پیام تولید شده، یک دیکشنری با کلیدهای "role" و "content" اضافه می‌کنیم. "role" نشانگر این است که چه کسی پیام را فرستاده و "content" محتوای پیام را نشان می‌دهد.

هنگامی که کد دوباره اجرا می‌شود، این پیام‌ها پاک نشده و قابل نمایش به کاربر هستند.

<pre> 165: ##### 166: if prompt == st.chat_input("چیزی"): 167: st.session_state.prompt = prompt 168: for message in st.session_state.messages: 169: with st.chat_message(message["role"]): 170: st.markdown(message["content"]) 171: st.markdown(prompt) 172: st.markdown(pr) </pre>	<p>ذخیره‌ی پیام جدید و نشان‌دادن آن</p>	<p>نشان‌دادن همه‌ی پیام‌ها</p>	<p>ساخت متغیر messages</p>
--	---	--------------------------------	----------------------------

شکل ۹-۵: ساخت یک برنامه‌ی چت

۳-۴-۵ وصل کردن مجموعه داده به برنامه‌ی کاربردی

در این بخش باید پیام‌های مشابه با پیام کاربر را از مجموعه داده پیدا کرده و سپس نمایش دهیم.

در این بخش ابتدا به بررسی چگونگی استفاده از مجموعه داده در کد می‌پردازیم، سپس بازگردن‌دها از مشابه‌ترین پیام‌ها را شرح می‌دهیم.

دانلود و ذخیره‌ی مجموعه داده

با استفاده از تکه کد زیر، مجموعه داده خود را که در بخش ۵-۵ به [huggingface](#) پوش کرده بودیم، مجدداً دانلود می‌کنیم. برای این منظور کد پایتونی در داخل پوشه‌ای به نام `data`، می‌سازیم تا داده‌ها نیز همان‌جا ذخیره شود.

```

bot > data > load.py > ...
1  from datasets import load_dataset
2  import os
3  # Download the dataset
4  # Get the current directory of this script
5  current_directory = os.path.dirname(os.path.abspath(__file__))
6  df_processed = load_dataset("keenGol/chatbot", split="train")
7  df_processed.save_to_disk(current_directory) # Save locally

```

شکل ۱۰-۵: دانلود مجموعه داده از هاب

خواندن مجموعه داده

حال باید در فایل اصلی برنامه، کدی بنویسیم که اولین بار که برنامه شروع به اجرا کرد، این مجموعه داده را بخواند و در حافظه‌ی cache خود نگهداری کند. در نتیجه لازم نیست هر بار با اجرای کد برنامه، از اول مجموعه داده را از دیسک بخواند. این امر موجب افزایش سرعت و بهینه بودن بیشتر کد می‌شود.] ۳۹ همان‌طور که در شکل ۱۱-۵ نشان داده شده است، مجموعه داده از دیسک خوانده شده سپس embedding‌ها (تعییه‌ها) و پیام‌های فارسی مجموعه داده به صورت جداگانه cache می‌شوند.

```

41  # Use the cached dataset
42  # Load dataset
43  @st.cache_resource
44  def load_dataset():
45      current_directory = os.path.dirname(os.path.abspath(__file__))
46
47      # Path to the Streamlit script in the current directory
48      data_path = os.path.join(current_directory, "data")
49      # Load the dataset from the saved location
50      dataset = load_from_disk(data_path)
51      dataset_embeddings = np.array(dataset["embedding"])
52      dataset_messages = np.array(dataset["Persian Messages"])
53      return dataset_embeddings, dataset_messages

```

شکل ۱۱-۵: ذخیره‌ی داده‌ها در دسترس بیشتر و افزایش سرعت با استفاده از cache

۴-۴-۵ پیدا کردن پیام مشابه با پیام کاربر

پس از اینکه کاربر در جعبه‌ی چت، پیام وارد کرد، با استفاده از اتصال خود به OpenAI، تعییه‌ی آن را به دست می‌آوریم سپس با استفاده از فاصله‌ی کسینوسی، پیام‌های نزدیک به پیام وارد شده را پیدا کرده و نمایش می‌دهیم.

```

180
181      # Get embedding of the prompt
182      query_embedding = client.embeddings.create(input=prompt, model=st.session_state.embedding_model).data[0].embedding
183      query_embedding = np.array(query_embedding).reshape(1, -1)
184      distances = cdist(query_embedding, dataset_embeddings, metric='cosine').flatten()
      sorted_indices = np.argsort(distances)

```

شکل ۱۲-۵: به دست آوردن تعییه‌ی پیام و پیدا کردن مشابه‌ها

۵-۴-۵ نمایش پیام‌ها

پس از مرتب کردن پیام‌ها بر اساس شباهت، دهتا از مشابه‌ترین‌ها را نمایش می‌دهیم و دکمه‌ی like و dislike استفاده می‌کنیم. کاربر پیام‌هایی که مورد علاقه‌اش بود را لایک می‌کند.

```

195     # Display the top nreturned sorted Persian messages
196     # for rank, (index, distance) in enumerate(sorted_indices[:st.session_state.nreturned]):
197     for rank, index in enumerate(sorted_indices[:st.session_state.nreturned]):
198         index = int(index)
199         distance = distances[index]
200         # st.session_state.rank = rank
201         print(f"rank: {rank}      index: {index}")
202         persian_message = dataset_messages[index] # Access the dataset row using the index
203         st.session_state.nearests.append(persian_message)
204         with st.container():
205             # Display the message with RTL alignment
206             st.markdown(
207                 f"""
208                 <div class="rtl-text">
209                     <strong>رتبه {rank + 1}:</strong> {persian_message} <br>
210                     # <em>نام: </em> {distance:.4f}
211                 </div>
212                 """,
213                 unsafe_allow_html=True,
214             )
215

```

شکل ۵-۵: نمایش پیام‌ها

```

216     # Create columns for Like/Dislike buttons
217     col1, col2 = st.columns([1, 1])
218     with col1:
219         key_name = f"like_{rank}"
220         st.button("👍", key=key_name, on_click=click_button, args=[key_name, persian_message, rank])
221
222     with col2:
223         key_name = f"dislike_{rank}"
224         st.button("👎", key=key_name, on_click=click_disButton, args=[key_name, persian_message, rank])
225

```

شکل ۵-۶: گذاشتن دکمه‌ی like و dislike

فصل ۵. روش پیاده‌سازی



شکل ۱۵-۵: نمای کاربری برنامه

پیاده سازی توابع دکمه‌ی like و dislike

وقتی دکمه‌ی لایک زده شد، تابع اجرایی باید آیدی پیام لایک شده را در یک مجموعه ذخیره کند یا چنانچه قبل اشتباهی dislike شده باشد، از مجموعه‌ی dislike‌ها حذف کرده و به مجموعه‌ی لایک اضافه کند. هر دکمه یک شناسه(id) دارد، شناسه‌ی این کلید را ذخیره می‌کنیم تا هنگام اجرای مجدد کد (با توجه به توضیحات بخش ۴-۵) دکمه‌های کلیدهای انتخاب شده disable باشد.

```
146  def click_button(key_name, persian_message, rank):  
147      st.session_state.key_name = True  
148      st.session_state.clicked += 1  
149      if f"dislike_{rank}" in st.session_state.disliked:  
150          st.session_state.disliked.remove(f"dislike_{rank}") # Remove from disliked  
151          st.session_state.cdisliked -= 1  
152          st.session_state.liked.add(key_name)  
153          st.session_state.selected_messages.append(persian_message)  
154          st.session_state.mrr += 1/(rank+1)  
155
```

شکل ۱۶-۵: تابع click-button برای پیام‌های like شده

```

155
156  def click_disButton(key_name, persian_message, rank):
157      st.session_state.key_name = True
158      st.session_state.cdisliked += 1
159      if f"like_{rank}" in st.session_state.liked:
160          # Remove from liked
161          st.session_state.liked.remove(f"like_{rank}")
162          st.session_state.clicked -= 1
163          st.session_state.selected_messages.remove(persian_message)
164          # add to disliked
165          st.session_state.disliked.add(key_name)
166

```

شکل ۱۷-۵: تابع click-disbutton برای پیام‌های dislike شده

۶-۴-۵ ساخت مجموعه داده‌ی بازخورد

داخل یک فایل json به نام feedback-dataset، ورودی کاربر(داخل فیلد prompt)، پیام‌های لایک شده(در فیلد liked)، پیام‌های دیس‌لایک شده(داخل فیلد disliked)، precision و MRR این درخواست ذخیره می‌شوند. بعدها از این مجموعه داده میتوان برای آموزش مدل با تکنیک RLHF (برای اطلاعات بیشتر به بخش ۳-۸-۲ مراجعه کنید) استفاده کرد.

تابع ذخیره‌ی بازخورد

تابع save_to_dataset، با توجه به آیدی پیام‌های ذخیره شده در مجموعه‌ی liked و disliked، پیام‌های متناظر را از لیست nearests بازیابی کرده و بعد یک دیکشنری با کلیدهایی که پیش‌تر اشاره شد، می‌سازد. سپس اطلاعات را به فایلی به نام feedback_dataset.json اضافه می‌کند و اگر این فایل وجود نداشته باشد، یکی می‌سازد.

```

67  def save_to_dataset(query, selected_messages, sorted_indices, filename="feedback_dataset.json"):
68      # Get the current directory and file path
69      current_directory = os.path.dirname(os.path.abspath(__file__))
70      print(current_directory)
71      file_path = os.path.join(current_directory, 'data', filename)
72      disliked_list = []
73      liked_list = []
74      if len(st.session_state.liked) != 0:
75          liked_list = [st.session_state.nearests[i] for i in range(st.session_state.nreturned) if f"like_{i}" in st.session_state.liked]
76      if len(st.session_state.disliked) == 0:
77          disliked_list = [st.session_state.nearests[i] for i in range(st.session_state.nreturned) if f"like_{i}" not in st.session_state.liked]
78      else:
79          disliked_list = [st.session_state.nearests[int(key.split('_')[1])]] for key in st.session_state.disliked]
80
81      # Create a dictionary of data to save
82      data = {
83          "prompt": query,
84          "liked": liked_list, # Keep as list; JSON handles serialization
85          "disliked": disliked_list,
86          "MRR": st.session_state.mrr,
87          "precision": precision()
88      }
89
90      # Check if the file exists and save appropriately
91      if not os.path.exists(file_path):
92          # If the file doesn't exist, create a new JSON file and write the data
93          with open(file_path, "w", encoding="utf-8") as file:
94              json.dump(data, file, ensure_ascii=False, indent=4) # Create a list with the first entry
95      else:
96          # If the file exists, append the new data
97          with open(file_path, "r+", encoding="utf-8") as file:
98              existing_data = json.load(file) # Load the existing JSON data
99              existing_data.append(data) # Append the new entry
100             file.seek(0) # Reset file pointer to the beginning
101             json.dump(existing_data, file, ensure_ascii=False, indent=4) # Write updated data
102

```

شکل ۱۸-۵: تابع ذخیره‌ی بازخورد

```

{
  "prompt": "از مینی امپل شروع می‌کنم جنگیدن برای رویاهام رو",
  "liked": [
    "لکر می‌کردم جنگ من و آکادمی و قلی خیلی بزرگتر شد شروع می‌کنم، ولی گویا از مینی امپرور شروع نمی‌کنم\n",
    "\nاین اول روزهایین من اینجا نیست می‌تویسم و برای دفعه آماده می‌نمم؛ این چه وضعیت‌نگاهداری داد و تا همیشه من رو به یه صلح عیقین با خودم رسونه. حال هرچند با پیروز جنگ شه قریب نمی‌باشد\n",
    "\nاین از فردا صبح شروع می‌شود و من نشستم اینجا نیست می‌تویسم و برای دفعه آماده می‌نمم؛ این چه وضعیت‌نگاهداری داد و تا همیشه من رو به یه صلح عیقین با خودم رسونه.\n",
    "\nامروز رو می‌تونم به عنوان روز انتقال در زندگی نامگذاری کنم. یه روزهایی خیلی گزند و خاک بیکنم. چه کارهایی که نمی‌کنم\n",
    "\nامثل اون روزی که بهم گفتمن می‌تونم پرتویم رو شروع کنم، حسما دارم می‌بینم بهم و از مینیان دوستدارم جیع بزم\n",
    "\nبه جول و قیوی‌الهی از امروز باشگاه رو شروع می‌کنم\n",
  ],
  "MRR": 2.661111111111111,
  "precision": 0.8
}

```

شکل ۱۹-۵: یک نمونه از بازخورد ثبت شده

پشتیانگیری

لازم به ذکر است که داخل فایل crontab، مشخص می‌کنیم که هر صبح ساعت 7:30 به وقت سرور،

یک پشتیانگیری از فایل feedback_dataset.json کرده و به رپایزیتوری مربوطه در huggingface بفرستد.

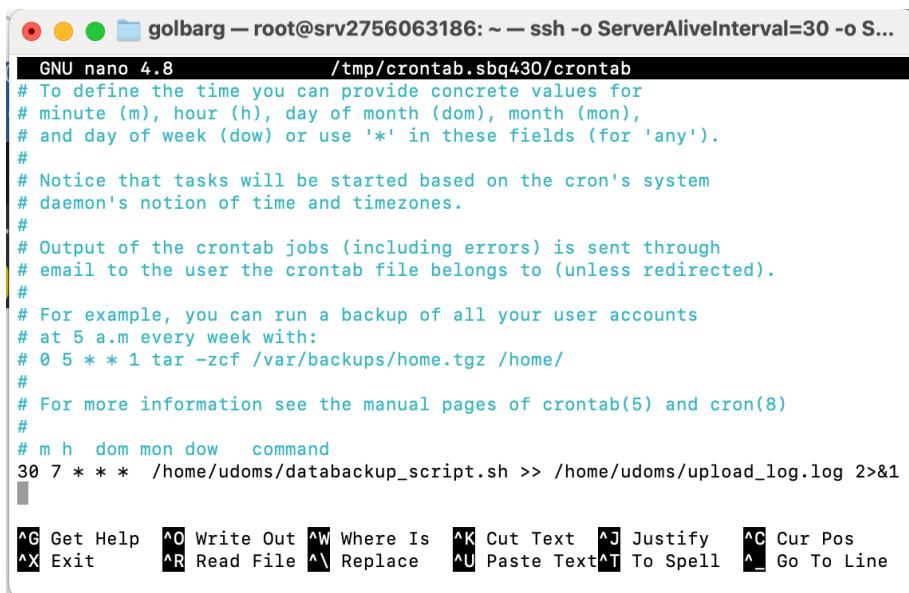
ابتدا یک فایل بس(bash) باید بنویسیم(۲۱-۵) سپس در فایل crontab مشخص کنیم آن را اجرا کند.

با استفاده از دستور -e، می‌توان فایل crontab را ویرایش کرد.



```
GNU nano 4.6
databasebackup_script.sh
#!/bin/bash
# Set the working directory if needed
#cd /home/udoms
# Run the huggingface-cli upload command
/test/final/911v/HuggingFace-cli upload keen001/processed_semantic-search-channels /home/udoms/Empathetic_Chatbot/bot/data/feedback_dataset.json --repo-type dataset --token [REDACTED]
```

شکل ۲۰-۵: فایل بش



```
GNU nano 4.8
/tmp/crontab.sjq430/crontab
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
30 7 * * * /home/udoms/databasebackup_script.sh >> /home/udoms/upload_log.log 2>&1
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
 ^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line

شکل ۲۱-۵: فایل crontab

۷-۴-۵ تولید پاسخ

در این مرحله پیام‌های لایک شده توسط کاربر را به مدل مولد، GPT-4o-mini، ارسال کرده و پس از گرفتن پاسخ از مدل، آن را به کاربر نشان می‌دهیم. در شکل ۵-۲۲، کد مربوط به ارسال ورودی کاربر همراه با پیام‌های منتخب، به مدل مولد نشان داده شده است. پس از اینکه کاربر دکمه‌ی انتخاب کردم را زد، متغیر "done" در Session State می‌گیرد. سپس پیام ورودی توسط کاربر به همراه پیام‌های منتخب، به مدل مولد ارسال می‌شود و در دستور ورودی، به مدل می‌گوییم که از آن پیام‌ها برای تولید پاسخ خود استفاده کند.

فصل ۵. روش پیاده‌سازی

۴-۵. ساخت برنامه‌ی کاربردی تحت وب

```
226 elif st.session_state.prompt!="": ##### selecting is done #####
227     # generating
228     ## Combine selected messages with user prompt
229     with st.spinner("جاری کار..."):
230         assistant_input = {
231             "selected_messages": st.session_state.selected_messages,
232             "prompt": st.session_state.prompt
233         }
234
235         # Pass selected messages and prompt to the assistant
236         with st.chat_message("assistant"):
237             stream = client.chat.completions.create(
238                 model=st.session_state["openai_model"],
239                 messages=[
240                     {"role": "system", "content": "Use the following selected messages to inform your response: {assistant_input['selected_messages']}"},
241                     {"role": "user", "content": assistant_input["prompt"]}
242                 ],
243                 stream=True,
244             )
245             response = st.write_stream(stream)
246             st.session_state.messages.append({"role": "assistant", "content": response})
247             # Save query, selected messages, and sorted messages
248             save_to_dataset(
249                 st.session_state.prompt,
250                 st.session_state.selected_messages,
251                 # st.session_state.sorted_indices
252                 st.session_state.nearests
253             )
```

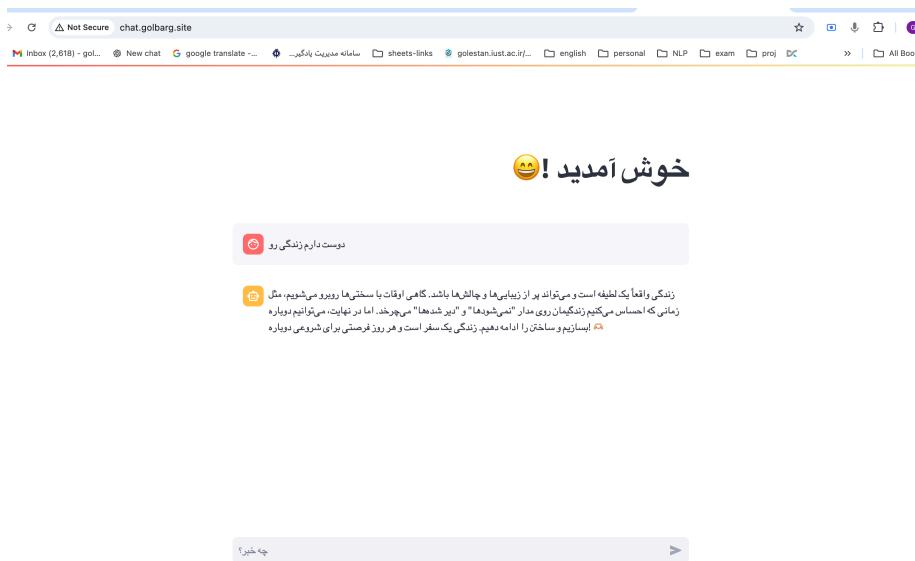
شکل ۵-۲۲: ارسال پیام‌های منتخب و دریافت پاسخ

```
{
  "prompt": "دوست دارم زندگی رو",
  "liked": [
    "\n    زندگی یک لطیفه است\n    ",
    "\n    می‌رم زندگی‌مو کنم\n    ",
    "\n    زندگی\n    ",
    "\n    مگه زندگی همین نیست؟ دوباره ساختن و ساختن\n    ",
    "\n    زندگی دوروزه و ایناون\n    ",
    "\n    زندگی‌م روی مدار \"نمیشه‌ها\"، \"دیر شده‌ها\" و \"نمیتونم‌ها\" میچرخه\n    "
  ],
  "disliked": [
    "\n    این بود زندگی\n    ",
    "\n    این بود، زندگی\n    ",
    "\n    زندگی در پیشرو | رونم گاری\n    ",
    "\n    من زندگی‌ای رو بیه می‌دم که خودم هیچوقت نداشتم عزیزم\n    "
  ],
  "MRR": 1.2861111111111112,
  "precision": 0.6
}
```

شکل ۵-۲۳: یک نمونه بازخورد ثبت شده

فصل ۵. روش پیاده‌سازی

۴-۵. ساخت برنامه‌ی کاربردی تحت وب



شکل ۵-۴: یک نمونه پاسخ تولید شده

۸-۴-۵ ارزیابی

برای ارزیابی عملکرد این برنامه، از معیار (precision) و دقت (Mean Reciprocal Rank (MRR)، استفاده

شده است. معیار (MRR)، ارزیابی می‌کند که اولین نتیجه مرتبط در پیام‌های

مشابه، چقدر بالا (با چه رتبه‌ای) ظاهر می‌شود و همچنین معیار دقت (precision)، تعداد پیام‌های

منتخب نسبت به کل پیام‌های نمایش داده شده را محاسبه می‌کند. البته لازم به ذکر است که چون

اینجا صرفا اولین پیام منتخب با کمترین رتبه نه بلکه تمام پیام منتخب مدل نظر هستند، برای هر

درخواست (prompt)، معکوس رتبه‌ی همه‌ی پیام‌های منتخب را جمع می‌زنیم. همان طور که در

شکل ۵-۱۶ نشان داده شده، با هر لایک، معکوس رتبه‌ها را روی هم انباشته می‌شود. همچنین تعداد

منتخب‌ها نیز در متغیر `clicked` داشته می‌شود. پس از اینکه انتخاب کاربر تمام شد (دکمه‌ی

انتخاب کردم زده شد)، دقت برنامه (فرمول ۴-۵) محاسبه می‌شود.

$$MRR = \sum_{i=1}^{\text{تعداد پیام‌های پسندیده شده}} \frac{1}{1 + \text{رتبه‌ی پیام منتخب آم}} \quad (۴-۵)$$

$$\text{Precision} = \frac{\text{تعداد پیام‌های منتخب}}{\text{تعداد کل پیام‌های نمایش داده شده}} \quad (۴-۶)$$

۹-۴-۵ بررسی میزان رضایت‌مندی

در تکه کد زیر میانگین MRR‌ها و همچنین میانگین precision‌ها به طور جداگانه حساب شده است. نتیجه‌ی هر دو تقریباً برابر با ۳۰ درصد شد. به این معنا که پیام مورد علاقه‌ی کاربران به طور متوسط در ۳۰ درصد مواقع، در میان پیام‌های نزدیک‌تر (رتبه‌ی کمتر) ظاهر می‌شود. همچنین به طور میانگین ۳۰ درصد پیام‌های نشان داده شده مرتبط است.

از میان بازخوردهای جمع‌آوری شده، ۵۶ پیام از ۶۱ پیام، دارای حداقل یک پیام منتخب هستند و این نشان می‌دهد که این برنامه با دقت ۹۱.۸ درصد، کاربردی بوده و موجب تغییر روحیه و آشنازی با نگرش‌های جدید شده است.

۱۰-۴-۵ خرید سرور

انواع مختلفی از سرورها، نظیر سرور مجازی، سرور اختصاصی، و سرور ابری، با سیستم‌عامل‌های لینوکس یا ویندوز قابل خریداری هستند. هر یک از این سرورها با توجه به امکانات و هزینه‌هایشان، موارد استفاده‌ی متفاوتی دارند. برای این پروژه، یک سرور ابری از شرکت پارس‌پک خریداری شده است. سرور ابری به دلیل هزینه‌ی مناسب، پایداری بالا، و امکان مقیاس‌پذیری انتخاب شده است.

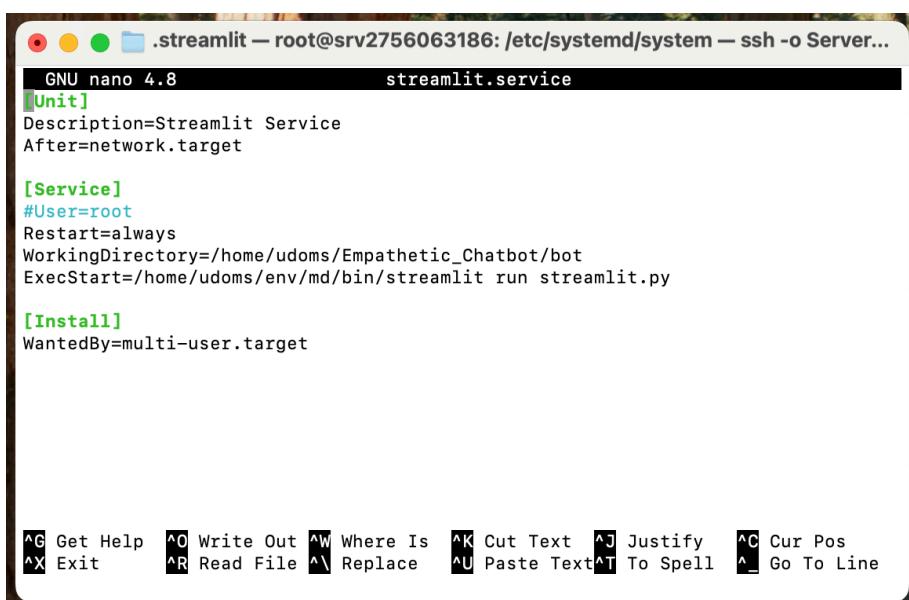
برای اطلاعات بیشتر می‌توانید به لینک قرار داده شده مراجعه نمایید.

۱۱-۴ راهاندازی برنامه بر روی سرور

تنظیم فایل Service در سرور

ایجاد یک فایل سرویس برای یک برنامه Streamlit در سرور تضمین می‌کند که برنامه به طور مداوم،

حتی پس از بسته شدن ترمینال، اجرا شود.^{۴۰}



```
GNU nano 4.8 streamlit.service
[Unit]
Description=Streamlit Service
After=network.target

[Service]
#User=root
Restart=always
WorkingDirectory=/home/udoms/Empathetic_Chatbot/bot
ExecStart=/home/udoms/env/md/bin/streamlit run streamlit.py

[Install]
WantedBy=multi-user.target
```

شکل ۵-۲۵: فایل سرویس

پیکربندی NGINX به عنوان یک پروکسی معکوس

برای پذیرش و مسیریابی درخواست، وب سرور NGINX به عنوان یک پروکسی معکوس^{۴۱} راه اندازی

می‌شود تا درخواست‌های کاربر را پذیرد و آن را به فایل سرویس هدایت کند.^{۴۰}

⁴⁰Reverse Proxy

```

GNU nano 4.8          Empathetic_Chatbot.conf          Modified
# configuration of the server
server {
    listen 80;
    server_name chat.golbarg.site;
    charset      utf-8;
    # max upload size
    client_max_body_size 75M;
    location / {
        proxy_pass http://127.0.0.1:8501; # Forward to Streamlit
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }
}

```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
 ^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line

شکل ۵-۲۶: فایل پیکربندی NGINX

خرید دامنه

همان گونه که در شکل ۵-۲۶ نمایش داده شده، نام دامنه‌ی سرور^۹ یعنی chat.golbarg.site به جای ip، گذاشته شده است. پس از خرید دامنه، از سایت پارس پک، باید دامنه‌ی سرور را در یک dns-server، یعنی سروری که نام دامنه و ip مربوط به سرورها را دارد و با مراجعه به آن و دادن نام دامنه، شما را به سرور مربوطه وصل کند. در واقع عملکرد آن مانند دفترچه‌ی شماره تلفن می‌باشد. یک شرکت آمریکایی است که خدمات DNS^۹ را نیز ارائه می‌دهد. پس از خرید دامنه، دامنه‌ی خود را در سایت آن باید ثبت کنیم، سپس با انجام دستورالعمل‌های داده شده، دامنه‌ی خود را به آن، متعلق Nameserver های آن، متصل کنیم.

⁹Domain Name Service

۵-۵ آموزش مدل پاداش

ما بازخورد انسانی را با یک مدل پاداش شبیه سازی خواهیم کرد. مدل پاداش هر متنی را می‌گیرد و به آن از این نظر که آن پیام چقدر برای انسان مفید خواهد بود، نمره‌ای می‌دهد. برای آموزش این مدل، باید مجموعه‌داده‌ای از جفت‌های پیام ورودی، پیام منتخب و غیر منتخب، را از مجموعه‌داده‌ی بازخورد خود^{۱۹-۵}، جمع آوری کنیم.

۵-۵-۱ تشکیل مجموعه‌داده‌ی جفت شده

در این بخش قصد داریم با استفاده از مجموعه‌داده‌ی بازخورد، مدلی آموزش دهیم که بتواند مانند انسان‌ها تصمیم بگیرد که کدام پاسخ‌ها مرتبط‌تر هستند. برای این کار نیاز است پیام‌های منتخب و غیر منتخب را به صورت زوجی گروه و همراه با ورودی کاربر به عنوان یک عضو مجموعه داده درآوریم. مدلی که جهت آموزش در نظر گرفته شده، ParsBert هست که در بخش ۶-۲^{۱۰} توضیحات لازم داده شده است. در شکل ۵-۲۷ دیتاست جفت شده^{۱۰} را می‌سازیم. توجه شود متغیر data در کد، همان مجموعه‌داده‌ی feedback_dataset.json است که از مجموعه‌داده‌مان در hugging face دانلود شده است. متغیر response_1 همان پاسخ منتخب، متغیر response_2 پاسخ غیر منتخب و label نشان می‌دهد کدام پاسخ بر دیگری برتری داده شده است. اگر label یک باشد یعنی response_1 به response_2 ترجیح داده شده است.

¹⁰ Pairwised dataset

```

❶ import torch
from torch.utils.data import Dataset
from transformers import AutoTokenizer

# Initialize tokenizer
model_name_or_path = "HooshvarLab/bert-fa-zwnj-base"
tokenizer = AutoTokenizer.from_pretrained(model_name_or_path)

# Generate paired data
paired_data = []
max_length = 256 # Adjust as needed

for record in data:
    prompt = record["prompt"]
    liked = record["liked"]
    disliked = record["disliked"]

    for liked_response in liked:
        for disliked_response in disliked:
            paired_data.append({
                "prompt": prompt,
                "response_1": liked_response,
                "response_2": disliked_response,
                "label": 1 # Label: liked > disliked
            })

```

/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret HF_TOKEN does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (<https://huggingface.co/settings/tokens>), set it as secret in your Google Colab and restart your session.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn()

tokenizer_config.json: 100% [██████████] 298/298 [00:00:00, 5.24MB/s]
config.json: 100% [██████████] 668/668 [00:00:00, 34.98kB/s]
vocab.txt: 100% [██████████] 298/298 [00:00:00, 17.5MB/s]
tokenizer.json: 100% [██████████] 1.11MB/1.11M [00:00:00, 462kB/s]
special_tokens_map.json: 100% [██████████] 134/134 [00:00:00, 4.85kB/s]

شکل ۲۷-۵: ساخت جفت‌های منتخب و نامنتخب

۲-۵-۵ بارگیری مدل

مدل استفاده شده برای این کار ParsBert می‌باشد (شکل ۲۷-۵). چون این مدل یک مدل زبانی بزرگ برای فهم زبان فارسی است، دقت آن برای فهم جملات فارسی و انجام تسكیهایی نظری جزیه و تحلیل احساسات^{۱۱}، طبقه‌بندی متن^{۱۲}، و شناسایی موجودیت نامگذاری شده^{۱۳}، در زبان فارسی، از بسیاری از مدل‌های چندزبانه بهتر است [۱۲].

این مدل را برای آموزش جهت تشخیص label نمونه‌های جفت شده، به عنوان like یا dislike استفاده می‌کنیم. در واقع می‌توان این مدل را برای انجام تسكیهایی و طبقه‌بندی متن منتخب و غیرمنتخب، آموزش داد. تنظیمات لازم برای پیکربندی مدل پاداش، در کلاس شکل ۲۸-۵^{۱۴} انجام شده است.

پس از لود مدل اصلی یعنی همان ParsBert، یک لایه‌ی خطی اضافه می‌کنیم تا از همه ویژگی‌های خروجی از لایه‌ی پنهان قبلی، یک نتیجه‌ی کلی در مورد منتخب بودن یا نبودن متن مربوطه بگیرد.

¹¹Sentiment Analysis (SA)

¹²Text Classification

¹³Name Entity Recognition (NER)

تابع forward نحوه پردازش یک دنباله ورودی از طریق مدل، برای تولید امتیاز/پاداش را تعریف می‌کند. این فرآیند شامل مراحل استخراج ویژگی‌ها با استفاده از مدل پایه (self.base_model)، تجمعی بازنمایی‌ها برای ایجاد یک خروجی فشرده (pooler_output)، و محاسبه پاداش با استفاده از یک لایه خطی آخر (self.reward_head) می‌باشد.

Reward Model

```
from transformers import AutoModel
import torch.nn as nn

class RewardModel(nn.Module):
    def __init__(self, model_name_or_path):
        super(RewardModel, self).__init__()
        self.base_model = AutoModel.from_pretrained(model_name_or_path)
        self.reward_head = nn.Linear(self.base_model.config.hidden_size, 1) # Regression head

    def forward(self, input_ids, attention_mask):
        outputs = self.base_model(input_ids=input_ids, attention_mask=attention_mask)
        pooled_output = outputs.pooler_output
        reward = self.reward_head(pooled_output)
        return reward
```

شکل ۲۸-۵: مدل پاداش

بارگیری tokenizer

BERT ۱-۱۰-۲، توکن‌های ویژه‌ای مانند [CLS] و [SEP] را که برای مدل‌های مبتنی بر ضروری هستند، به متن اضافه می‌کند. در نهایت، توکن‌ها به شناسه‌های عددی متناظرشان در واژگان مدل نگاشت می‌شوند تا بتوانند به عنوان ورودی مدل استفاده شوند. این فرآیند به بهبود کارایی و دقیقی مدل‌های یادگیری عمیق در وظایف مختلف زبانی کمک می‌کند. همان‌طور که در شکل ۲۷-۵ نمایش داده شده، مربوط به این مدل بارگیری می‌شود.

۳-۵-۵ ساخت مجموعه‌داده‌ی مناسب برای مدل

حال باید مجموعه‌داده‌ای مناسب برای پردازش و آموزش توسط مدل، بسازیم. کد ۲۹-۵، مجموعه داده‌ی ساخته‌شده در شکل ۲۷-۵ را می‌گیرد (pairs) و برای هر عضو آن، ترکیب ورودی کاربر در کنار

پیام منتخب (inputs_1)، و همچنین ورودی کاربر در کنار پیام غیر منتخب (inputs_2) را به شناسه‌های مربوطه در واژگان مدل نگاشت می‌کند. سپس این شناسه‌ها را به همراه ماسک‌های توجه برمی‌گرداند. label خروجی نیز که طبق شکل ۲۷-۵ همیشه یک است، نشان می‌دهد که inputs_1 به inputs_2 بترتیب داده شده است. max_length برای نشانه‌گذار این مدل برابر ۱۲ است. درصد از مجموعه‌داده را برای آموزش و باقی را برای ارزیابی جدا می‌کنیم.

```
[ ] class PairwiseDataset(Dataset):
    def __init__(self, pairs, tokenizer, max_length):
        self.pairs = pairs
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.pairs)

    def __getitem__(self, idx):
        pair = self.pairs[idx]
        prompt = pair["prompt"]
        response_1 = pair["response_1"]
        response_2 = pair["response_2"]
        label = pair["label"]

        # Tokenize the inputs
        inputs_1 = self.tokenizer(
            prompt + " " + response_1,
            truncation=True,
            padding="max_length",
            max_length=self.max_length,
            return_tensors="pt"
        )
        inputs_2 = self.tokenizer(
            prompt + " " + response_2,
            truncation=True,
            padding="max_length",
            max_length=self.max_length,
            return_tensors="pt"
        )

        # Prepare output
        return {
            "input_ids_1": inputs_1["input_ids"].squeeze(0),
            "attention_mask_1": inputs_1["attention_mask"].squeeze(0),
            "input_ids_2": inputs_2["input_ids"].squeeze(0),
            "attention_mask_2": inputs_2["attention_mask"].squeeze(0),
            "label": torch.tensor(label, dtype=torch.float)
        }
```

شکل ۵-۵: ساخت مجموعه‌داده جهت آموزش مدل

۴-۵-۵ آموزش

فرایند آموزش طبق شکل ۳۱-۵ در سه epoch، انجام می‌شود. از تابع `nn.BCEWithLogitsLoss()` در سه epoch، انجام می‌شود. همچنین از تابع `optim.Adam` به عنوان تابع ضرر (توضیحات بیشتر در بخش ۱۰-۲) و بهینه‌ساز (توضیحات بیشتر در بخش ۱۰-۶) استفاده شده است. در ابتدای هر epoch، تابع `optimizer.zero_grad()` قبل به‌طور پیش‌فرض در PyTorch جمع‌آوری می‌شوند، بنابراین قبل از محاسبه گرادیان‌های جدید، انجام این فرایند، لازم است. سپس پیش‌بینی‌ها (`reward_1` و `reward_2`) برای دو مجموعه ورودی در دسته محاسبه می‌شود.

محاسبه‌ی ضرر

اختلاف بین دو پاداش (`reward_1` و `reward_2`) می‌باشد. تابع زیان (`nn.BCEWithLogitsLoss()`) می‌باشد. اگر `label = 1` باشد، مدل یاد می‌گیرد مقدار $reward_1 - reward_2$ را افزایش و اگر `label = 0` باشد، مدل یاد می‌گیرد مقدار $reward_2 - reward_1$ را افزایش دهد. بنابراین، کمینه کردن تابع زیان اطمینان می‌دهد که `label = 1` (یعنی $reward_1 - reward_2$) در صورتی که `label = 0` باشد تا حد ممکن بزرگ شود، یا اگر `label = 0` باشد، تا حد ممکن کوچک شود.

```

from torch.utils.data import DataLoader
import torch
import torch.nn as nn
import torch.optim as optim

model = RewardModel(model_name_or_path)
model = model.to("cuda") # Use GPU if available

criterion = nn.BCEWithLogitsLoss() # Binary Cross Entropy Loss
optimizer = optim.Adam(model.parameters(), lr=5e-5)

pytorch_model.bin: 100% [██████████] 470M/479M [00:01<00:00, 228MB/s]
Some weights of BertModel were not initialized from the model checkpoint at HuggingFace/Bert-fa-zwj-base and are newly initialized: ['bert.pooler.dense.bias', 'bert.pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

```

شکل ۵-۳۰: تنظیم تابع ضرر، بهینه‌ساز و مدل پاداش

```

from tqdm import tqdm

# Training loop
for epoch in range(num_of_epochs): # Adjust the number of epochs
    model.train()
    epoch_loss = 0

    # Training with Tqdm progress bar
    progress_bar = tqdm(train_loader, desc=f"Epoch {epoch + 1} [Training]")
    for batch in progress_bar:
        optimizer.zero_grad()

        # Forward pass
        reward_1 = model(batch["input_ids_1"].to("cuda"), batch["attention_mask_1"].to("cuda"))
        reward_2 = model(batch["input_ids_2"].to("cuda"), batch["attention_mask_2"].to("cuda"))

        # Compute pairwise loss
        logits = reward_1 - reward_2
        loss = criterion(logits, batch["label"].to("cuda").unsqueeze(1))
        epoch_loss += loss.item()

        # Backward pass and optimization
        loss.backward()
        optimizer.step()

        # Update progress bar with current loss
        progress_bar.set_postfix(loss=loss.item())

    # Validation accuracy
    val_accuracy = evaluate(model, val_loader)
    print(f"Epoch {epoch + 1} | Loss: {epoch_loss / len(train_loader):.4f} | Validation Accuracy: {val_accuracy:.4f}")

    # Epoch 1 [Training]: 100%[██████████] 29/29 [01:15<00:00, 2.61s/it, loss=0.0741]
    Epoch 1 | Loss: 0.3125 | Validation Accuracy: 0.5776
    Epoch 2 [Training]: 100%[██████████] 29/29 [01:16<00:00, 2.65s/it, loss=0.000628]
    Epoch 2 | Loss: 0.0041 | Validation Accuracy: 0.4569
    Epoch 3 [Training]: 100%[██████████] 29/29 [01:19<00:00, 2.73s/it, loss=9.18e-5]
    Epoch 3 | Loss: 0.0004 | Validation Accuracy: 0.4655

```

شکل ۳۱-۵: آموزش مدل

۵-۵-۵ ارزیابی

پس از هر epoch، مدل بر روی مجموعه داده‌ی جداشده برای ارزیابی، آزموده می‌شود (شکل ۳۱-۵).

تابع ارزیابی به صورت زیر (شکل ۳۲-۵) پیاده‌سازی شده است. در این تابع تعداد دفعاتی که مدل مقدار reward_1 را که امتیاز مربوط به پیام منتخب می‌باشد، بیشتر از reward_2 قرار داده، شمرده می‌شود و در نهایت تعداد کل تشخیص‌های درست نسبت به تعداد کل اعضای مجموعه‌ی مورد ارزیابی، به عنوان دقت برگرددانده می‌شود.

```

# Function to evaluate validation accuracy
def evaluate(model, dataloader):
    model.eval()
    correct = 0
    total = 0

    with torch.no_grad():
        for batch in dataloader:
            reward_1 = model(batch["input_ids_1"].to("cuda"), batch["attention_mask_1"].to("cuda"))
            reward_2 = model(batch["input_ids_2"].to("cuda"), batch["attention_mask_2"].to("cuda"))

            # Predict the preferred response
            predictions = (reward_1 > reward_2).squeeze() # 1 if reward_1 > reward_2
            correct += (predictions == batch["label"].to("cuda")).sum().item()
            total += len(batch["label"])

    return correct / total

```

شکل ۵-۳۲: تابع ارزیابی

۶-۵-۵ امتحان یک پیام ورودی

در این بخش یک پیام به عنوان ورودی به مدل می‌دهیم و بررسی می‌کنیم چه پیام‌هایی را به عنوان منتخب بر می‌گرداند (شکل ۵-۳۴). تابع `batch_rank_candidates` (شکل ۵-۳۳)، پس از نشانه‌گذاری پیام ورودی و پیام‌های مجموعه‌داده، امتیازی را به هر ترکیب (پیام ورودی + پیام مجموعه‌داده) جهت منتخب بودن یا نبودن نسبت می‌دهد. سپس این امتیازها به صورت نزولی مرتب می‌شوند. پیام‌هایی که امتیاز بالاتری گرفته‌اند نمایش داده می‌شوند. (شکل ۵-۳۵)

```

prompt = "می‌خوام مستقل بشم"
candidates = candidates["Persian Messages"]

# Rank candidates for the given prompt
ranked_candidates = batch_rank_candidates(model, tokenizer, prompt, candidates)

print(f"Prompt: {prompt}")
print("Ranked Candidates:")
for i, (message, score) in enumerate(ranked_candidates):
    print(f"{i + 1}. {message} (Score: {score:.4f})")

```

شکل ۵-۳۳: بررسی یک نمونه ورودی

فصل ۵. روش پیاده‌سازی

۵-۵. آموزش مدل پاداش

```
def batch_rank_candidates(model, tokenizer, prompt, candidates, batch_size=16, max_length=512):
    model.eval()
    scores = []

    for i in range(0, len(candidates), batch_size):
        batch_candidates = candidates[i:i + batch_size]

        # Tokenize the batch
        encoding = tokenizer(
            [prompt] * len(batch_candidates), # Repeat prompt for each candidate
            batch_candidates,
            return_tensors="pt",
            padding="max_length",
            truncation=True,
            max_length=max_length
        )
        input_ids = encoding["input_ids"].to("cuda")
        attention_mask = encoding["attention_mask"].to("cuda")

        with torch.no_grad():
            # Get reward scores
            rewards = model(input_ids, attention_mask).squeeze(-1) # Adjust if model outputs differently
            scores.extend(rewards.cpu().tolist())

    # Rank candidates
    ranked_candidates = sorted(zip(candidates, scores), key=lambda x: x[1], reverse=True)
    return ranked_candidates
```

شکل ۵-۴: رتبه‌بندی پیام‌ها بر اساس ورودی

شکل ۵-۳۵: خروجی مدل

فصل ۶

بررسی نتایج

۱-۶ جمع‌بندی

در این پژوهه، مراحل طراحی و ساخت یک برنامه‌ی شباهت‌سنجی با استفاده از ابزارها و framework‌های موجود در زمینه‌ی هوش مصنوعی و پردازش زبان شرح داده و اجرا شد. بدین وسیله می‌توان پیام‌های مشابه با پیام خود را از مجموعه‌داده‌ی مورد علاقه، بازیابی کرد (با استفاده از فاصله‌ی کسینوسی). با پیشنهاد پیام‌های مرتبط از مجموعه‌داده‌های مورد علاقه، که در اینجا کانال‌های تلگرامی پرطرفدار بود، حس هم‌دلی بیشتری القا شده و همچنین، چون لحن این کانال‌ها به صورت خودمانی است و مطالب آن اکثراً تجربه‌شده و ملموس است، موجب بهتر شدن روحیه و شادی و پیدا کردن دید جدیدی به یک موضوع می‌شود. بعلاوه قابلیت دادن بازخورد به پیام‌ها قرار داده شد. سپس با جمع‌آوری بازخوردها و آموزش یک مدل پاداش، فرایند RLHF بررسی و پیاده سازی گردید. در این رساله، مراحل جمع‌آوری داده، پیاده‌سازی نرم‌افزار و راهاندازی بر روی سرور و آموزش مدل پاداش، همگی توضیح داده شدند.

۱-۱-۶ بررسی چند نمونه خروجی

شکل ۶-۱: زندگی ساده!

شکل ۶-۲: خوابم میاد!

```

116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
{
  "prompt": "کد نش نش نش",
  "liked": [
    "ند نش میز نشاخنون میزم"\n,
    "ند نش میرم کافه میزینم قهقهه میخونم و نکاره میخونم"\n,
    "اگر از میان درد رش نشکنی درد رش خواهند شد"\n,
    "ند نش درد خواهند شد اگر کوچکی میسازم و تو زنگی میکنم"\n,
    "ند نش باید از داره نهاد"\n
  ],
  "disliked": [
    "ند دوستیم"\n,
    "ند اچیزی"\n,
    "ند نشونون نشند"\n,
    "ند ایزی"\n,
    "ند نشیدند"\n
  ],
  "MRR": 1.553968253968254,
  "precision": 0.5
}

```

شکل ۶-۳: هر چی شد شد!

فصل ۶. بررسی نتایج

شکل ۶-۴: شروع از همین حالا!

شکل ۶-۵: غر نزن و ادامه بده!

،**"prompt": "اون دند کار مینکن ختنما"**
"liked": [
 ،**"80" دن لمبادونم باید چیکار کنم ترین حالت ممکن"**
 ،**"81" میتوون بلهمن کارهایی که دارن میکن، چوڑا داره اتفاقی می افته، این په صاده انگاریه و قلی که تصور میکنیدون مغلالت، میزونم چه چیکار داریم**
]
"disliked": [
 ،**"82" چی اونجا صد میله که نمیتوان بلهمن کاره که کردی درست بوده، پا افشار، نمیتوان بلهمن اک برگردی به این دوباره اخراجی بیندی پنهان**
 ،**"83" چم اونقدر خوب بوده که بیام پیمانه کاری کردی که شایان میگردید اما اونقدر بد بود که بیام نهاده چیزی**
 ،**"84" چم میگردید که بیام درست ننمایم، با هم درس و تختی که بیام عقیونم اکر بگیختم و توکم کنم از کنکا**
 ،**"85" چم بیام بیار درست ننمایم بشه و بزه میگردید بیام بیار درست ننمایم بشه و بزه میگردید**
 ،**"86" چم بیام بیار درست ننمایم بشه و بزه میگردید بیام بیار درست ننمایم بشه و بزه میگردید**
 ،**"87" چم بیام بیار درست ننمایم بشه و بزه میگردید بیام بیار درست ننمایم بشه و بزه میگردید**
 ،**"88" چم بیام بیار درست ننمایم بشه و بزه میگردید بیام بیار درست ننمایم بشه و بزه میگردید**
]
"MRR": 0.7,
"precision": 0.2

شکل ۶-۶: درست شد!

فصل ۶. بررسی نتایج

توانایی انجام دادن شاره ای از این به بعد می خواه قلچه شعرکرم روزه خودم باشه. خوبی به این حسادت به نظرم توجیهی نداره"
باشد اعتماد نکن، به خود میگم اعتماد نکن، انتظار، لکلکنم اگه درست و با تمیز و دلت پیشینم پیار و شلاق کنم میتوونم از پیش بزیر باشم"
لئن ناکامها و خشم اونها میم، بعضها بخاطر مبلغ و آرایش درستی ای که در مصیر رسیدن به اعتماد نکنند به خودم"
مشتی، بلکه خودت روز به عنوان پیش انسان دانها درحال تغییر پیدا شوند و حظاها روز مثل پیش تو صرف نمیکویی و تو میل و آرامش شاره بزیر بزیر میکنند"
به این می رسیدن، تایید تیازه مدمیگر و یکم کمی بغل کمی کمی کمی که این امروز گرفتم، که به خودمون یاد اویز کنیم که از اون چیزی که فکر می کنیم لز ندیده"
۱،
"disliked": [
"۱۰. از خستگی و شاره ای نهایه دارم یکی رو بغل کنم و قایم شم. ترجیحاً با موقعاً بگیرید" \n"۱۱. بجزی از حسادت (غیر)، خانه اسوزر ندیده"
"۱۲. از نظر دیگری برای میم نیست، غیلی و قندما بجذب کم گرفتی نیست، غیلی و قندما دارم به کاری رو میگذری از اینها بیشام، وقتی شنایام، به خود اعتمادی نه ام"
"۱۳. خود میگیرم که میتوونم تمام کارهایم غلبه شدم و فقط در این مورد نشیمن فکر کنم و بخونم، خوبی خود دوستی میم در نهاده"
"۱۴. ری میکنم، این خوشبیت مثل نفس کشیدن میمونه، وقتی خواست بیهی نیست درست کار میکنه، وقتی بیهی توجه میکنم تبدیل به اورتینگ و لوب دانمی میش، جالبه" \n"
"MRR": 1.478968253968294,
"precision": 0.5

شکل ۶-۷: نه به حسادت!

۲-۶ کارهای آینده

می توان از بازخوردهای جمع آوری شده برای آموزش یک مدل توصیه گر یا پیشنهاددهنده استفاده کرد.

در واقع، RLHF یا یادگیری تقویتی از بازخورد انسانی، تکنیکی برای همسو کردن یک عامل هوشمند

با ترجیحات انسان است. این شامل آموزش یک مدل پاداش برای نشان دادن ترجیحات است که

سپس می تواند برای آموزش مدل های دیگر از طریق یادگیری تقویتی استفاده شود.

در این پروژه می توان برای هر نمونه‌ی موجود در مجموعه‌داده‌ی feedback_dataset.json، هر عضو

لیست لایک شده‌ها را با هر عضو لیست لایک نشده‌ها به صورت جفت‌جفت درآورد و همراه با پیام

ورودی (prompt) به مدل داد (برای مرور ساختار مجموعه‌داده‌ی بازخورد به بخش ۶-۴-۵ مراجعه

شود). با آموزش مدل بر روی مجموعه‌داده‌ی جفت‌های تشکیل شده، می توان مدلی آموزش داد که

با احساسات و مکالمات انسانی و صمیمانه آشناتر باشد و قدرت هم صحبتی بهتری داشته باشد.

كتاب نامه

- [1] Nisan Stiennon, Long Ouyang, Jeff Wu Daniel M. Ziegler Ryan Lowe Chelsea Voss Alec Radford Dario Amodei Paul Christiano. Learning to summarize from human feedback. *arXiv preprint arXiv:2009.01325*, 2024.
- [2] McCarthy, John. Artificial intelligence, logic and formalizing common sense. in Thomason, Richmond, ed. , *Philosophical Logic and Artificial Intelligence*. Kluwer Academic, 1989.
- [3] Nilson, Nils J. Introduction to machine learning. *AN EARLY DRAFT OF A PROPOSED TEXTBOOK. Robotics Laboratory. Department of Computer Science Stanford University. Stanford. USA. Recuperado de: <http://ai.stanford.edu/~nilsson/MLBOOK.pdf>*, 1998.
- [4] Kelleher, John D. *Deep learning*. MIT press, 2019.
- [5] Boykis, Vicki. What are embeddings?, jun 2023. Version 1.0.1. Published on GitHub. DOI: 10.5281/zenodo.8015029.
- [6] Minaee, Shervin et al. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.
- [7] Gao, Tianyu, Yao, Xingcheng, and Chen, Danqi. Simcse: Simple contrastive learning of sentence embeddings. in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6894–6910, 2021.
- [8] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. Accessed: 2025-01-06.
- [9] Conneau, Alexis, Kiela, Douwe, Schwenk, Holger, Barrault, Loïc, and Bordes, Antoine. Supervised learning of universal sentence representations from natural language inference data.

in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

- [10] Kiros, Ryan, Zhu, Yukun, Salakhutdinov, Ruslan, Zemel, Richard S, Torralba, Antonio, Urtasun, Raquel, and Fidler, Sanja. Skip-thought vectors. in *Advances in Neural Information Processing Systems (NIPS)*, pp. 3294–3302, 2015.
- [11] openai. A new small text embedding model. <https://openai.com/index/new-embedding-models-and-api-updates/>, 2023.
- [12] Mehrdad Farahani, Mohammad Gharachorloo, Marzieh Farahani Mohammad Manthouri. Parsbert: Transformer-based model for persian language understanding. *Neural Processing Letters*, 2021.
- [13] Harshvardhan, GM, Gourisaria, Mahendra Kumar, Pandey, Manjusha, and Rautaray, Siddharth Swarup. A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38:100285, 2020.
- [14] Sakib, Md Sakibul Islam. What is chatgpt. *ResearchGate*, 2023.
- [15] ul Abideen, Zain. Reinforcement learning from human feedback (rlhf): Empowering chatgpt with user guidance. <https://medium.com/@zaiinn440/reinforcement-learning-from-human-feedback-rlhf-empowering-chatgpt-with-user-gui> 2023.
- [16] Nadkarni, Prakash M, Ohno-Machado, Lucila, and Chapman, Wendy W. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551, 2011.
- [17] Lewis, Patrick, Perez, Ethan, Piktus, Aleksandra, Petroni, Fabio, Karpukhin, Vladimir, Goyal, Naman, Kütller, Heinrich, Lewis, Mike, Yih, Wen-tau, Rocktäschel, Tim, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
- [18] Uc-Cetina, Victor, Navarro-Guerrero, Nicolás, Martin-Gonzalez, Anabel, Weber, Cornelius, and Wermter, Stefan. Survey on reinforcement learning for language processing. *Artificial Intelligence Review*, 56(2):1543–1575, 2023.

- [19] Chaudhari, Shreyas, Aggarwal, Pranjal, Murahari, Vishvak, Rajpurohit, Tanmay, Kalyan, Ashwin, Narasimhan, Karthik, Deshpande, Ameet, and da Silva, Bruno Castro. Rlhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. *arXiv preprint arXiv:2404.08555*, 2024.
- [20] Reinforcement learning from human. <https://aws.amazon.com/what-is/reinforcement-learning-from-human-feedback/>, 2024.
- [21] Medium. Create a high-quality dataset for reinforcement learning from human feedback. <https://labelstud.io/blog/create-a-high-quality-rlhf-dataset/>, 2023.
- [22] Hou, Zhenyu, Niu, Yilin, Du, Zhengxiao, Zhang, Xiaohan, Liu, Xiao, Zeng, Aohan, Zheng, Qinkai, Huang, Minlie, Wang, Hongning, Tang, Jie, et al. Chatglm-rlhf: Practices of aligning large language models with human feedback. *arXiv preprint arXiv:2404.00934*, 2024.
- [23] Colla, Davide, Mensa, Enrico, and Radicioni, Daniele P. Novel metrics for computing semantic similarity with sense embeddings. *Knowledge-Based Systems*, 206:106346, 2020.
- [24] Grefenstette, Gregory. Tokenization. in *Syntactic wordclass tagging*, pp. 117–133. Springer, 1999.
- [25] Rabanser, Stephan, Shchur, Oleksandr, and Günnemann, Stephan. Introduction to tensor decompositions and their applications in machine learning. *arXiv preprint arXiv:1711.10781*, 2017.
- [26] Ouyang, Long, Wu, Jeffrey, Jiang, Xu, Almeida, Diogo, Wainwright, Carroll, Mishkin, Pamela, Zhang, Chong, Agarwal, Sandhini, Slama, Katarina, Ray, Alex, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.
- [27] Inkster, Becky, Sarda, Shubhankar, Subramanian, Vinod, et al. An empathy-driven, conversational artificial intelligence agent (wysa) for digital mental well-being: real-world data evaluation mixed-methods study. *JMIR mHealth and uHealth*, 6(11):e12106, 2018.
- [28] Pentina, Iryna, Hancock, Tyler, and Xie, Tianling. Exploring relationship development with social chatbots: A mixed-method study of replika. *Computers in Human Behavior*, 140:107600, 2023.

- [29] Hakim, Fauzia Zahira Munirul, Indrayani, Lia Maulia, and Amalia, Rosaria Mita. A dialogic analysis of compliment strategies employed by replika chatbot. in *Third International conference of arts, language and culture (ICALC 2018)*, pp. 266–271. Atlantis Press, 2019.
- [30] Skjuve, Marita, Følstad, Asbjørn, Fostervold, Knut Inge, and Brandtzaeg, Petter Bae. My chatbot companion-a study of human-chatbot relationships. *International Journal of Human-Computer Studies*, 149:102601, 2021.
- [31] Stephens, Taylor N, Joerin, Angela, Rauws, Michiel, and Werk, Lloyd N. Feasibility of pediatric obesity and prediabetes treatment support through tess, the ai behavioral coaching chatbot. *Translational behavioral medicine*, 9(3):440–447, 2019.
- [32] tess-support chatbot. <https://www.x2ai.com/individuals>, 2024.
- [33] How to count tokens with tiktoken.
- [34] pineconequickstart. <https://docs.pinecone.io/guides/get-started/quickstart>, 2025.
- [35] distance-metrics. <https://docs.pinecone.io/guides/indexes/understanding-indexes#distance-metrics>, 2024.
- [36] push dataset to huggingface. https://huggingface.co/docs/datasets/en/upload_dataset#upload-with-python, 2024.
- [37] How to build an llm-powered chatbot with streamlit. <https://blog.streamlit.io/how-to-build-an-llm-powered-chatbot-with-streamlit/>, 2024.
- [38] What is state? <https://docs.streamlit.io/develop/concepts/architecture/session-state>, 2024.
- [39] Caching overview. <https://docs.streamlit.io/develop/concepts/architecture/caching>, 2024.
- [40] Borghare, Shubham. Caching overview. <https://medium.com/@borghareshubham510/deploying-streamlit-application-on-aws-ec2-instances-with-nginx-5a2a2a2a2a2a>, 2023.

Abstract:

With the remarkable advancements in robotics and artificial intelligence, these human creations have evolved into omnipresent and versatile assistants for people. The development and training of large language models, along with their relatively satisfactory performance in various tasks such as sentiment analysis and text generation, have accelerated AI's progress toward understanding and interacting in human language. Currently, artificial intelligence is evaluated as a consultant/assistant with vast knowledge and insights. However, the question arises: can it also exhibit emotional empathy and provide supportive feedback to the same extent?

This project implements an application software designed to suggest messages similar to the user's input message from several popular Telegram channels. The dataset comprises content from popular Telegram channels where authors share daily life experiences, perspectives, and informal, impromptu thoughts, creating a personal and safe space. Consequently, the retrieved messages resonate with a more tangible and empathetic tone. Additionally, the casual and informal tone of these messages promotes a sense of well-being and positivity. In this project, similar messages are identified by calculating the cosine distance between the user's input message and messages in the dataset. The ten messages with the smallest distance are displayed to the user, who can then select their preferred messages. The software's performance is evaluated using metrics such as the number of selected messages relative to the total displayed and the sum of the inverse ranks of selected messages.

Evaluation results based on the collected feedback dataset show that selected messages are sent to the GPT-4o-mini generative model, which uses the selected messages to generate a response. A reward model is subsequently trained using the feedback dataset. On average, 30% of the displayed messages are favored by users, and users can quickly find relevant content without manually browsing through messages. Furthermore, of the 61 feedback messages collected, 56 contain at least one selected message, indicating that the program has a 91.8% success rate in being functional, uplifting, and introducing users to new perspectives.

Keywords: Text generation, Empathetic messages, sentence vector representation, retrieval, cosine distance, GPT-4o-mini, Persian Messages, application



**Iran University of Science and Technology
Computer Engineering Department**

Empathetic Chatbot

Bachelor of Science Thesis in Computer Engineering

By:

Golbarg Sepehrara

Supervisor:

Dr.Seyyed Sauleh Eetemadi

January 2025