پردازش زبان‌های طبیعی

نیم‌سال دوم ۱۴۰۳-۱۴۰۴

مدرس: دکتر عسگری

دانشکده مهندسی کامپیوتر

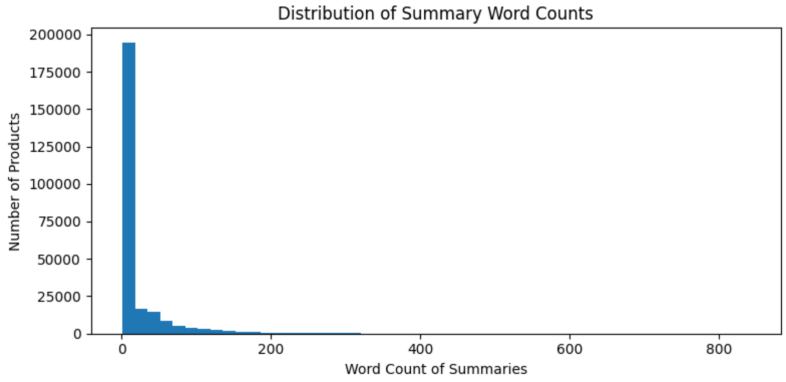
گزارش تمرین سری دوم، دیجیکالا

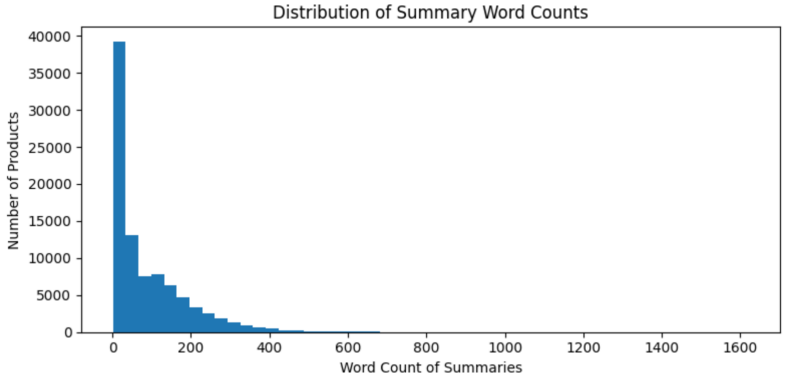
نام و نام‌خانوادگی: سینا بیرامی، سینا دانشگر، الهه ظهیری

شماره دانشجویی: ۴۰۰۱۰۵۴۳۳، ۴۰۱۱۰۰۳۶۸، ۴۰۰۱۰۳۵۱۶

۱ پیش‌پردازش داده‌ها

برای شروع و درک بهتر داده‌ها، ابتدا روی فایل کامنت‌ها ستون‌ها را بر اساس آیدی دسته‌بندی کرده و تعداد کامنت‌های هر آیدی را محاسبه کرده و آماره‌های آن را بررسی کردیم. از آنجایی که فایل "کامنت‌ها" را بررسی می‌کردیم طبیعتا تعداد کامنت‌ها برای هیچ آیدی‌ای صفر نبود و با وجود این اشتباه وارد مرحلهٔ خلاصه‌سازی کامنت‌ها شدیم. در این مرحله ابتدا با استفاده از رندوم سمپلینگ یک پنجم از داده‌ها را انتخاب کرده و به روش TF-IDF از این داده‌ها خلاصه ساختیم، پس از بررسی آماری متوجه شدیم که بسیاری از محصولات کامنت ندارند و در نتیجه فایل محصولات را آنالیز کردیم به این صورت که کتگوری‌هایی که بیشترین محصول با تعداد صفر کامنت داشتند را پیدا کرده و از فایل پروداکت حذف کردیم. سپس دوباره از این داده‌های جدید زیرمجموعه‌ای شامل یک پنجم داده‌ها ساختیم و مراحل ساخت خلاصه را انجام دادیم. نمودارهای این مراحل را قبل و بعد از حذف محصولات بدون کامنت در زیر مشاهده می‌کنید:





در خلاصه‌های ساخته‌شده طبق مشاهداتمان نقاط قوت و ضعف برخی محصولات که متن‌هایی مانند "ندارد"، "خوب" و ... را حذف کردیم. سپس تابع is-weak را تعریف و روی دیتا صدا زدیم که این تابع با استفاده از تعداد کلمات کامنت‌ها و وجود/عدم وجود نقاط قوت و ضعف و غیره خلاصه‌های ضعیف را مشخص می‌کند و آن‌ها را نیز حذف کردیم. سپس با یک قالب ثابت ستون full-summary را به داده‌ها اضافه کردیم. قبل از وارد شدن به مرحلهٔ بعدی در این بخش ابتدا

تلاش کردیم با بعضی از مدل‌های فاین‌تیون شده برای زبان فارسی برای خلاصه‌هایی که تا این مرحله ساخته شد، سوال تولید کنیم که تلاش ناموفقی بود و توابع اصلی ساخت سوال را در بخش بعدی تشریح کرده‌ایم. برای اتمام این مرحله تمام توابع تست‌شده روی یک پنجم داده‌ها را روی کل داده‌ها پیاده‌سازی کرده و براساس تعداد واژه‌های خلاصهٔ هر محصول مرتب‌سازی کردیم. سپس ۳۰۰۰تای اول را در یک فایل قرار داده و با این دیتاست آماده شده مراحل بعدی را انجام دادیم.

۲ تولید سوال

برای تسک ساخت ۵ سوال برای هر متن ابتدا با استفاده از مدل‌های مناسب برای زبان فارسی تلاش کردیم سوالات را بسازیم، در ادامه به طور مختصر هر کدام از آن‌ها را شرح می‌دهیم.

myrkur/persian-question-generator
یک مدل Seq۲Seq مبتنی بر معماری ترنسفورمر است که به‌طور خاص برای تولید سؤال از متن‌های فارسی آموزش‌دیده. در نوت‌بوک هم برای تولید خودکار پرسش (QG) و هم برای تسک «خلاصه‌سازی» (با استفاده از همان مدل) به کار رفته است.

HooshvareLab/bert-fa-base-uncased-ner-peyma
یک مدل BERT پیش‌آموزش‌دیده‌ی بدون حروف بزرگ (uncased) برای زبان فارسی، مناسب تشخیص موجودیت‌های اسمی (NER). این مدل با استفاده از $token - classification\ pipeline$ به صورت ساده و خودکار، نام اشخاص، مکان‌ها، سازمان‌ها و... را از متن استخراج می‌کند.

universitytehran/PersianMind-v1.0
یک مدل زبانی CausalLM برای تولید متن پیوسته و طبیعی به زبان فارسی. این مدل نیز با استفاده از $text - pipeline$ $generation$ برای تولید سؤال از خلاصه نظرات محصول استفاده شده است.

سوالات ساخته‌شده با این مدل‌ها کیفیت مطلوب را نداشتند به همین دلیل تصمیم گرفتیم که ابتدا خلاصه‌ها را به انگلیسی ترجمه کرده و سپس سوالات را بسازیم، اما در عملی‌کردن این ایده در لود کردن مدل‌ها دچار مشکل شدیم و از API ها استفاده کردیم که Gemini ناموفق و gpt-۴o-mini موفق بود و با تنظیم پرامت‌های مناسب ۱۵۰۰۰ سوال با کیفیت مطلوب تولید کردیم. سپس با دسته‌بندی محصولات و بررسی خلاصه‌های آن‌ها ۵۰ سوال به طور دستی برای ارزیابی مدل‌ها در مراحل بعدی آماده کردیم.

۳ آموزش مدل زبانی و خروجی گرفتن از مدل‌ها

بازیابی اطلاعات با روش TF-IDF
در این بخش، ابتدا داده‌های تکراری را حذف کردیم تا روابط و پردازش بین محصولات و سوالات بهتر درک شوند. سپس متون ستون full-summary که شامل خلاصه‌های توصیفی از محصولات است، به عنوان مجموعهٔ اسناد مرجع در نظر گرفته شدند.
برای بردارسازی متن‌ها، از TfIdfVectorizer در پایتون استفاده کردیم. بعد از آموزش بردارساز، هر پرسش کاربر نیز به فضای TF-IDF نگاشت شد و شباهت کسینوسی بین آن و تمام خلاصه‌های محصولات محاسبه گردید. برای هر پرسش، ۶ مورد با بالاترین شباهت به عنوان کاندید پاسخ انتخاب شدند.
ارزیابی عملکرد مدل با دو معیار Hit@۶ و MRR@۶ انجام شد:
- Hit@۶ درصد پرسش‌هایی که پاسخ صحیح آن‌ها در بین ۶ نتیجه برتر حضور دارد.
- MRR@۶ میانگین معکوس رتبه پاسخ صحیح در بین نتایج برتر، که حساسیت بیشتری به رتبه دارد.
علاوه بر رتبه‌بندی کلی، برای هر پرسش، سه نتیجهٔ برتر همراه با بهترین جملهٔ شاهد (best evidence) از متن خلاصه، استخراج و در فایل خروجی ذخیره شد تا تحلیل کیفی نتایج نیز ممکن باشد.
عملکرد بالای TF-IDF در این پروژه تا حد زیادی ناشی از این واقعیت بود که پرسش‌ها و خلاصه‌های محصولات، اشتراک واژگانی و عبارت‌های مشابه بسیار زیادی داشتند. به همین دلیل، یک روش کلاسیک مبتنی بر فراوانی واژه مانند TF-IDF توانست حتی از مدل چندزبانه در حالت Zero-Shot عملکرد بهتری ارائه دهد.

این نتیجه همچنین خط‌مبنای بسیار خوبی ایجاد کرد تا بتوانیم کیفیت مدل‌های یادگیری عمیق را نسبت به آن بسنجیم. در ادامه نشان خواهیم داد که مدل فاین‌تیون‌شده ما، با اختلاف قابل توجه، از مدل Zero-Shot قوی‌تر شده و توانسته دقت و رتبه‌بندی نتایج را به طور محسوسی بهبود دهد.

ارزیابی زیروشات (مدل پایهٔ $GLOT500$)

هدف:

برای ارزیابی عملکرد مدل‌های زبانی پیشرفته بدون آموزش اضافی، از مدل GLOT۵۰۰-base در حالت zero-shot استفاده کردیم. هدف نهایی مقایسهٔ این رویکرد با TF-IDF و همینطور مقایسه با بخش بعد (آموزش رو دادهٔ خاص) است که توانایی مدل در فهم معنا و روابط ضمنی بین پرسش و خلاصهٔ محصول را به چالش می‌کشد.

روش:

هر پرسش و خلاصه محصول جداگانه توسط مدل GLOT۵۰۰ به یک بردار embedding تبدیل شد. شباهت بین بردار پرسش، و بردار خلاصه، با شباهت کسینوسی محاسبه گردید. برای هر پرسش، سه نتیجه برتر ذخیره شدند.

نتایج:

در کتگوری‌های عمومی، قدرت مدل نسبت به TF-IDF در شناسایی معانی مشابه با واژگان متفاوت دیده شد، اما با این حال در حوزه‌های خاص (مثلا دسته‌بندی‌های محصولی یا اصطلاحات تخصصی) به دلیل عدم تطبیق کامل با دامنهٔ داده‌ها، قدرت آن پایین‌تر از حد مطلوب بود. این موضوع انگیزهٔ اصلی برای حرکت به سمت فاین‌تیون کردن این مدل شد.

فاین‌تیون کردن مدل GLOT۵۰۰

هدف:

هدف این بخش، بهبود عملکرد مدل پایهٔ GLOT۵۰۰ (زیرو شات) در بازیابی پاسخ صحیح به پرسش‌های مرتبط با محصولات بود. مدل پایه، به صورت پیش‌فرض روی داده‌های چندزبانه آموزش دیده است. (البته تجربه‌های ابتدایی استفاده از این مدل نشان داد که آموزش آن روی داده‌های فارسی چندان غنی نبوده و می‌توان در کارهای آتی، این مدل را برای فارسی بهبود داد)

داده‌ها:

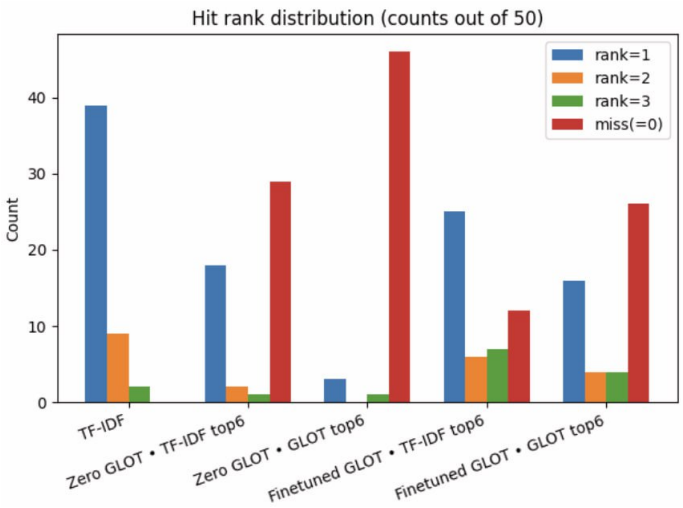
دادهٔ آموزش شامل دو ستون اصلی بود؛ یکی full-summary که خلاصهٔ توصیفی از محصولات است، و دیگری qa-pairs که مجموعه‌ای از پرسش و پاسخ‌ها است که برای همان محصول ثبت شده‌اند. از هر جفت پرسش_پاسخ، پرسش را به عنوان query و خلاصه محصول مربوطه را به عنوان $positive\ context$ اضافه کردیم تا مدل یاد بگیرد فقط بین پرسش و خلاصهٔ مرتبط، شباهت بالا بدهد.

روش:

برای این کار از روش $Contrastive\ Learning$ با معماری $dual\ encoder$ استفاده کردیم. - هر پرسش و هر خلاصه محصول، جداگانه با انکودر GLOT۵۰۰ به یک بردار امبدینگ تبدیل می‌شوند. - با استفاده از تابع خطای Multi-pleNegativesRankingLoss، مدل یاد می‌گیرد که شباهت کسینوسی ($Cosine\ Similarity$) بین پرسش و خلاصه صحیح، ماکزیمم شود و بین پرسش و خلاصه‌های دیگر، مینیمم شود. این روش، دو مزیت بزرگ دارد؛ اول اینکه بعد از آموزش، همهٔ خلاصه‌ها یک‌بار انکد می‌شوند و جستجو با مقایسه کسینوسی انجام می‌شود. دوم اینکه برای هر پرسش دیگر لازم نیست کل متن‌ها را مجددا پردازش کنیم و در نتیجه برای داده‌های با مقیاس بزرگ، انعطاف‌پذیر است.

خروجی و ارزیابی:

بعد از فاین‌تیون، مدل روی ۵۰ پرسش ارزیابی، اجرا شد. برای هر پرسش، سه پاسخ برتر بر اساس بیشترین شباهت کسینوسی بین embedding پرسش و embedding خلاصه‌ها انتخاب شد. نتایج نشان داد که مدل فاین‌تیون شده، در مقایسه با مدل پایه، دقت بالاتری در رتبه‌بندی پاسخ‌های صحیح دارد.



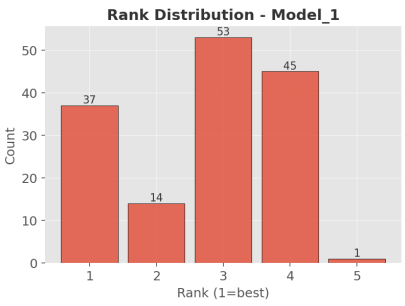
۴ ارزیابی انسانی با استفاده از *Label Studio*

پس از آماده‌سازی ۹ پاسخ برای هر سوال، همهٔ پاسخ‌ها را در یک فایل جمع‌آوری کرده و با استفاده از لیبل استودیو رنک‌بندی کردیم. استراتژی رنک‌بندی انتخاب‌شده در گروه به صورت زیر است:

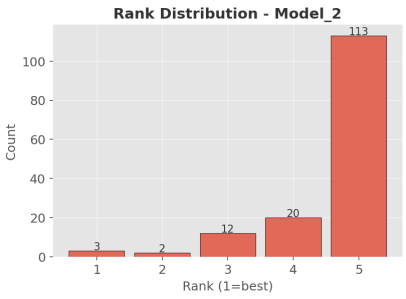
۱ = تشخیص درست محصول و دلیل خوب.
۲ = تشخیص درست محصول و دلیل نامناسب.
۳ = تشخیص نادرست محصول (کتگوری صحیح)و دلیل خوب.
۴ = تشخیص نادرست محصول(کتگوری غلط) و دلیل خوب.
۵ = تشخیص نادرست و دلیل بد و موارد دیگر که در رنک‌های قبلی نمی‌گنجد.
فایل اکسپورت‌شده از لیبل استودیو در فایل نهایی موجود است.

۵ تحلیل نتایج

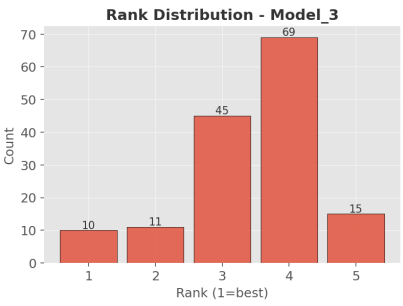
تصاویر زیر هستوگرام تعداد رنک‌های هر مدل را نشان می‌دهد. همانطور که انتظار می‌رفت فاین‌تیون کردن مدل، آن را بهبود داده‌است.



(a) Model 1



(b) Model 2



(c) Model 3

نتایج آماری روی مدل‌ها به شرح زیر است:

model	mean_rank	median_rank	std_rank	n
TF-IDF	2.727	3.000	1.158	150
GLOT500-base	4.587	5.000	0.853	150
GLOT500-Fine-Tuned	3.453	4.000	1.001	150

با توجه به منطق رنک‌ها هر چه میانگین رنک‌های پاسخ‌های یک مدل به ۱ نزدیک‌تر باشد یعنی آن مدل عملکرد بهتری داشته‌است.

تحلیل و تفسیر نتایج ارزیابی مدل‌ها

در این مطالعه، عملکرد سه رویکرد متفاوت بازیابی پاسخ شامل مدل آماری مبتنی بر TF-IDF، مدل زبانی چندزبانه GLOT۵۰۰ به‌صورت zero-shot، و نسخهٔ بهینه‌سازی‌شده (fine-tuned) همین مدل زبانی مورد مقایسه قرار گرفت. هر مدل برای ۵۰ سؤال ارزیابی، سه خروجی ارائه داد و این خروجی‌ها به‌صورت انسانی در مقیاس ۱ (بهترین) تا ۵ (ضعیف‌ترین) رتبه‌بندی شدند.

شاخص‌های آماری کلیدی

نتایج آماری نشان داد که مدل مبتنی بر TF-IDF با میانگین رتبهٔ ۲/۷۲۷ و میانهٔ رتبهٔ ۳، بر سایر مدل‌ها برتری داشته است. در مقابل، مدل GLOT۵۰۰ بدون آموزش مجدد با میانگین رتبهٔ ۴/۵۸۷ و میانهٔ ۵ ضعیف‌ترین عملکرد را ثبت کرده و مدل GLOT۵۰۰ بهینه‌سازی‌شده با میانگین رتبهٔ ۳/۴۵۳ عملکردی بین دو مدل دیگر داشته است. همچنین، پراکندگی نتایج (انحراف معیار) برای مدل TF-IDF بالاتر بوده که نشان‌دهندهٔ تغییرپذیری بیشتر عملکرد آن در میان سؤالات است.

تأثیر ویژگی‌های مجموعهٔ سؤال‌ها

ویژگی بارز مجموعه‌سؤال‌های مورد استفاده در این ارزیابی، همپوشانی واژگانی بالا با متن کانتکست بود؛ به این معنا که بسیاری از واژگان کلیدی موجود در سؤال‌ها عیناً در متن منبع نیز حضور داشتند. این ویژگی یک مزیت ذاتی برای رویکردهای آماری مبتنی بر فراوانی واژگان نظیر TF-IDF ایجاد می‌کند، چراکه چنین مدل‌هایی اساس رتبه‌بندی خود را بر شباهت‌های سطح واژه بنا می‌کنند و در حضور تطابق‌های مستقیم، می‌توانند بخش‌های مرتبط متن را با دقت بالایی بازیابی کنند.
در مقابل، مدل‌های زبانی عمیق مانند GLOT۵۰۰، چه به‌صورت zero-shot و چه در حالت fine-tuned، بر بازنمایی‌های معنایی و تطابق‌های مفهومی متکی‌اند. این رویکرد، هرچند برای سناریوهایی با شباهت واژگانی پایین مزیت دارد، در این آزمایش خاص سبب شده که مدل‌ها الزماً همان بخش‌های واژگانی دقیق را انتخاب نکنند و در نتیجه در مقایسه با TF-IDF امتیاز کمتری کسب کنند.

مقایسهٔ مدل‌های زبانی

عملکرد ضعیف مدل GLOT۵۰۰ در حالت zero-shot را می‌توان به فقدان هرگونه سازگاری با دامنهٔ دادهٔ خاص این مطالعه نسبت داد. نسخهٔ بهینه‌سازی‌شدهٔ این مدل، اگرچه نسبت به نسخهٔ اولیه بهبود نسبی در رتبه‌های میانی و پایین نشان داده است، اما همچنان نتوانسته بر مزیت واژگان‌محور مدل TF-IDF غلبه کند. این امر نشان می‌دهد که حتی پس از فاین‌تیون، در شرایطی که شباهت لغوی بسیار بالا باشد، مدل‌های آماری کلاسیک ممکن است کارایی بیشتری داشته باشند.

۶ امتیازی

در این بخش، هدف شناسایی داده‌های تکراری (Duplicate) در مجموعهٔ محصولات بود. از آنجا که مدل GLOT۵۰۰ در مراحل قبل با روش *Contrastive Learning* روی داده‌های پرسش و پاسخ و متن متناظر با آن‌ها فاین‌تیون شده بود، از خروجی این مدل (بردارهای امبدینگ) برای تشخیص شباهت استفاده شد.

مراحل کلی انجام کار:

مدل را در مرحله قبل ذخیره کرده بودیم تا برای استفاده‌های آتی بارگذاری کنیم. سپس، تمام خلاصه متن‌ها ($full - summary$) هایی که از محصولات داشتیم، به وسیلهٔ مدل مورد نظر، با استفاده از پارامتری به نام normalize_embeddings=True به بردارهایی با طول ثابت نرمال‌شده تبدیل شدند. این نرمال‌سازی، باعث می‌شود تا ضرب داخلی این بردارها، معادل شباهت کسانی ($Cosine\ Similarity$) باشد.

محاسبهٔ ماتریس شباهت:

با استفاده از ضرب ماتریسی کتابخانه PyTorch، ماتریس شباهت کسانی بین همهٔ نمونه‌ها محاسبه شد. قطر اصلی ماتریس برابر با ۱ بود (شباهت هر داده با خودش) اما برای جلوگیری از تطبیق یک نمونه با خودش، مقدار آن را به ۱- تغییر دادیم.

مرحلهٔ اصلی شناسایی:

دو استراتژی در این مرحله پیاده کردیم. Threshold-based و Top-K-based: تمام جفت‌هایی را که امتیاز شباهت آن‌ها بزرگتر از یک حد مشخص بود را نگه می‌داشتیم. برای مثال در کد ما، این عدد ۸۵.۰ بود. Top-K-based: برای هر نمونه، نزدیک‌ترین همسایه‌ها از نظر شباهت انتخاب می‌شدند (چیزی شبیه $nearest\ neighbors$)

چالش‌های پیاده‌سازی:

۱. به منظور جلوگیری از شمارش دوباره، جفت‌ها به صورت (i، j) با ترتیب ثابت ذخیره شدند تا (i، j) و (j، i) را دو بار حساب نکنیم. ۲. تست اولیه روی ۵۰۰ داده انجام شد که نتایج مطلوبی به همراه داشت. سعی کردیم تعداد duplicate ها را در دیتاست به طور همگن توزیع کنیم تا خروجی راحت‌تر مانیتور شود. ۳. مقدار ترشولد رابطه مستقیمی با حجم خروجی

داشت، هرچقدر پایین‌تر مقداردهی شود، نمونه‌های بیشتری شناسایی می‌شوند (اما خطر شناسایی $False\ Positive$ وجود دارد) و هرچه آستانه بالاتر باشد، دقت و قطعیت بیشتر است، اما ممکن است بعضی نمونه‌های واقعا تکراری از دست بروند.

مزایای این کد:

به کتاب‌خانه‌های FAISS که برای جست‌وجوی سریع‌تر (روی GPU) استفاده می‌شوند، نیازی نبود و درنتیجه روی CPU مشکلی نداشتیم. مقیاس پذیری بالا برای مجموعه‌های تا چندهزار نمونه (تا ۵ هزارتا تست شد) استفادهٔ مستقیم از دانش مدل فاین‌تیون شده، بدون نیاز به یادگیری مجدد برای این تسک خاص. این روش از متدهایی که صرفا متنی هستند، دقت بسیار بالاتری دارد.