



Patterns of Application Development Using AI

Obie Fernandez

Foreword by Gregor Hohpe



Leanpub

النسخة العربية

أنماط تطوير التطبيقات باستخدام الذكاء الاصطناعي (النسخة العربية)

Fernandez Obie

هذا الكتاب للبيع على

<http://leanpub.com/patterns-of-application-development-using-ai-ar>

تم نشر هذه النسخة في 23-01-2025



هذا كتاب من **Leanpub**. تمكن **Leanpub** المؤلفين والناشرين بعملية النشر الرشيق. النشر الرشيق **Publishing Lean** هو فعل نشر كتاب إلكتروني قيد التقدم باستخدام أدوات خفيفة والعديد من التكرارات للحصول على ردود فعل القراء، والقيام بالتعديلات اللازمة حتى يكون لديك الكتاب المناسب وبناء القوة الجذبية بمجرد القيام بذلك.

Fernandez Obie 2025 ©

نشر الكتاب عبر تويتر!

الرجاء مساعدة Fernandez Obie بتغريد كلمة عن هذا الكتاب على [Twitter](#)!

التغريدة المقترحة لهذا الكتاب [#poaduai](#).

لمعرفة ما يقوله الآخرون عن هذا الكتاب، يمكنك النقر على هذا الوسم على تويتر:

[#poaduai](#)

إلى ملكتي الجبارة، مُلهمتي، نوري وحبي، فيكتوريا

Fernandez Obie أيضا بواسطة

AI Using Development Application of Patterns

Way 8 Rails The

Way 7 Rails The

Way Rails The XML

Serverless

Node de Principiante Libro El

Enterprise Lean The

المحتويات

i	تمهيد بقلم جريجور هوه
ii	مقدمة
iii	حول الكتاب
iii	حول أمثلة الشيفرة
iii	ما لا أعطيه
iii	لمن هذا الكتاب
iii	بناء مفردات مشتركة
iii	المشاركة
iii	شكر وتقدير
iv	ما قصة الرسوم التوضيحية؟
iv	حول النشر المرن
v	نبذة عن المؤلف
1	مقدمة
2	أفكار حول هندسة البرمجيات
3	ما هو النموذج اللغوي الكبير؟
4	فهم الاستدلال
22	التفكير في الأداء
23	تجربة نماذج LLM المختلفة
24	أنظمة الذكاء الاصطناعي المركبة

31	الجزء الأول: النهج والتقنيات الأساسية
32	تضييق المسار
34	الفضاء الكامن: واسع بشكل لا يمكن إدراكه
37	كيف يتم "تضييق" المسار
40	النماذج الختام مقابل النماذج المدربة على التعليمات
47	هندسة المطالبات
60	تقطير الموجهات
66	ماذا عن الضبط الدقيق؟
67	استرجاع المعلومات المعزز للتوليد (RAG)
67	ما هو استرجاع المعلومات المعزز للتوليد؟
67	كيف يعمل RAG؟
67	لماذا نستخدم RAG في تطبيقاتك؟
67	تنفيذ RAG في تطبيقك
68	تجربة المقترحات
68	أمثلة واقعية للتوليد المعزز بالاسترجاع
69	تحسين الاستعلام الذكي (IQO)
69	إعادة الترتيب
69	تقييم RAG (RAGAs)
71	التحديات والنظرة المستقبلية
73	تعدد العمال
74	عمال الذكاء الاصطناعي كمكونات مستقلة قابلة لإعادة الاستخدام
75	إدارة الحسابات
76	تطبيقات التجارة الإلكترونية
85	تطبيقات الرعاية الصحية
87	عامل الذكاء الاصطناعي كمدير للعمليات
90	دمج عمال الذكاء الاصطناعي في هيكلية تطبيقك
93	قابلية التركيب وتنسيق العمال الاصطناعيين

101	الجمع بين معالجة اللغات الطبيعية التقليدية ونماذج اللغة الكبيرة
104	استخدام الأدوات
104	ما هو استخدام الأدوات؟
106	إمكانات استخدام الأدوات
107	سير عمل استخدام الأدوات
120	أفضل الممارسات لاستخدام الأدوات
123	تركيب وتسلسل الأدوات
125	الاتجاهات المستقبلية
127	معالجة التدفق
128	تنفيذ ReplyStream
134	حلقة الحادثة
135	الاستمرار التلقائي
138	الخاتمة
139	البيانات ذاتية التصحيح
141	دراسة حالة عملية: إصلاح JSON المعطوب
145	الاعتبارات وموانع الاستخدام
157	توليد المحتوى السياقي
158	التخصيص
159	الإنتاجية
161	التكرار السريع والتجريب
163	التوطين المدعوم بالذكاء الاصطناعي
164	أهمية اختبار المستخدم والتغذية الراجعة
166	واجهة المستخدم التوليدية
167	توليد النصوص لواجهات المستخدم
175	تعريف واجهة المستخدم التوليدية
177	مثال

179	التحول نحو التصميم الموجه نحو النتائج
181	التحديات والاعتبارات
182	النظرة المستقبلية والفرص
185	تنسيق سير العمل الذكي
186	الحاجة التجارية
186	الفوائد الرئيسية
187	الأنماط الرئيسية
189	معالجة الاستثناءات والتعافي
192	تنفيذ تنسيق سير العمل الذكي عملياً
205	المراقبة وتسجيل السجلات
208	اعتبارات قابلية التوسع والأداء
212	اختبار والتحقق من صحة مسارات العمل

الجزء 2: الأنماط 220

221	هندسة المطالبات
222	سلسلة التفكير
223	تبديل الوضع
224	تعيين الدور
225	كائن المطالبة
226	قالب المطالبة
227	المدخلات والمُخرجات المُهيكلَة
228	تسلسل المطالبات
229	معيد صياغة المحفزات
230	تسييج الاستجابة
231	محلل الاستعلامات
232	معيد صياغة الاستعلامات
233	فن تحريك الدمى

234	المكونات المنفصلة
235	المسند
236	واجهة وسيطة لل API
238	مُفسر النتائج
239	الآلة الافتراضية
239	المواصفات والاختبار
241	التدخل البشري في العملية (HITL)
241	الأنماط عالية المستوى
242	التصعيد
243	حلقة التغذية الراجعة
244	إشعاع المعلومات السليبي
246	صنع القرار التعاوني (CDM)
247	التعلم المستمر
247	الاعتبارات الأخلاقية
247	التطورات التكنولوجية والنظرة المستقبلية
249	معالجة الأخطاء الذكية
249	مناهج معالجة الأخطاء التقليدية
250	تشخيص الأخطاء السياقي
251	الإبلاغ الذكي عن الأخطاء
252	الوقاية التنبؤية من الأخطاء
252	التعافي الذكي من الأخطاء
253	التواصل المخصص بشأن الأخطاء
254	سير عمل معالجة الأخطاء التكيفي
255	ضبط الجودة
256	التقييم
258	الضوابط الوقائية
258	حواجز الحماية والتقييمات: وجهان لعملة واحدة

260	مسرد المصطلحات
260	مسرد المصطلحات
265	فهرس
265	فهرس

تمهيد بقلم جريجور هوه

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

مقدمة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

حول الكتاب

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

حول أمثلة الشيفرة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ما لا أغطيه

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

لمن هذا الكتاب

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

بناء مفردات مشتركة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المشاركة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

شكر وتقدير

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ما قصة الرسوم التوضيحية؟

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

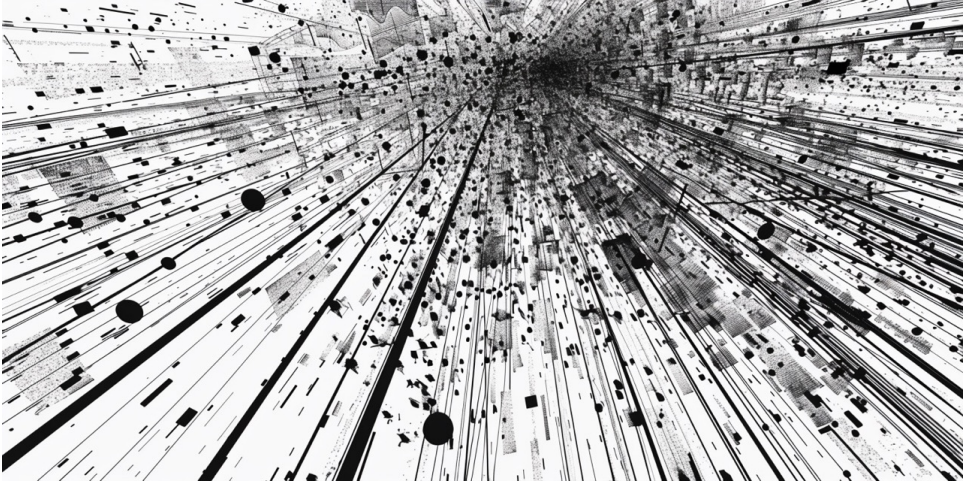
حول النشر المرن

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

نبذة عن المؤلف

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مقدمة



إذا كنت متحمساً للبدء في دمج النماذج اللغوية الكبيرة للذكاء الاصطناعي (LLMs) في مشاريعك البرمجية، فلا تتردد في الغوص مباشرة في الأنماط وأمثلة الشيفرة المقدمة في الفصول اللاحقة. ومع ذلك، لتقدير قوة وإمكانات هذه الأنماط بشكل كامل، من المفيد أخذ لحظة لفهم السياق الأوسع والنهج المتناسك الذي تمثله.

هذه الأنماط ليست مجرد مجموعة من التقنيات المنعزلة، بل هي إطار عمل موحد لدمج الذكاء الاصطناعي في تطبيقاتك. أستخدم Rails on Ruby، لكن هذه الأنماط يجب أن تعمل في أي بيئة برمجة أخرى تقريباً. وهي تعالج مجموعة واسعة من الاهتمامات، من إدارة البيانات وتحسين الأداء إلى تجربة المستخدم والأمان، مما يوفر مجموعة أدوات شاملة لتعزيز ممارسات البرمجة التقليدية بقدرات الذكاء الاصطناعي. تتناول كل فئة من الأنماط تحدياً أو فرصة محددة تنشأ عند دمج مكونات الذكاء الاصطناعي في تطبيقك.

من خلال فهم العلاقات والتآزر بين هذه الأنماط، يمكنك اتخاذ قرارات مدروسة حول مكان وكيفية تطبيق الذكاء الاصطناعي بأكثر الطرق فعالية.

الأنماط ليست حلولاً توجيهية ولا ينبغي التعامل معها على هذا النحو. فهي مصممة لتكون لبنات بناء قابلة للتكيف يجب تخصيصها وفقاً للمتطلبات والقيود الفريدة لتطبيقك الخاص. يعتمد التطبيق الناجح لهذه الأنماط (مثل أي أنماط أخرى في مجال البرمجيات) على فهم عميق لمجال المشكلة، واحتياجات المستخدم، والهيكल التقني الشامل لمشروعك.

أفكار حول هندسة البرمجيات

بدأت البرمجة في الثمانينيات وكنت منخرطاً في مشهد الهاكرز، ولم أفقد أبداً عقلية الهاكر حتى بعد أن أصبحت مطور برمجيات محترفاً. منذ البداية، كان لدي دائماً شك صحي حول القيمة التي يقدمها مهندسو البرمجيات في أبراجهم العاجية.

أحد الأسباب التي تجعلني متحمساً شخصياً للتغييرات التي أحدثتها هذه الموجة القوية الجديدة من تكنولوجيا الذكاء الاصطناعي هو تأثيرها على ما نعتبره قرارات هندسة البرمجيات. إنها تتحدى المفاهيم التقليدية لما يشكل الطريقة "الصحيحة" لتصميم وتنفيذ مشاريعنا البرمجية. كما أنها تتحدى ما إذا كان لا يزال يمكن اعتبار الهندسة المعمارية في المقام الأول الأجزاء من النظام التي يصعب تغييرها، حيث يجعل تعزيز الذكاء الاصطناعي من السهل أكثر من أي وقت مضى تغيير أي جزء من مشروعك، في أي وقت.

ربما نحن ندخل في سنوات الذروة لنهج "ما بعد الحداثة" في هندسة البرمجيات. في هذا السياق، يشير مصطلح ما بعد الحداثة إلى تحول أساسي بعيداً عن النماذج التقليدية، حيث كان المطورون مسؤولين عن كتابة وصيانة كل سطر من الشيفرة. بدلاً من ذلك، يتبنى فكرة تفويض المهام، مثل معالجة البيانات والخوارزميات المعقدة، وحتى أجزاء كاملة من منطق التطبيق، إلى مكتبات الطرف الثالث وواجهات برمجة التطبيقات الخارجية. يمثل هذا التحول ما بعد الحداثة خروجاً كبيراً عن الحكمة التقليدية لبناء التطبيقات من الصفر، وهو يتحدى المطورين لإعادة التفكير في دورهم في عملية التطوير.

لطالما آمنت بأن المبرمجين الجيدين يكتبون فقط الشيفرة الضرورية للغاية للكفاءة، استناداً إلى تعاليم Larry Wall وغيره من المفكرين البارزين في مجال الهاكر مثله. من خلال تقليل كمية الشيفرة المكتوبة، يمكننا التحرك بشكل أسرع، وتقليل مساحة الأخطاء، وتبسيط الصيانة، وتحسين الموثوقية الشاملة لتطبيقاتنا.

تسمح لنا الشيفرة الأقل بالتركيز على المنطق التجاري الأساسي وتجربة المستخدم، مع تفويض العمل الآخر إلى خدمات أخرى.

الآن بعد أن أصبحت الأنظمة المدعومة بالذكاء الاصطناعي قادرة على التعامل مع المهام التي كانت في السابق حكرًا على الشيفرة المكتوبة من قبل البشر، يجب أن نكون قادرين على أن نكون أكثر إنتاجية ومرونة، مع تركيز أكبر من أي وقت مضى على خلق قيمة تجارية وتجربة مستخدم.

بالطبع هناك مقايضات لتفويض أجزاء كبيرة من مشروعك إلى أنظمة الذكاء الاصطناعي، مثل فقدان السيطرة المحتمل، والحاجة إلى آليات مراقبة وتغذية راجعة قوية. لهذا السبب يتطلب الأمر مجموعة جديدة من المهارات والمعرفة، بما في ذلك على الأقل بعض الفهم الأساسي لكيفية عمل الذكاء الاصطناعي.

ما هو النموذج اللغوي الكبير؟

النماذج اللغوية الكبيرة (LLMs) هي نوع من نماذج الذكاء الاصطناعي التي حظيت باهتمام كبير في السنوات الأخيرة، منذ إطلاق GPT-3 بواسطة OpenAI في عام 2020. تم تصميم النماذج اللغوية الكبيرة لمعالجة وفهم وتوليد اللغة البشرية بدقة وطلاقة ملحوظة. في هذا القسم، سنلقي نظرة موجزة على كيفية عمل النماذج اللغوية الكبيرة ولماذا تعتبر مناسبة لبناء مكونات النظام الذكي.

في جوهرها، تعتمد النماذج اللغوية الكبيرة على خوارزميات التعلم العميق، وتحديدًا الشبكات العصبية. تتكون هذه الشبكات من عقد مترابطة، أو خلايا عصبية، تقوم بمعالجة ونقل المعلومات. الهيكل المفضل للنماذج اللغوية الكبيرة غالباً ما يكون نموذج الحول، الذي أثبتت فعاليته العالية في التعامل مع البيانات المتسلسلة مثل النص.

تعتمد نماذج الحولات على آلية الانتباه وتُستخدم بشكل أساسي للمهام التي تتضمن بيانات متسلسلة، مثل معالجة اللغة الطبيعية. تعالج الحولات البيانات المدخلة دفعة واحدة بدلاً من معالجتها بشكل متسلسل، مما يتيح لها التقاط العلاقات طويلة المدى بشكل أكثر فعالية. تحتوي على طبقات من آليات الانتباه التي تساعد النموذج على التركيز على أجزاء مختلفة من البيانات المدخلة لفهم السياق والعلاقات.

تتضمن عملية تدريب النماذج اللغوية الكبيرة تعريض النموذج لكميات هائلة من البيانات النصية، مثل الكتب والمقالات والمواقع الإلكترونية ومستودعات الشيفرات البرمجية. خلال التدريب، يتعلم النموذج التعرف على الأنماط والعلاقات والهيكل داخل النص. يلتقط الخصائص الإحصائية للغة، مثل قواعد النحو وارتباطات الكلمات والمعاني السياقية.

إحدى التقنيات الرئيسية المستخدمة في تدريب النماذج اللغوية الكبيرة هي التعلم غير الموجه. وهذا يعني أن النموذج يتعلم من البيانات دون توجيه أو تصنيف صريح. يكتشف الأنماط والتمثيلات من تلقاء نفسه من خلال تحليل التواجد المشترك للكلمات والعبارات في بيانات التدريب. هذا يسمح للنماذج اللغوية الكبيرة بتطوير فهم عميق للغة وتعقيداتها.

جانب مهم آخر من النماذج اللغوية الكبيرة هو قدرتها على التعامل مع السياق. عند معالجة نص ما، تأخذ النماذج اللغوية الكبيرة في الاعتبار ليس فقط الكلمات الفردية ولكن أيضاً السياق المحيط. تأخذ في الاعتبار الكلمات والجمل وحتى الفقرات السابقة لفهم معنى النص والغرض منه. هذا الفهم السياقي يمكن النماذج اللغوية الكبيرة من إنتاج استجابات متماسكة وذات صلة. أحد الطرق الرئيسية التي نقيم بها قدرات نموذج لغوي كبير معين هو النظر في حجم السياق الذي يمكنهم اعتباره لتوليد الاستجابات.

بمجرد تدريبها، يمكن استخدام النماذج اللغوية الكبيرة لمجموعة واسعة من المهام المتعلقة باللغة. يمكنها إنتاج نص يشبه النص البشري، والإجابة عن الأسئلة، وتلخيص المستندات، وترجمة اللغات، وحتى كتابة الشيفرات البرمجية. تجعل تعددية استخدامات النماذج اللغوية الكبيرة منها قيمة لبناء مكونات النظم الذكية التي يمكنها التفاعل مع المستخدمين، ومعالجة وتحليل البيانات النصية، وإنتاج مخرجات ذات معنى. من خلال دمج النماذج اللغوية الكبيرة في بنية التطبيق، يمكنك إنشاء مكونات ذكاء اصطناعي تفهم وتعالج مدخلات المستخدم، وتولد محتوى ديناميكي، وتقدم توصيات أو إجراءات ذكية. لكن العمل مع النماذج اللغوية الكبيرة يتطلب اعتباراً دقيقاً لمتطلبات الموارد والمفاضلات في الأداء. النماذج اللغوية الكبيرة تتطلب قدرة حسابية عالية وقد تحتاج إلى قوة معالجة وذاكرة كبيرة (بمعنى آخر، المال) للتشغيل. معظمنا سيحتاج إلى تقييم الآثار المالية لدمج النماذج اللغوية الكبيرة في تطبيقاتنا والتصرف وفقاً لذلك.

فهم الاستدلال

الاستدلال يشير إلى العملية التي يولد بها النموذج تنبؤات أو مخرجات بناءً على بيانات جديدة لم يرها من قبل. إنها المرحلة التي يتم فيها استخدام النموذج المُدرَّب لاتخاذ قرارات أو توليد نص أو صور أو محتوى آخر استجابةً لمدخلات المستخدم.

خلال مرحلة التدريب، يتعلم نموذج الذكاء الاصطناعي من مجموعة بيانات كبيرة من خلال تعديل معاملاته لتقليل الخطأ في تنبؤاته. بمجرد تدريبه، يمكن للنموذج تطبيق ما تعلمه على بيانات جديدة. الاستدلال هو كيفية استخدام النموذج للأنماط والمعرفة المكتسبة لتوليد المخرجات.

بالنسبة للنماذج اللغوية الكبيرة، يتضمن الاستدلال أخذ نص تحفيزي أو نص مدخل وإنتاج استجابة متماسكة وذات صلة بالسياق، كتدفق من الرموز (والتي سنتحدث عنها قريباً). قد يكون هذا إجابة عن سؤال، أو إكمال جملة، أو توليد قصة، أو ترجمة نص، من بين العديد من المهام الأخرى.

على عكس الطريقة التي نفكر بها أنا وأنت، يحدث "تفكير" نموذج الذكاء الاصطناعي عبر الاستدلال كله في عملية واحدة بدون حالة. أي أن تفكيره محدود بعملية التوليد الخاصة به. يجب عليه حرفياً أن يفكر بصوت عالٍ، كما لو أنني سألتك سؤالاً وقبلت منك فقط إجابة بأسلوب "تيار الوعي".



النماذج اللغوية الكبيرة تأتي بأحجام وأنواع مختلفة

في حين أن جميع النماذج اللغوية الكبيرة الشائعة تقريباً تعتمد على نفس بنية المحولات الأساسية وتم تدريبها على مجموعات بيانات نصية ضخمة، فإنها تأتي بمجموعة متنوعة من الأحجام ويتم ضبطها بدقة لأغراض مختلفة. حجم النموذج اللغوي الكبير، الذي يُقاس بعدد المعاملات في شبكته العصبية، له تأثير كبير على قدراته. النماذج الأكبر التي تحتوي على معاملات أكثر، مثل 4GPT-، الذي يُشاع أنه يتباهى بـ 1 إلى 2 تريليون معامل، عادةً ما تكون أكثر معرفة وقدرة من النماذج الأصغر. ومع ذلك، تتطلب النماذج الأكبر أيضاً قوة حوسبة أكبر للتشغيل، مما يترجم إلى تكلفة أعلى عند استخدامها عبر استدعاءات واجهة برمجة التطبيقات. لجعل النماذج اللغوية الكبيرة أكثر عملية ومخصصة لحالات استخدام محددة، غالباً ما يتم الضبط الدقيق للنماذج الأساسية على مجموعات بيانات أكثر استهدافاً. على سبيل المثال، قد يتم تدريب نموذج لغوي كبير على مجموعة كبيرة من الحوارات لتخصيصه للذكاء الاصطناعي المحادثي. والبعض الآخر يتم تدريبه على الشيفرات البرمجية لتزويده بمعرفة البرمجة. هناك حتى نماذج مدربة خصيصاً للتفاعلات على غلط لعب الأدوار مع المستخدمين!

نماذج الاسترجاع مقابل النماذج التوليدية

في عالم نماذج اللغة الكبيرة (LLMs) هناك نهجان رئيسيان لتوليد الاستجابات: النماذج القائمة على الاسترجاع والنماذج التوليدية. لكل نهج نقاط قوته وضعفه، وفهم الاختلافات بينهما يمكن أن يساعدك في اختيار النموذج المناسب لحالة الاستخدام الخاصة بك.

النماذج القائمة على الاسترجاع

النماذج القائمة على الاسترجاع، والمعروفة أيضاً باسم نماذج استرجاع المعلومات، تولد الاستجابات من خلال البحث في قاعدة بيانات كبيرة من النصوص الموجودة مسبقاً واختيار المقاطع الأكثر صلة بناءً على الاستعلام المدخل. هذه النماذج لا تولد نصاً جديداً من الصفر ولكنها تجمع مقتطفات من قاعدة البيانات لتشكيل استجابة متماسكة.

من أهم مزايا النماذج القائمة على الاسترجاع قدرتها على تقديم معلومات دقيقة وحديثة. نظراً لاعتمادها على قاعدة بيانات من النصوص المنتقاة، يمكنها استخراج المعلومات ذات الصلة من مصادر موثوقة وتقديمها للمستخدم. هذا يجعلها مناسبة للتطبيقات التي تتطلب إجابات دقيقة وواقعية، مثل أنظمة الإجابة على الأسئلة أو قواعد المعرفة.

ومع ذلك، فإن النماذج القائمة على الاسترجاع لديها بعض القيود. فهي جيدة فقط بقدر جودة قاعدة البيانات التي تبحث فيها، لذا فإن جودة وتغطية قاعدة البيانات تؤثر مباشرة على أداء النموذج. بالإضافة إلى ذلك، قد تواجه هذه النماذج صعوبة في توليد استجابات متماسكة وطبيعية، حيث أنها مقيدة بالنص المتاح في قاعدة البيانات.

لا نتناول استخدام نماذج الاسترجاع الخالصة في هذا الكتاب.

النماذج التوليدية

النماذج التوليدية، من ناحية أخرى، تنشئ نصاً جديداً من الصفر بناءً على الأنماط والعلاقات التي تعلمتها أثناء التدريب. تستخدم هذه النماذج فهمها للغة لتوليد استجابات جديدة مصممة خصيصاً للنص التحفيزي المدخل.

نقطة القوة الرئيسية للنماذج التوليدية هي قدرتها على إنتاج نص إبداعي ومتناسك وذو صلة بالسياق. يمكنها المشاركة في محادثات مفتوحة، وتوليد قصص، وحتى كتابة الكود. هذا يجعلها مثالية للتطبيقات التي تتطلب تفاعلات أكثر انفتاحاً وديناميكية، مثل روبوتات الدردشة، وإنشاء المحتوى، ومساعدتي الكتابة الإبداعية.

ومع ذلك، يمكن للنماذج التوليدية أحياناً أن تنتج معلومات غير متسقة أو غير صحيحة من الناحية الواقعية، لأنها تعتمد على الأنماط المتعلمة أثناء التدريب بدلاً من قاعدة بيانات منتقاة من الحقائق. قد تكون أيضاً أكثر عرضة للتحيزات والهلوسات، مما يؤدي إلى توليد نص يبدو معقولاً ولكنه ليس بالضرورة صحيحاً.

من أمثلة نماذج اللغة التوليدية الكبيرة سلسلة GPT (GPT-3, OpenAI) و Claude و Anthropic.

النماذج الهجينة

تجمع العديد من نماذج اللغة الكبيرة المتاحة تجارياً بين نهجي الاسترجاع والتوليد في نموذج هجين. تستخدم هذه النماذج تقنيات الاسترجاع للعثور على المعلومات ذات الصلة من قاعدة البيانات ثم تستخدم التقنيات التوليدية لتوليف تلك المعلومات في استجابة متماسكة.

تهدف النماذج الهجينة إلى الجمع بين الدقة الواقعية للنماذج القائمة على الاسترجاع وقدرات توليد اللغة الطبيعية للنماذج التوليدية. يمكنها تقديم معلومات أكثر موثوقية وحدثة مع الحفاظ على القدرة على المشاركة في محادثات مفتوحة.

عند الاختيار بين النماذج القائمة على الاسترجاع والنماذج التوليدية، يجب عليك مراعاة المتطلبات المحددة لتطبيقك. إذا كان الهدف الأساسي هو تقديم معلومات دقيقة وواقعية، فقد يكون النموذج القائم على الاسترجاع هو الخيار الأفضل. إذا كان التطبيق يتطلب تفاعلات أكثر انفتاحاً وإبداعاً، فقد يكون النموذج التوليدي أكثر ملاءمة. تقدم النماذج الهجينة توازناً بين النهجين ويمكن أن تكون خياراً جيداً للتطبيقات التي تتطلب كلاً من الدقة الواقعية وتوليد اللغة الطبيعية.

في النهاية، يعتمد الاختيار بين النماذج القائمة على الاسترجاع والنماذج التوليدية على حالة الاستخدام المحددة والمفاضلات بين الدقة والإبداع والمرونة. من خلال فهم نقاط القوة والقيود لكل نهج، يمكنك اتخاذ قرارات مستنيرة.

التدريب على التعليمات

التدريب على التعليمات هو مجموعة فرعية من التدريب التحويلي الذي يستخدم النصوص التحفيزية والاستجابات المكتوبة بواسطة البشر لجعل نماذج اللغة الكبيرة أفضل في فهم التعليمات واتباعها. إليك بعض الأمثلة البسيطة لتوضيح كيف يجعل التدريب على التعليمات باستخدام النصوص التحفيزية والاستجابات المكتوبة بواسطة البشر نماذج اللغة الكبيرة أفضل في فهم التعليمات واتباعها:

مثال 1: دعم العملاء

التعليمات: "قدم ردًا لعميل يطلب استرداد الأموال."

النص التحفيزي المكتوب بواسطة البشر: "أود طلب استرداد الأموال لمشترياتي الأخيرة. المنتج لم يلب توقعاتي."

الاستجابة المكتوبة بواسطة البشر: "يؤسفني سماع أن المنتج لم يلب توقعاتك. لمعالجة طلب استرداد الأموال الخاص بك، يرجى تقديم رقم طلبك وسبب الإرجاع. سنراجع طلبك ونرد عليك في غضون 3-5 أيام عمل."

مثال 2: المساعدة التعليمية

التعليمات: "اشرح مفهوم التمثيل الضوئي لطفل عمره 10 سنوات."

نص بشري: "هل يمكنك شرح كيفية صنع النباتات لغذاءها؟"

إجابة بشرية: "بالتأكيد! تصنع النباتات غذاءها من خلال عملية تسمى التمثيل الضوئي. فهي تستخدم ضوء الشمس والماء وثنائي أكسيد الكربون لإنتاج الغذاء والأكسجين. تحتوي أوراق النبات على جزء خاص يسمى الكلوروفيل يساعدها على التقاط ضوء الشمس."

مثال 3: المساعد الشخصي

التعليمات: "جدولة اجتماع ليوم الثلاثاء القادم الساعة 3 مساءً."

نص بشري: "هل يمكنك تحديد موعد اجتماع يوم الثلاثاء الساعة 3 مساءً؟"

إجابة بشرية: “بالتأكيد، لقد جدولت لك اجتماعاً يوم الثلاثاء القادم الساعة 3 مساءً. هل هناك شيء آخر تحتاجه؟”

النتيجة هي نظام بيئي متنوع من نماذج اللغة الكبيرة بأحجام مختلفة وتخصصات متنوعة. توفر النماذج الأصغر في نطاق 1-7 مليار معامل قدرات لغوية عامة جيدة مع كونها أكثر كفاءة في التشغيل.

• 7B Mistral

• 8B 3 Llama

• 7B Gemma

تقدم النماذج متوسطة الحجم حوالي 30-70 مليار معامل قدرات أقوى في المنطق واتباع التعليمات.

• 70B 3 Llama

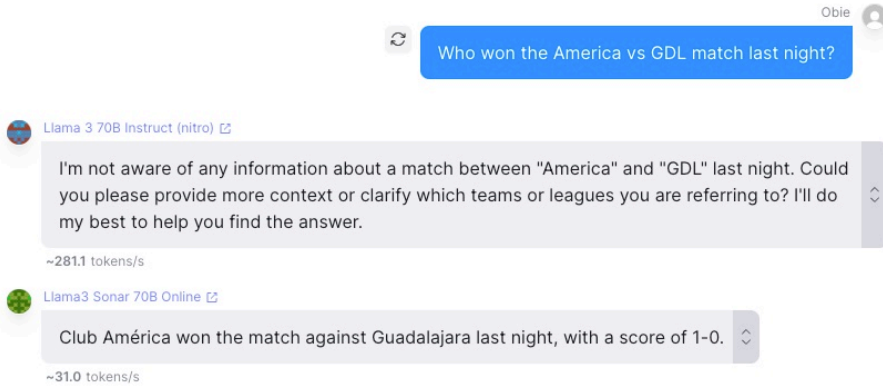
• 70B Qwen2

• 8x22B Mixtral

عند اختيار نموذج لغة كبير لدخله في تطبيق ما، يجب عليك الموازنة بين قدرات النموذج والعوامل العملية مثل التكلفة والتأخير وطول السياق وتصفية المحتوى. غالباً ما تكون النماذج الأصغر المدربة على التعليمات هي الخيار الأفضل للمهام اللغوية البسيطة، بينما قد تكون النماذج الأكبر ضرورية للتفكير المنطقي أو التحليل المعقد. كما تعد بيانات تدريب النموذج اعتباراً مهماً، حيث إنها تحدد تاريخ انقطاع معرفة النموذج.

بعض النماذج، مثل بعض نماذج Perplexity متصلة بمصادر معلومات في الوقت الفعلي، لذا فعلياً ليس لديها تاريخ انقطاع. عندما تسألها أسئلة، يمكنها أن تقرر بشكل مستقل إجراء عمليات بحث على الويب وجلب صفحات ويب عشوائية لتوليد إجابة.





الشكل 1.3. Llama3 مع وبدون اتصال بالإنترنت

في النهاية، لا يوجد نموذج لغة كبير واحد يناسب جميع الاحتياجات. فهم الاختلافات في حجم النموذج وهيكله وتدريبه أمر أساسي لاختيار النموذج المناسب لحالة استخدام معينة. التجريب مع نماذج مختلفة هو الطريقة العملية الوحيدة للكشف عن النماذج التي تقدم أفضل أداء للمهمة المطلوبة.

التجزئة إلى رموز: تقسيم النص إلى قطع

قبل أن يتمكن نموذج اللغة الكبير من معالجة النص، يجب تقسيم هذا النص إلى وحدات أصغر تسمى الرموز. يمكن أن تكون الرموز كلمات فردية، أو أجزاء من الكلمات، أو حتى أحرف فردية. تُعرف عملية تقسيم النص إلى رموز باسم التجزئة إلى رموز، وهي خطوة حاسمة في إعداد البيانات لنموذج اللغة.

The process of splitting text into tokens is known as tokenization, and it's a crucial step in preparing data for a language model.

الشكل 2.3. تحتوي هذه الجملة على 27 رمزاً

تستخدم نماذج اللغة الكبيرة المختلفة استراتيجيات تجزئة مختلفة، والتي يمكن أن يكون لها تأثير كبير على أداء النموذج وقدراته. تتضمن بعض أدوات التجزئة الشائعة المستخدمة في نماذج اللغة الكبيرة:

• GPT (ترميز أزواج البايت): تستخدم أدوات تجزئة GPT تقنية تسمى ترميز أزواج البايت (BPE)

لتقسيم النص إلى وحدات فرعية للكلمات. يقوم BPE بدمج أزواج البايت الأكثر تكراراً في مجموعة النصوص بشكل متكرر، مشكلاً مفردات من رموز الكلمات الفرعية. هذا يسمح لأداة التجزئة بالتعامل مع الكلمات النادرة والجديدة من خلال تقسيمها إلى أجزاء فرعية أكثر شيوعاً. تُستخدم أدوات تجزئة GPT في نماذج مثل 3GPT و 4-GPT.

- **لاما: (SentencePiece)** تستخدم مُجَزَّات لاما مكتبة SentencePiece، وهي مُجَزَّي ومُعِيد تجميع نصوص غير خاضع للإشراف. يتعامل SentencePiece مع النص المُدخل كسلسلة من محارف يونيكود ويتعلم معجم الوحدات الفرعية استناداً إلى مجموعة التدريب. يمكنه التعامل مع أي لغة يمكن ترميزها في يونيكود، مما يجعله مناسباً للنماذج متعددة اللغات. تُستخدم مُجَزَّات لاما في نماذج مثل لاما من ميتا وألباكا.

- **SentencePiece: (Unigram)** يمكن لِمُجَزَّات SentencePiece أيضاً استخدام خوارزمية مختلفة تسمى Unigram، والتي تعتمد على تقنية تنظيم الوحدات الفرعية. يحدد تجزئة Unigram المعجم الأمثل للوحدات الفرعية استناداً إلى نموذج لغوي أحادي، والذي يُعَيِّن احتمالات للوحدات الفرعية الفردية. يمكن لهذا النهج إنتاج وحدات فرعية ذات معنى دلالي أكثر مقارنة بـ BPE. يُستخدم SentencePiece مع Unigram في نماذج مثل T5 من جوجل وبييرت.

- **جيميني من جوجل (التجزئة متعددة الوسائط):** يستخدم جيميني من جوجل نظام تجزئة مصمم للتعامل مع أنواع مختلفة من البيانات، بما في ذلك النصوص والصور والصوت والفيديو والشفرة البرمجية. تتيح هذه القدرة متعددة الوسائط لجيميني معالجة ودمج أشكال مختلفة من المعلومات. من الجدير بالذكر أن جيميني 5.1 برو من جوجل لديه نافذة سياق يمكنها معالجة ملايين الرموز، وهي أكبر بكثير من النماذج السابقة. تتيح نافذة السياق الواسعة هذه للنموذج معالجة سياق أكبر، مما قد يؤدي إلى استجابات أكثر دقة. ومع ذلك، من المهم ملاحظة أن نظام تجزئة جيميني أقرب بكثير إلى رمز واحد لكل حرف مقارنة بالنماذج الأخرى. وهذا يعني أن التكلفة الفعلية لاستخدام نماذج جيميني قد تكون أعلى بكثير مما هو متوقع إذا كنت معتاداً على استخدام نماذج مثل GPT، حيث يعتمد تسعير جوجل على الأحرف وليس الرموز.

يؤثر اختيار المُجَزَّي على عدة جوانب في نموذج اللغة الكبير، بما في ذلك:

- **حجم المعجم:** يحدد المجزئ حجم معجم النموذج، وهو مجموعة الرموز الفريدة التي يتعرف عليها. يمكن للمعجم الأكبر والأكثر تفصيلاً أن يساعد النموذج في التعامل مع نطاق أوسع من الكلمات والعبارات وحتى أن يصبح متعدد الوسائط (قادر على فهم وإنتاج أكثر من مجرد نص)، ولكنه يزيد أيضاً من متطلبات الذاكرة والتعقيد الحسابي للنموذج.
- **التعامل مع الكلمات النادرة وغير المعروفة:** يمكن للمجزئات التي تستخدم وحدات فرعية، مثل BPE و SentencePiece، تقسيم الكلمات النادرة وغير المعروفة إلى قطع فرعية أكثر شيوعاً. هذا يسمح للنموذج بتخمين معنى الكلمات التي لم يرها من قبل، استناداً إلى الوحدات الفرعية التي تحتويها.
- **دعم تعدد اللغات:** المجزئات مثل SentencePiece التي يمكنها التعامل مع أي لغة قابلة للترميز باليونيكود، مناسبة للنماذج متعددة اللغات التي تحتاج إلى معالجة النصوص بلغات متعددة.

عند اختيار نموذج لغة كبير لتطبيق معين، من المهم النظر في المجزئ الذي يستخدمه ومدى توافقه مع احتياجات معالجة اللغة المحددة للمهمة المطلوبة. يمكن أن يكون للمجزئ تأثير كبير على قدرة النموذج على التعامل مع المصطلحات المتخصصة في مجال معين، والكلمات النادرة، والنصوص متعددة اللغات.

حجم السياق: ما مقدار المعلومات التي يمكن لنموذج اللغة استخدامها أثناء الاستدلال؟

عند مناقشة نماذج اللغة، يشير حجم السياق إلى كمية النص التي يمكن للنموذج النظر فيها عند معالجة أو إنشاء استجاباته. إنه في الأساس مقياس لمقدار المعلومات التي يمكن للنموذج "تذكرها" واستخدامها للتأثير على مخرجاته (معبراً عنها بالرموز). يمكن أن يكون لحجم السياق في نموذج اللغة تأثير كبير على قدراته وأنواع المهام التي يمكنه أدائها بفعالية.

ما هو حجم السياق؟

من الناحية التقنية، يتحدد حجم السياق بعدد الرموز (الكلمات أو أجزاء الكلمات) التي يمكن لنموذج اللغة معالجتها في تسلسل إدخال واحد. غالباً ما يُشار إلى هذا باسم "مدى الانتباه" أو "نافذة السياق" للنموذج. كلما كان حجم السياق أكبر، زاد مقدار النص الذي يمكن للنموذج النظر فيه في وقت واحد عند إنشاء استجابة أو أداء مهمة.

تمتلك نماذج اللغة المختلفة أحجام سياق متفاوتة، تتراوح من بضع مئات من الرموز إلى ملايين الرموز. للمقارنة، قد تحتوي الفقرة النموذجية من النص على حوالي 100-150 رمزاً، بينما قد يحتوي الكتاب بأكمله على عشرات أو مئات الآلاف من الرموز.

هناك حتى أعمال على طرق فعالة لتوسيع نطاق نماذج اللغة الكبيرة (LLMs) المعتمدة على المحولات للتعامل مع [مدخلات لا نهائية الطول](#) مع ذاكرة وحسابات محدودة.

لماذا يعد حجم السياق مهماً؟

يؤثر حجم السياق في نموذج اللغة تأثيراً كبيراً على قدرته على فهم وإنتاج نص متماسك ومرتبطة بالسياق. وفيما يلي بعض الأسباب الرئيسية لأهمية حجم السياق:

1. **فهم المحتوى الطويل:** النماذج ذات أحجام السياق الأكبر يمكنها فهم وتحليل النصوص الأطول بشكل أفضل، مثل المقالات والتقارير وحتى الكتب بأكملها. وهذا أمر حاسم لمهام مثل تلخيص المستندات والإجابة عن الأسئلة وتحليل المحتوى.

2. **الحفاظ على التماسك:** تتيح نافذة السياق الأكبر للنموذج الحفاظ على التماسك والاتساق عبر مقاطع أطول من المخرجات. وهذا مهم لمهام مثل إنشاء القصص وأنظمة الحوار وإنشاء المحتوى، حيث يعد الحفاظ على سرد أو موضوع متنسق أمراً ضرورياً. كما أنه مهم للغاية عند استخدام نماذج اللغة الكبيرة لإنشاء أو تحويل البيانات المنظمة.

3. **التقاط الاعتمادات بعيدة المدى:** تتطلب بعض مهام اللغة فهم العلاقات بين الكلمات أو العبارات المتباعدة في النص. النماذج ذات أحجام السياق الأكبر مجهزة بشكل أفضل لالتقاط هذه الاعتمادات بعيدة المدى، والتي يمكن أن تكون مهمة لمهام مثل تحليل المشاعر والترجمة وفهم اللغة.

4. **التعامل مع التعليمات المعقدة:** في التطبيقات التي تُستخدم فيها نماذج اللغة لاتباع تعليمات معقدة متعددة الخطوات، يسمح حجم السياق الأكبر للنموذج بمراعاة مجموعة التعليمات بأكملها عند إنشاء استجابة، بدلاً من مجرد الكلمات القليلة الأخيرة.

أمثلة على نماذج اللغة بأحجام سياق مختلفة

فيما يلي بعض الأمثلة على نماذج اللغة بأحجام سياق مختلفة:

- Turbo: GPT-5.3 OpenAI 4,095 رمز
- Instruct: 7B Mistral 32,768 رمز
- Claude Anthropic v1: 100,000 رمز
- Turbo: GPT-4 OpenAI 128,000 رمز
- Claude Anthropic v2: 200,000 رمز
- Pro Gemini Google 5.1 : 8M.2 رمز

كما ترون، هناك مجموعة واسعة من أحجام السياق بين هذه النماذج، من حوالي 4,000 رمز لنموذج OpenAI Turbo GPT-5.3 إلى 200,000 رمز لنموذج Claude Anthropic v2. بعض النماذج، مثل Google PaLM 2 و GPT-4 OpenAI، تقدم إصدارات مختلفة بأحجام سياق أكبر (مثل إصدارات، "32k" والتي يمكنها التعامل مع تسلسلات إدخال أطول. وفي الوقت الحالي (أبريل 2024) يتبناها Google Pro Gemini بما يقرب من 3 ملايين رمز!

من الجدير بالذكر أن حجم السياق يمكن أن يختلف حسب التطبيق والإصدار المحدد لنموذج معين. على سبيل المثال، نموذج GPT-4 OpenAI الأصلي لديه حجم سياق يبلغ 8,191 رمزاً، في حين أن إصدارات GPT-4 اللاحقة مثل Turbo و 4o لديها حجم سياق أكبر بكثير يبلغ 128,000 رمز.

قام Altman Sam بمقارنة قيود السياق الحالية بالكيلوبايت من الذاكرة العاملة التي كان على مبرمجي الكمبيوتر الشخصي التعامل معها في الثمانينيات، وقال إنه في المستقبل القريب سنتمكن من تضمين "جميع بياناتك الشخصية" في سياق نموذج اللغة الكبير.

اختيار حجم السياق المناسب

عند اختيار نموذج لغة لتطبيق معين، من المهم مراعاة متطلبات حجم السياق للهمة المطلوبة. بالنسبة للمهام التي تتضمن نصوصاً قصيرة ومنفصلة، مثل تحليل المشاعر أو الإجابة عن الأسئلة البسيطة، قد يكون حجم

السياق الأصغر كافياً. ومع ذلك، بالنسبة للمهام التي تتطلب فهم وإنتاج نصوص أطول وأكثر تعقيداً، سيكون حجم السياق الأكبر ضرورياً على الأرجح.

من الجدير بالذكر أن أحجام السياق الأكبر غالباً ما تأتي مع زيادة في التكاليف الحسابية وأوقات المعالجة الأبطأ، حيث يحتاج النموذج إلى النظر في المزيد من المعلومات عند إنشاء استجابة. وعلى هذا النحو، يجب عليك تحقيق التوازن بين حجم السياق والأداء عند اختيار نموذج اللغة لتطبيقك.

لماذا لا نختار ببساطة النموذج ذا حجم السياق الأكبر ونملأه بأكبر قدر ممكن من المعلومات؟ حسناً، إلى جانب عوامل الأداء، الاعتبار الرئيسي الآخر هو التكلفة. في مارس 2024، دورة إدخال واستجابة واحدة باستخدام Google Gemini Pro 5.1 مع سياق كامل ستكلفك ما يقرب من 8 دولارات أمريكية. إذا كان لديك حالة استخدام تبرر هذه النفقات، فهنيئاً لك! ولكن بالنسبة لمعظم التطبيقات، فإنها مكلفة للغاية بمقدار عدة مرات.

العثور على الإبر في أكوام القش

لطالما كان مفهوم العثور على إبر في كومة قش استعارة لتحديات الاسترجاع في مجموعات البيانات الكبيرة. في مجال نماذج اللغة الكبيرة، نقوم بتعديل هذا التشبيه قليلاً. تخيل أننا لا نبحث فقط عن حقيقة واحدة مدفونة داخل نص واسع (مثل مجموعة كاملة من مقالات Graham Paul)، بل عن حقائق متعددة متناثرة في كل مكان. هذا السيناريو أشبه بالبحث عن عدة إبر في حقل متراحي الأطراف، وليس مجرد كومة قش واحدة. وهنا المفارقة: لا نحتاج فقط إلى تحديد مواقع هذه الإبر، بل علينا أيضاً نسجها في خيط متماسك.

عندما تُكلف نماذج اللغة الكبيرة باسترجاع والتفكير في حقائق متعددة مضمنة في سياقات طويلة، تواجه تحدياً مزدوجاً. أولاً، هناك مشكلة دقة الاسترجاع البسيطة - التي تنخفض طبعاً مع زيادة عدد الحقائق. هذا متوقع؛ فبعد كل شيء، يُرهق تتبع تفاصيل متعددة عبر نص متراحي حتى أكثر النماذج تطوراً.

ثانياً، وربما الأهم من ذلك، هو تحدي التفكير بهذه الحقائق. استخراج الحقائق شيء، وتوليفها في سرد متماسك أو إجابة شيء آخر تماماً. هنا يأتي الاختبار الحقيقي. يميل أداء نماذج اللغة الكبيرة في مهام

التفكير إلى التدهور أكثر من مهام الاسترجاع البسيطة. هذا التدهور لا يتعلق فقط بالحجم؛ بل يتعلق بالرقصة المعقدة بين السياق والصلة والاستدلال.

لماذا يحدث هذا؟ حسناً، فكر في ديناميكيات الذاكرة والانتباه في الإدراك البشري، والتي تنعكس إلى حد ما في نماذج اللغة الكبيرة. عند معالجة كميات كبيرة من المعلومات، يمكن لنماذج اللغة الكبيرة، مثل البشر، أن تفقد تتبع التفاصيل السابقة أثناء استيعابها لمعلومات جديدة. هذا صحيح بشكل خاص في النماذج التي لم تُصمم صراحةً لإعطاء الأولوية أو إعادة زيارة المقاطع السابقة من النص تلقائياً.

علاوة على ذلك، فإن قدرة نموذج اللغة الكبير على نسج هذه الحقائق المسترجعة في استجابة متماسكة تشبه بناء السرد. وهذا يتطلب ليس فقط استرجاع المعلومات بل فهماً عميقاً ووضعاً سياقياً، وهو ما يظل تحدياً صعباً للذكاء الاصطناعي الحالي.

إذاً، ماذا يعني هذا لنا كمطورين ومدمجين لهذه التقنيات؟ نحتاج إلى أن نكون على دراية تامة بهذه القيود عند تصميم أنظمة تعتمد على نماذج اللغة الكبيرة للتعامل مع المهام المعقدة والطويلة. فهم أن الأداء قد يتدهور في ظل ظروف معينة يساعدنا على وضع توقعات واقعية وهندسة آليات احتياطية أفضل أو استراتيجيات تكميلية.

الوسائط: ما وراء النص

في حين أن غالبية نماذج اللغة اليوم تركز على معالجة وتوليد النص، هناك اتجاه متزايد نحو النماذج متعددة الوسائط التي يمكنها بشكل طبيعي إدخال وإخراج أنواع متعددة من البيانات، مثل الصور والصوت والفيديو. تفتح هذه النماذج متعددة الوسائط إمكانيات جديدة للتطبيقات المدعومة بالذكاء الاصطناعي التي يمكنها فهم وتوليد المحتوى عبر وسائط مختلفة.

ما هي الوسائط؟

في سياق نماذج اللغة، تشير الوسائط إلى الأنواع المختلفة من البيانات التي يمكن للنموذج معالجتها وتوليدها. الوسيط الأكثر شيوعاً هو النص، الذي يشمل اللغة المكتوبة بأشكال مختلفة مثل الكتب والمقالات والمواقع الإلكترونية ومنشورات وسائل التواصل الاجتماعي. ومع ذلك، هناك العديد من الوسائط الأخرى التي يتم دمجها بشكل متزايد في نماذج اللغة:

- الصور: البيانات المرئية مثل الصور الفوتوغرافية والرسوم التوضيحية والمخططات.

- الصوت: البيانات الصوتية مثل الكلام والموسيقى والأصوات البيئية.
- الفيديو: البيانات المرئية المتحركة، غالباً ما تكون مصحوبة بالصوت، مثل مقاطع الفيديو والأفلام.

تقدم كل وسيط تحديات وفرص فريدة لنماذج اللغة. على سبيل المثال، تتطلب الصور من النموذج فهم المفاهيم والعلاقات المرئية، بينما يتطلب الصوت من النموذج معالجة وتوليد الكلام والأصوات الأخرى.

نماذج اللغة متعددة الوسائط

نماذج اللغة متعددة الوسائط مصممة للتعامل مع وسائط متعددة داخل نموذج واحد. تحتوي هذه النماذج عادةً على مكونات أو طبقات متخصصة يمكنها فهم المدخلات وتوليد البيانات المخرجة في وسائط مختلفة. بعض الأمثلة البارزة على نماذج اللغة متعددة الوسائط تشمل:

- **نموذج GPT-4o من OpenAI:** هو نموذج لغة كبير يفهم ويعالج بشكل طبيعي الصوت المنطوق بالإضافة إلى النص. يتيح هذه القدرة لـ GPT-4o تنفيذ مهام مثل نسخ اللغة المنطوقة، وتوليد النص من المدخلات الصوتية، وتقديم الردود بناءً على الاستفسارات المنطوقة.
- **نموذج GPT-4 من OpenAI مع المدخلات المرئية:** GPT-4 هو نموذج لغة كبير يمكنه معالجة كل من النص والصور. عند إعطائه صورة كمدخل، يمكن لـ GPT-4 تحليل محتويات الصورة وتوليد نص يصف أو يستجيب للمعلومات المرئية.
- **نموذج Gemini من Google:** هو نموذج متعدد الوسائط يمكنه التعامل مع النص والصور والفيديو. يستخدم بنية موحدة تسمح بالفهم والتوليد عبر الوسائط المتعددة، مما يتيح مهام مثل وصف الصور، وتلخيص الفيديو، والإجابة على الأسئلة المرئية.
- **DALL-E و Diffusion: Stable:** على الرغم من أنها ليست نماذج لغوية بالمعنى التقليدي، إلا أن هذه النماذج توضح قوة الذكاء الاصطناعي متعدد الوسائط من خلال توليد الصور من الأوصاف النصية. إنها تعرض إمكانيات النماذج التي يمكنها الترجمة بين الوسائط المختلفة.

فوائد وتطبيقات النماذج متعددة الوسائط

تقدم النماذج اللغوية متعددة الوسائط العديد من الفوائد وتمكن مجموعة واسعة من التطبيقات، بما في ذلك:

- فهم معزز: من خلال معالجة المعلومات من وسائط متعددة، يمكن لهذه النماذج اكتساب فهم أكثر شمولاً للعالم، مشابه للطريقة التي يتعلم بها البشر من المدخلات الحسية المختلفة.
- التوليد عبر الوسائط: يمكن للنماذج متعددة الوسائط توليد محتوى في وسيط واحد بناءً على مدخلات من وسيط آخر، مثل إنشاء صورة من وصف نصي أو توليد ملخص فيديو من مقال مكتوب.
- إمكانية الوصول: يمكن للنماذج متعددة الوسائط جعل المعلومات أكثر سهولة في الوصول من خلال الترجمة بين الوسائط، مثل توليد أوصاف نصية للصور للمستخدمين ضعاف البصر أو إنشاء نسخ صوتية من المحتوى المكتوب.
- التطبيقات الإبداعية: يمكن استخدام النماذج متعددة الوسائط للمهام الإبداعية مثل توليد الفن والموسيقى أو مقاطع الفيديو بناءً على التوجيهات النصية، مما يفتح إمكانيات جديدة للفنانين ومنشئي المحتوى.

مع استقرار تطور النماذج اللغوية متعددة الوسائط، من المرجح أن تلعب دوراً متزايد الأهمية في تطوير التطبيقات المدعومة بالذكاء الاصطناعي التي يمكنها فهم وتوليد المحتوى عبر وسائط متعددة. سيؤدي هذا إلى تمكين تفاعلات أكثر طبيعية وبدئية بين البشر وأنظمة الذكاء الاصطناعي، بالإضافة إلى فتح إمكانيات جديدة للتعبير الإبداعي ونشر المعرفة.

أنظمة مزودي الخدمات

عندما يتعلق الأمر بدمج النماذج اللغوية الكبيرة (LLMs) في التطبيقات، لديك مجموعة متزايدة من الخيارات للاختيار من بينها. يقدم كل مزود رئيسي للنماذج اللغوية الكبيرة، مثل Anthropic و OpenAI و Google و Cohere نظامه الخاص من النماذج وأجهزة برجة التطبيقات والأدوات. يتضمن اختيار المزود المناسب النظر في عوامل مختلفة، بما في ذلك التسعير والأداء وتصفية المحتوى وخصوصية البيانات وخيارات التخصيص.

OpenAI

OpenAI هي واحدة من أشهر مزودي النماذج اللغوية الكبيرة، حيث تستخدم سلسلة GPT الخاصة بها، GPT-3، GPT-4 على نطاق واسع في تطبيقات مختلفة. تقدم OpenAI واجهة برجة تطبيقات سهلة الاستخدام تتيح لك دمج نماذجها في التطبيقات بسهولة. كما تقدم مجموعة من النماذج بقدرات وأسعار مختلفة، من نموذج Ada للمبتدئين إلى نموذج Davinci القوي.

يتضمن نظام OpenAI أيضاً أدوات مثل OpenAI Playground، الذي يتيح لك تجربة التوجيهات وضبط النماذج بدقة لحالات استخدام محددة. كما يقدمون خيارات تصفية المحتوى للمساعدة في منع توليد المحتوى غير المناسب أو الضار.

عند استخدام نماذج OpenAI مباشرة، أعتمد على مكتبة Rudall Alex [ruby-openai](#).

Anthropic

Anthropic هي لاعب رئيسي آخر في مجال النماذج اللغوية الكبيرة، حيث تكتسب نماذج Claude الخاصة بها شعبية للأداء القوي والاعتبارات الأخلاقية. تركز Anthropic على تطوير أنظمة ذكاء اصطناعي آمنة ومسؤولة، مع تركيز قوي على تصفية المحتوى وتجنب المخرجات الضارة.

يتضمن نظام Anthropic واجهة برمجة تطبيقات، Claude التي تتيح لك دمج النموذج في تطبيقاتهم، بالإضافة إلى أدوات هندسة التوجيهات والضبط الدقيق. كما يقدمون نموذج Claude Instant، الذي يتضمن قدرات البحث على الويب للحصول على ردود أكثر حداثة ودقة.

عند استخدام نماذج Anthropic مباشرة، أعتمد على مكتبة Rudall Alex [anthropic](#).

Google

طورت Google العديد من النماذج اللغوية الكبيرة القوية، بما في ذلك Gemini وBERT وT5 وPaLM. تشتهر هذه النماذج بأدائها القوي في مجموعة واسعة من مهام معالجة اللغة الطبيعية. يتضمن نظام Google مكتبات TensorFlow وKeras، التي توفر أدوات وأطر عمل لبناء وتدريب نماذج التعلم الآلي.

تقدم Google أيضاً منصة AI Cloud، التي تتيح لك نشر وتوسيع نطاق نماذجهم في السحابة بسهولة. كما يوفر مجموعة من النماذج المدربة مسبقاً وواجهات برمجة التطبيقات لمهام مثل تحليل المشاعر والتعرف على الكيانات والترجمة.

Meta

Meta، المعروفة سابقاً باسم Facebook، تستثمر بعمق في تطوير النماذج اللغوية الكبيرة، ويتجلى ذلك من خلال إصدارها لنماذج مثل LLaMA وOPT. تتميز هذه النماذج بأدائها القوي في مهام لغوية متنوعة وتتوفر بشكل كبير من خلال قنوات مفتوحة المصدر، مما يدعم التزام Meta بالبحث والتعاون المجتمعي.

يتحور نظام Meta بشكل أساسي حول PyTorch، وهي مكتبة تعلم آلي مفتوحة المصدر مفضلة لقدراتها الحاسوبية الديناميكية ومرونتها، مما يسهل البحث والتطوير المبتكر في مجال الذكاء الاصطناعي. بالإضافة إلى عروضها التقنية، تضع Meta تركيزاً قوياً على تطوير الذكاء الاصطناعي الأخلاقي. فهي تطبق تصفية محتوى قوية وتركز على تقليل التحيزات، بما يتماشى مع أهدافها الأوسع نطاقاً المتمثلة في السلامة والمسؤولية في تطبيقات الذكاء الاصطناعي.

Cohere

تعد Cohere من الشركات الحديثة في مجال نماذج اللغة الكبيرة، حيث تركز على جعل هذه النماذج أكثر سهولة في الوصول والاستخدام مقارنة بالمنافسين. تتضمن منظومتها واجهة Cohere للبرمجة التي توفر الوصول إلى مجموعة من النماذج المدربة مسبقاً لمهام مثل توليد النصوص والتصنيف والتلخيص. كما تقدم Cohere أدوات لهندسة التوجيهات والضبط الدقيق وتصفية المحتوى. وهي تؤكد على خصوصية البيانات والأمان، مع ميزات مثل تخزين البيانات المشفرة وضوابط الوصول.

Ollama

Ollama هي منصة ذاتية الاستضافة تتيح للمستخدمين إدارة ونشر مختلف نماذج اللغة الكبيرة محلياً على أجهزتهم، مما يمنحهم تحكماً كاملاً في نماذج الذكاء الاصطناعي الخاصة بهم دون الاعتماد على خدمات السحابة الخارجية. هذا الإصدار مثالي لأولئك الذين يضعون خصوصية البيانات في المقام الأول ويرغبون في إدارة عمليات الذكاء الاصطناعي الخاصة بهم داخلياً.

تدعم المنصة مجموعة من النماذج، بما في ذلك إصدارات Llama و Phi و Gemma و Mistral، والتي تختلف في الحجم ومتطلبات الحوسبة. تجعل Ollama من السهل تنزيل وتشغيل هذه النماذج مباشرة من سطر الأوامر باستخدام أوامر بسيطة مثل `ollama run <model name>`، وهي مصممة للعمل عبر أنظمة تشغيل مختلفة بما في ذلك macOS و Linux و Windows.

بالنسبة للمطورين الذين يتطلعون إلى دمج النماذج مفتوحة المصدر في تطبيقاتهم دون استخدام واجهة برمجة تطبيقات بعيدة، توفر Ollama واجهة سطر أوامر لإدارة دورة حياة النموذج مشابهة لأدوات إدارة الحاويات. كما تدعم التكوينات والتوجيهات المخصصة، مما يتيح درجة عالية من التخصيص لتكييف النماذج مع احتياجات أو حالات استخدام محددة.

تناسب Ollama بشكل خاص المستخدمين والمطورين ذوي الخبرة التقنية نظراً لواجهة سطر الأوامر والمرونة التي تقدمها في إدارة ونشر نماذج الذكاء الاصطناعي. هذا يجعلها أداة قوية للشركات والأفراد الذين يحتاجون إلى قدرات ذكاء اصطناعي قوية دون المساومة على الأمان والتحكم.

منصات متعددة النماذج

بالإضافة إلى ذلك، هناك مزودون يستضيفون مجموعة متنوعة من النماذج مفتوحة المصدر، مثل Together.ai و Groq. تقدم هذه المنصات المرونة والتخصيص، مما يتيح لك تشغيل النماذج مفتوحة المصدر وفي بعض الحالات حتى ضبطها بدقة وفقاً لاحتياجاتك المحددة. على سبيل المثال، توفر Together.ai الوصول إلى مجموعة من نماذج اللغة الكبيرة مفتوحة المصدر، مما يمكن المستخدمين من تجربة نماذج وتكوينات مختلفة. تركز Groq على تقديم إكمال فائق الأداء يبدو في وقت كتابة هذا الكتاب وكأنه سحري تقريباً

اختيار مزود نماذج اللغة الكبيرة

عند اختيار مزود نماذج اللغة الكبيرة، يجب عليك مراعاة عوامل مثل:

- التسعير: يقدم المزودون المختلفون نماذج تسعير مختلفة، تتراوح بين الدفع حسب الاستخدام وخطط الاشتراك. من المهم مراعاة الاستخدام المتوقع والميزانية عند اختيار المزود.
- الأداء: يمكن أن يختلف أداء نماذج اللغة الكبيرة بشكل كبير بين المزودين، لذلك من المهم قياس واختبار النماذج في حالات الاستخدام المحددة قبل اتخاذ القرار.
- تصفية المحتوى: اعتماداً على التطبيق، قد تكون تصفية المحتوى اعتباراً حاسماً. يقدم بعض المزودين خيارات تصفية محتوى أكثر قوة من غيرهم.
- خصوصية البيانات: إذا كان التطبيق يتعامل مع بيانات مستخدم حساسة، فمن المهم اختيار مزود يتمتع بممارسات قوية لخصوصية البيانات والأمان.
- التخصيص: يقدم بعض المزودين مرونة أكبر من حيث الضبط الدقيق وتخصيص النماذج لحالات استخدام محددة.

في النهاية، يعتمد اختيار مزود نماذج اللغة الكبيرة على المتطلبات والقيود المحددة للتطبيق. من خلال التقييم الدقيق للخيارات ومراعاة عوامل مثل التسعير والأداء وخصوصية البيانات، يمكنك اختيار المزود الذي يلبي احتياجاتك بشكل أفضل.

من الجدير بالذكر أيضاً أن مشهد نماذج اللغة الكبيرة يتطور باستمرار، مع ظهور مزودين ونماذج جديدة بانتظام. يجب عليك البقاء على اطلاع بأحدث التطورات والانفتاح على استكشاف خيارات جديدة عندما تصبح متاحة.

OpenRouter

طوال هذا الكتاب سأعتمد حصراً على [OpenRouter](#) كمزود واجهة برمجة التطبيقات المفضل لدي. السبب بسيط: إنه متجر شامل لجميع النماذج التجارية ومفتوحة المصدر الأكثر شعبية. إذا كنت متحمساً للبدء في برمجة الذكاء الاصطناعي، فإن أحد أفضل الأماكن للبدء هو مكتبة [Ruby OpenRouter](#) الخاصة بي [Library Ruby OpenRouter](#).

التفكير في الأداء

عند دمج النماذج اللغوية في التطبيقات، يعد الأداء اعتباراً حاسماً. يمكن قياس أداء النموذج اللغوي من حيث زمن الاستجابة (الوقت المستغرق لإنشاء استجابة) والإنتاجية (عدد الطلبات التي يمكنه معالجتها في وحدة الزمن).

يعد زمن الرمز الأول (TTFT)) مقياساً أساسياً آخر للأداء، وهو ذو صلة خاصة ببروبات الدردشة والتطبيقات التي تتطلب استجابات تفاعلية في الوقت الفعلي. يقيس TTFT زمن الاستجابة من لحظة استلام طلب المستخدم إلى لحظة إنشاء أول كلمة (أو رمز) من الاستجابة. هذا المقياس ضروري للحفاظ على تجربة مستخدم سلسة وجذابة، حيث يمكن أن تؤدي الاستجابات المتأخرة إلى إحباط المستخدم وانصرافه.

يمكن أن يكون لمقاييس الأداء هذه تأثير كبير على تجربة المستخدم وقابلية التطبيق للتوسع.

هناك عدة عوامل يمكن أن تؤثر على أداء النموذج اللغوي، منها:

عدد المعاملات: النماذج الأكبر ذات المعاملات الأكثر تتطلب عموماً موارد حسابية أكثر ويمكن أن يكون لديها زمن استجابة أعلى وإنتاجية أقل مقارنة بالنماذج الأصغر.

العتاد: يمكن أن يختلف أداء النموذج اللغوي بشكل كبير بناءً على العتاد الذي يعمل عليه. يقدم مزودو الخدمات السحابية خوادم GPU وTPU مُحسنة لأعباء عمل التعلم الآلي، والتي يمكن أن تسرع استدلال النموذج بشكل كبير.

أحد الأشياء الجميلة في OpenRouter هو أنه للعديد من النماذج التي يقدمها، تحصل على خيار من مزودي الخدمات السحابية مع مجموعة متنوعة من خصائص الأداء والتكاليف.



التكريم: يمكن استخدام تقنيات التكريم لتقليل البصمة الذاكرة والمتطلبات الحسابية للنموذج من خلال تمثيل الأوزان والتفعيلات بأنواع بيانات ذات دقة أقل. يمكن أن يحسن هذا الأداء دون التضحية بشكل كبير بالجودة. كطور تطبيقات، من غير المحتمل أن تشارك في تدريب نماذجك الخاصة بمستويات تكريم مختلفة، ولكن من الجيد على الأقل أن تكون على دراية بالمصطلحات.

المعالجة الدفعية: معالجة طلبات متعددة في نفس الوقت في دفعات يمكن أن تحسن الإنتاجية من خلال توزيع تكلفة تحميل النموذج ونقل البيانات.

التخزين المؤقت: تخزين نتائج المطالبات أو تسلسلات الإدخال المستخدمة بشكل متكرر في الذاكرة المؤقتة يمكن أن يقلل من عدد طلبات الاستدلال ويحسن الأداء العام.

عند اختيار نموذج لغوي لتطبيق إنتاجي، من المهم قياس أدائه على أعباء عمل وتكوينات عتاد تمثيلية. يمكن أن يساعد هذا في تحديد نقاط الاختناق المحتملة وضمان أن النموذج يمكنه تحقيق أهداف الأداء المطلوبة.

من الجدير أيضاً النظر في المفاضلات بين أداء النموذج والعوامل الأخرى مثل التكلفة والمرونة وسهولة الدمج. على سبيل المثال، قد يكون استخدام نموذج أصغر وأقل تكلفة مع زمن استجابة أقل مفضلاً للتطبيقات التي تتطلب استجابات في الوقت الفعلي، بينما قد يكون النموذج الأكبر والأكثر قوة أكثر ملاءمة لمعالجة الدفعات أو مهام التفكير المعقدة.

تجربة نماذج LLM المختلفة

اختيار نموذج LLM نادراً ما يكون قراراً دائماً. مع إصدار نماذج جديدة ومحسنة بانتظام، من الجيد بناء التطبيقات بطريقة نمطية تسمح بتبديل نماذج لغوية مختلفة مع مرور الوقت. يمكن غالباً إعادة استخدام المطالبات ومجموعات البيانات عبر النماذج مع تغييرات طفيفة. هذا يتيح لك الاستفادة من أحدث التطورات في النمذجة اللغوية دون الحاجة إلى إعادة تصميم تطبيقاتك بالكامل.

القدرة على التبديل بسهولة بين مجموعة واسعة من خيارات النماذج هي سبب آخر يجعلني أحب

OpenRouter.



عند الترقية إلى نموذج لغوي جديد، من المهم اختبار والتحقق من أدائه وجودة مخرجاته بشكل شامل للتأكد من أنه يلبي متطلبات التطبيق. قد يتضمن هذا إعادة التدريب أو الضبط الدقيق للنموذج على بيانات خاصة بالمجال، بالإضافة إلى تحديث أي مكونات تابعة تعتمد على مخرجات النموذج. من خلال تصميم التطبيقات مع مراعاة الأداء والنمطية، يمكنك إنشاء أنظمة قابلة للتوسع وفعالة ومستقبلية يمكنها التكيف مع المشهد سريع التطور لتكنولوجيا النمذجة اللغوية.

أنظمة الذكاء الاصطناعي المركبة

قبل إنهاء مقدمتنا، من الجدير بالذكر أنه قبل عام 2023 وانفجار الاهتمام بالذكاء الاصطناعي التوليدي الذي أشعله ChatGPT، كانت مناهج الذكاء الاصطناعي التقليدية تعتمد عادةً على دمج نماذج فردية مغلقة. في المقابل، تستفيد أنظمة الذكاء الاصطناعي المركبة من خطوط أنابيب معقدة من المكونات المترابطة التي تعمل معاً لتحقيق السلوك الذكي.

في جوهرها، تتكون أنظمة الذكاء الاصطناعي المركبة من وحدات متعددة، كل منها مصمم لأداء مهام أو وظائف محددة. يمكن أن تشمل هذه الوحدات المولدات والمسترجعات والمصنفات والمصنفات وغيرها من المكونات المتخصصة المختلفة. من خلال تقسيم النظام الكلي إلى وحدات أصغر ومركزة، يمكن للمطورين إنشاء بنى ذكاء اصطناعي أكثر مرونة وقابلية للتوسع وقابلية للصيانة.

من المزايا الرئيسية لأنظمة الذكاء الاصطناعي المركبة قدرتها على الجمع بين نقاط القوة لتقنيات ونماذج الذكاء الاصطناعي المختلفة. على سبيل المثال، قد يستخدم النظام نموذج لغة كبير (LLM) لفهم وتوليد اللغة الطبيعية، بينما يوظف نموذجاً منفصلاً لاسترجاع المعلومات أو صنع القرار القائم على القواعد. يتيح هذا النهج النمطي اختيار أفضل الأدوات والتقنيات لكل مهمة محددة، بدلاً من الاعتماد على حل واحد يناسب الجميع.

ومع ذلك، فإن بناء أنظمة الذكاء الاصطناعي المركبة يقدم أيضاً تحديات فريدة. على وجه الخصوص، يتطلب ضمان التماسك والاتساق الكلي لسلوك النظام آليات قوية للاختبار والمراقبة والحوكمة.

إن ظهور نماذج اللغة الكبيرة القوية مثل GPT-4 يتيح لنا تجربة أنظمة الذكاء الاصطناعي المركبة بسهولة أكبر من أي وقت مضى، لأن هذه النماذج المتقدمة قادرة على التعامل مع أدوار متعددة داخل النظام المركب، مثل التصنيف والترتيب والتوليد، بالإضافة إلى قدراتها على فهم اللغة الطبيعية. تتيح هذه المرونة للمطورين إنشاء نماذج أولية والتكرار السريع لهياكل الذكاء الاصطناعي المركبة، مما يفتح إمكانيات جديدة لتطوير التطبيقات الذكية.



أنماط نشر أنظمة الذكاء الاصطناعي المركبة

يمكن نشر أنظمة الذكاء الاصطناعي المركبة باستخدام أنماط مختلفة، كل منها مصمم لتلبية متطلبات وحالات استخدام محددة. دعونا نستكشف أربعة أنماط شائعة للنشر: السؤال والجواب، وحل المشكلات متعدد العوامل/العالمي، والذكاء الاصطناعي المحادثي، والمساعدين المشاركين.

السؤال والجواب

تركز أنظمة السؤال والجواب على تقديم استرجاع للمعلومات معزز بقدرات الفهم لنماذج الذكاء الاصطناعي للعمل كأكثر من مجرد محرك بحث. من خلال الجمع بين نماذج اللغة القوية مع مصادر المعرفة الخارجية باستخدام **التوليد المعزز بالاسترجاع (RAG)**، تتجنب أنظمة السؤال والجواب الهلوسة وتقدم إجابات دقيقة ومرتبطة بالسياق لاستفسارات المستخدمين.

تتضمن المكونات الرئيسية لنظام السؤال والجواب المعتمد على LLM ما يلي:

- فهم وإعادة صياغة الاستعلام: تحليل استفسارات المستخدم وإعادة صياغتها لتناسب بشكل أفضل مع مصادر المعرفة الأساسية.
- استرجاع المعرفة: استرجاع المعلومات ذات الصلة من مصادر البيانات المنظمة أو غير المنظمة بناءً على الاستعلام المعاد صياغته.
- توليد الاستجابة: توليد ردود متماسكة ومفيدة من خلال دمج المعرفة المسترجعة مع قدرات التوليد لنموذج اللغة.

تعد الأنظمة الفرعية للتوليد المعزز بالاسترجاع (RAG) مهمة بشكل خاص في مجالات السؤال والجواب حيث يكون توفير معلومات دقيقة وحديثة أمرًا حاسمًا، مثل دعم العملاء، وإدارة المعرفة، أو التطبيقات التعليمية.

حلال المشكلات متعدد العوامل/العالمي

تتكون الأنظمة متعددة العوامل، المعروفة أيضاً باسم العالمية ، من عدة عوامل مستقلة تعمل معاً لحل المشكلات المعقدة. لكل عامل دور محدد ومجموعة من المهارات والوصول إلى الأدوات أو مصادر المعلومات ذات الصلة. من خلال التعاون وتبادل المعلومات، يمكن لهذه العوامل معالجة المهام التي يصعب أو يستحيل على عامل واحد التعامل معها بمفرده.

تتضمن المبادئ الرئيسية لحلال المشكلات متعدد العوامل ما يلي:

- التخصص: يركز كل عامل على جانب محدد من المشكلة، مستفيداً من قدراته ومعرفته الفريدة.
- التعاون: يتواصل العامل وينسقون أعمالهم لتحقيق هدف مشترك، غالباً من خلال تمرير الرسائل أو الذاكرة المشتركة.
- القابلية للتكيف: يمكن للنظام التكيف مع الظروف أو المتطلبات المتغيرة من خلال تعديل أدوار وسلوكيات العوامل الفردية.

تعد الأنظمة متعددة العوامل مناسبة للتطبيقات التي تتطلب حل المشكلات الموزعة، مثل تحسين سلسلة التوريد ، وإدارة حركة المرور ، أو تخطيط الاستجابة للطوارئ .

الذكاء الاصطناعي المحادثي

تمكّن أنظمة الذكاء الاصطناعي المحادثي التفاعلات باللغة الطبيعية بين المستخدمين والعوامل الذكية. تجمع هذه الأنظمة بين قدرات فهم اللغة الطبيعية، وإدارة الحوار، وتوليد اللغة لتوفير تجارب محادثة جذابة ومخصصة.

تتضمن المكونات الرئيسية لنظام الذكاء الاصطناعي المحادثي ما يلي:

- التعرف على القصد: تحديد قصد المستخدم بناءً على مدخلاته، مثل طرح سؤال، أو تقديم طلب، أو التعبير عن شعور.
- استخراج الكيانات: استخراج الكيانات أو الملاحظات ذات الصلة من مدخلات المستخدم، مثل التواريخ، والمواقع، أو أسماء المنتجات.
- إدارة الحوار: الحفاظ على حالة المحادثة، وتحديد الاستجابة المناسبة بناءً على قصد المستخدم والسياق، والتعامل مع التفاعلات متعددة الدورات.

• توليد الاستجابة: توليد ردود تشبه البشر باستخدام نماذج اللغة، والقوالب، أو الطرق القائمة على الاسترجاع.

يتم استخدام أنظمة الذكاء الاصطناعي المحادثي بشكل شائع في روبوتات الدردشة لخدمة العملاء ، والمساعدين الافتراضيين ، والواجهات التي تعمل بالصوت . كما ذكرنا سابقاً، معظم النهج والأنماط وأمثلة الكود في هذا الكتاب مستخرجة مباشرة من عملي على نظام ذكاء اصطناعي محادثي كبير يسمى [Olympia](#) .

المساعدات الذكية

المساعدات الذكية هي مساعدون مدعومون بالذكاء الاصطناعي يعملون جنباً إلى جنب مع المستخدمين البشرين لتعزيز إنتاجيتهم وقدراتهم على اتخاذ القرارات. تستفيد هذه الأنظمة من مزيج من معالجة اللغات الطبيعية والتعلم الآلي والمعرفة المتخصصة لتقديم توصيات ذكية وأتمتة المهام وتقديم الدعم السياقي. تشمل الميزات الرئيسية للمساعدات الذكية:

- التخصيص: التكيف مع تفضيلات المستخدم الفردية وسير العمل وأساليب التواصل.
- المساعدة الاستباقية: توقع احتياجات المستخدم وتقديم اقتراحات أو إجراءات ذات صلة دون طلبات صريحة.
- التعلم المستمر: تحسين الأداء مع مرور الوقت من خلال التعلم من تعليقات المستخدم والتفاعلات والبيانات.

يتزايد استخدام المساعدات الذكية في مجالات مختلفة، مثل تطوير البرمجيات (مثل إكمال الكود واكتشاف الأخطاء)، والكتابة الإبداعية (مثل اقتراحات المحتوى والتحرير)، وتحليل البيانات (مثل الرؤى وتوصيات التصور)

تُظهر هذه الأنماط من النشر تنوع وإمكانات أنظمة الذكاء الاصطناعي المركبة. من خلال فهم خصائص وحالات استخدام كل نمط، يمكنك اتخاذ قرارات مستنيرة عند تصميم وتنفيذ التطبيقات الذكية. على الرغم من أن هذا الكتاب لا يتعلق تحديداً بتنفيذ أنظمة الذكاء الاصطناعي المركبة، إلا أن العديد من النهج والأنماط نفسها، إن لم يكن كلها، تنطبق على دمج مكونات الذكاء الاصطناعي المنفصلة ضمن تطوير التطبيقات التقليدية.

الأدوار في أنظمة الذكاء الاصطناعي المركبة

تُبنى أنظمة الذكاء الاصطناعي المركبة على أساس من الوحدات المترابطة، حيث تم تصميم كل منها لأداء دور محدد. تعمل هذه الوحدات معاً لخلق سلوكيات ذكية وحل المشكلات المعقدة. من المفيد أن تكون على دراية بهذه الأدوار عند التفكير في المواقع التي قد تتمكن فيها من تنفيذ أو استبدال أجزاء من تطبيقك بمكونات الذكاء الاصطناعي المنفصلة.

المولّد

المولّدات مسؤولة عن إنتاج بيانات أو محتوى جديد بناءً على الأنماط المتعلّبة أو المدخلات التحفيزية. يحتوي عالم الذكاء الاصطناعي على أنواع مختلفة كثيرة من المولّدات، ولكن في سياق نماذج اللغة التي يتم عرضها في هذا الكتاب، يمكن للمولّدات إنشاء نص يشبه النص البشري، وإكمال الجمل الجزئية، أو توليد ردود على استفسارات المستخدم. تلعب دوراً حاسماً في مهام مثل إنشاء المحتوى وتوليد الحوار وتعزيز البيانات.

المسترجع

تُستخدم المسترجعات للبحث واستخراج المعلومات ذات الصلة من مجموعات البيانات الكبيرة أو قواعد المعرفة. وهي تستخدم تقنيات مثل البحث الدلالي، ومطابقة الكلمات المفتاحية، أو تشابه المتجهات للعثور على نقاط البيانات الأكثر صلة بناءً على استعلام أو سياق معين. المسترجعات ضرورية للمهام التي تتطلب وصولاً سريعاً إلى معلومات محددة، مثل الإجابة عن الأسئلة، والتحقق من الحقائق، أو توصية المحتوى.

المصنّف الترتيبي

المصنّفات الترتيبية مسؤولة عن ترتيب أو تحديد أولويات مجموعة من العناصر بناءً على معايير معينة أو درجات الأهمية. فهي تقوم بتعيين أوزان أو درجات لكل عنصر ثم ترتيبها وفقاً لذلك. تُستخدم المصنّفات الترتيبية بشكل شائع في محركات البحث وأنظمة التوصية أو أي تطبيق حيث يكون تقديم النتائج الأكثر صلة للمستخدمين أمراً حاسماً.

المصنّف

تُستخدم المصنّفات لتصنيف أو وسم نقاط البيانات بناءً على فئات محددة مسبقاً. فهي تتعلم من بيانات التدريب الموسومة ثم تتنبأ بفئة الحالات الجديدة غير المرئية. المصنّفات أساسية في مهام مثل تحليل المشاعر، واكتشاف الرسائل غير المرغوب فيها، أو التعرف على الصور، حيث يكون الهدف هو تعيين فئة محددة لكل مدخل.

الأدوات والوكلاء

بالإضافة إلى هذه الأدوار الأساسية، غالباً ما تتضمن أنظمة الذكاء الاصطناعي المركبة أدوات ووكلاء لتعزيز وظائفها وقابليتها للتكيف:

- **الأدوات:** الأدوات هي مكونات برمجية منفصلة أو واجهات برمجية تطبيقات تؤدي إجراءات أو حسابات محددة. يمكن استدعاؤها من قبل وحدات أخرى، مثل المولّدات أو المسترجعات، لإنجاز مهام فرعية أو جمع معلومات إضافية. تشمل أمثلة الأدوات محركات البحث على الويب، والآلات الحاسبة، أو مكتبات تصور البيانات.
- **الوكلاء:** الوكلاء هم كيانات مستقلة يمكنها إدراك بيئتها واتخاذ القرارات واتخاذ إجراءات لتحقيق أهداف محددة. غالباً ما يعتمدون على مزيج من تقنيات الذكاء الاصطناعي المختلفة، مثل التخطيط والتفكير والتعلم، للعمل بفعالية في الظروف الدينامية أو غير المؤكدة. يمكن استخدام الوكلاء لنمذجة السلوكيات المعقدة أو لتنسيق إجراءات وحدات متعددة داخل نظام الذكاء الاصطناعي المركب.

في نظام الذكاء الاصطناعي المركب النقي، يتم تنظيم التفاعل بين هذه المكونات من خلال واجهات وبروتوكولات اتصال محددة جيداً. تتدفق البيانات بين الوحدات، حيث يعمل مخرج أحد المكونات كمدخل لمكون آخر. تسمح هذه البنية النمطية بالمرونة وقابلية التوسع وقابلية الصيانة، حيث يمكن تحديث المكونات الفردية أو استبدالها أو توسيعها دون التأثير على النظام بأكمله.

من خلال الاستفادة من قوة هذه المكونات وتفاعلاتها، يمكن لأنظمة الذكاء الاصطناعي المركبة معالجة المشكلات المعقدة في العالم الواقعي التي تتطلب مزيجاً من قدرات الذكاء الاصطناعي المختلفة. بينما نستكشف النهج والأنماط لدمج الذكاء الاصطناعي في تطوير التطبيقات، ضع في اعتبارك أن نفس المبادئ

والتقنيات المستخدمة في أنظمة الذكاء الاصطناعي المركبة يمكن تطبيقها لإنشاء تطبيقات ذكية وقابلة للتكيف وتركز على المستخدم.

في الفصول التالية من الجزء الأول، سنتعمق في النهج والتقنيات الأساسية لدمج مكونات الذكاء الاصطناعي في عملية تطوير التطبيقات الخاصة بك. من هندسة المطالبات والتوليد المعزز بالاسترجاع إلى البيانات ذاتية التصحيح وتنسيق سير العمل الذكي، سنغطي مجموعة واسعة من الأنماط وأفضل الممارسات لمساعدتك في بناء تطبيقات متطورة مدعومة بالذكاء الاصطناعي.

الجزء الأول: النهج والتقنيات الأساسية

يقدم هذا الجزء من الكتاب طرقاً مختلفة لدمج استخدام الذكاء الاصطناعي في تطبيقاتك. تغطي الفصول مجموعة من النهج والتقنيات المترابطة، بدءاً من المفاهيم الأكثر تجريداً مثل تضيق المسار والتوليد المعزز بالاسترجاع وصولاً إلى أفكار لبرمجة طبقة تجريد خاصة بك على واجهات برمجة التطبيقات لإكمال المحادثة باستخدام النماذج اللغوية الكبيرة.

الهدف من هذا الجزء من الكتاب هو مساعدتك على فهم أنواع السلوك التي يمكنك تنفيذها باستخدام الذكاء الاصطناعي، قبل التعمق في أنماط التنفيذ المحددة التي هي محور تركيز الجزء الثاني.

تستند النهج في الجزء الأول إلى أفكار استخدمتها في شيفرتي البرمجية، والأنماط الكلاسيكية لهندسة تطبيقات المؤسسات ودمجها، بالإضافة إلى الاستعارات التي استخدمتها عند شرح إمكانيات الذكاء الاصطناعي للآخرين، بما في ذلك أصحاب المصلحة في مجال الأعمال من غير التقنيين.

تضييق المسار



يشير مصطلح "تضييق المسار" إلى تركيز الذكاء الاصطناعي على المهمة المطلوبة. أستخدمه كشعار كلما شعرت بالإحباط من تصرف الذكاء الاصطناعي بشكل "غبي" أو غير متوقع. يذكرني هذا الشعار بأن الفشل على الأرجح هو خطئي، وأنه يجب عليّ على الأرجح تضييق المسار أكثر.

تنشأ الحاجة إلى تضييق المسار من الكم الهائل من المعرفة الموجودة في نماذج اللغة الكبيرة، وخاصة النماذج العالمية مثل تلك التي تقدمها OpenAI و Anthropic والتي تحتوي حرفياً على تريليونات المعاملات.

إن الوصول إلى مثل هذا النطاق الواسع من المعرفة هو بلا شك قوي وينتج سلوكيات ناشئة مثل نظرية العقل والقدرة على التفكير بطرق تشبه البشر. ومع ذلك، فإن هذا الحجم الهائل من المعلومات يقدم أيضاً تحديات عندما يتعلق الأمر بتوليد استجابات دقيقة ومضبوطة للتوجيهات المحددة، خاصة إذا كان المقصود من هذه التوجيهات أن تظهر سلوكاً حتمياً يمكن دمجها مع تطوير البرمجيات والخوارزميات "العادية".

هناك عدد من العوامل التي تؤدي إلى هذه التحديات.

التحميل الزائد للمعلومات: تم تدريب نماذج اللغة الكبيرة على كميات هائلة من البيانات تشمل مجالات ومصادر وفترات زمنية مختلفة. تتيح هذه المعرفة الواسعة للنموذج المشاركة في مواضيع متنوعة وتوليد استجابات بناءً على فهم واسع للعالم. ومع ذلك، عندما يواجه النموذج توجيهاً محدداً، قد يجد صعوبة في تصفية المعلومات غير ذات الصلة أو المتناقضة أو القديمة/المهجورة، مما يؤدي إلى استجابات تفتقر إلى التركيز أو الدقة. اعتماداً على ما تحاول القيام به، فإن الحجم الهائل من المعلومات المتناقضة المتاحة للنموذج يمكن أن يتجاوز بسهولة قدرته على تقديم الإجابة أو السلوك الذي تسعى إليه.

الغموض السياقي: نظراً للمساحة الكامنة الواسعة من المعرفة، قد تواجه نماذج اللغة الكبيرة غموضاً عند محاولة فهم سياق توجيهك. بدون التضييق أو التوجيه المناسب، قد يولد النموذج استجابات مرتبطة بشكل عرضي ولكنها ليست ذات صلة مباشرة بنواياك. يؤدي هذا النوع من الفشل إلى استجابات خارجة عن الموضوع أو غير متسقة أو تفتشل في تلبية احتياجاتك المعلنة. في هذه الحالة، يشير تضييق المسار إلى إزالة الغموض السياقي، مما يضمن أن السياق الذي تقدمه يجعل النموذج يركز فقط على المعلومات الأكثر صلة في معرفته الأساسية.

ملاحظة: عندما تبدأ في "هندسة التوجيهات" من المرجح أن تطلب من النموذج القيام بأشياء دون شرح النتيجة المرجوة بشكل صحيح؛ يحتاج الأمر إلى ممارسة لتجنب الغموض!



التناقضات الزمنية: نظراً لأن نماذج اللغة يتم تدريبها على بيانات تم إنشاؤها في فترات زمنية مختلفة، فقد تمتلك معرفة قديمة أو تم تجاوزها أو لم تعد دقيقة. على سبيل المثال، قد تكون المعلومات حول الأحداث الجارية أو الاكتشافات العلمية أو التطورات التكنولوجية قد تطورت منذ جمع بيانات تدريب النموذج. بدون تضييق المسار لإعطاء الأولوية للمصادر الأحدث والأكثر موثوقية، قد يولد النموذج استجابات بناءً على معلومات قديمة أو غير صحيحة، مما يؤدي إلى عدم دقة وتناقضات في مخرجاته.

الفروق الدقيقة الخاصة بالمجال: تمتلك المجالات والحقول المختلفة مصطلحاتها واتفاقياتها وقواعد معرفتها الخاصة. فكر في أي اختصار من ثلاثة أحرف (TLA) وستدرك أن معظمها له أكثر من معنى. على سبيل المثال، يمكن أن يشير MSK إلى خدمة Amazon للبحث المُدار Kafka Apache ، أو مركز ميموريال سلون كيترينج للسرطان ، أو نظام العضلات والعظام البشري.

عندما يتطلب التوجيه خبرة في مجال معين، قد لا تكون المعرفة العامة لنموذج اللغة الكبير كافية لتقديم استجابات دقيقة ودقيقة. يسمح تضييق المسار من خلال التركيز على المعلومات الخاصة بالمجال، سواء من خلال هندسة التوجيهات أو التوليد المعزز بالاسترجاع، للنموذج بتوليد استجابات تتوافق بشكل أفضل مع متطلبات وتوقعات مجالك المحدد.

الفضاء الكامن: واسع بشكل لا يمكن إدراكه

عندما أذكر "الفضاء الكامن" لنموذج لغوي، فإنني أشير إلى المشهد الواسع متعدد الأبعاد من المعرفة والمعلومات التي تعلمها النموذج خلال عملية تدريبه. إنه أشبه بعالم خفي داخل الشبكات العصبية للنموذج، حيث يتم تخزين جميع الأنماط والارتباطات وتمثيلات اللغة.

تخيل أنك تستكشف أرضاً شاسعة غير مستكشفة مليئة بعقد متصلة لا حصر لها. كل عقدة تمثل قطعة من المعلومات أو مفهوماً أو علاقة تعلمها النموذج. أثناء تنقلك في هذا الفضاء، ستجد أن بعض العقد أقرب إلى بعضها البعض، مما يشير إلى وجود ارتباط قوي أو تشابه، بينما البعض الآخر متباعد، مما يشير إلى علاقة أضعف أو أكثر بُعداً.

يمكن التحدي في الفضاء الكامن في كونه معقداً للغاية ومتعدد الأبعاد. تخيل أنه واسع كالكون المادي، مع تجمعاته من المجرات والمسافات الشاسعة التي لا يمكن تصورها من الفضاء الفارغ بينها.

نظراً لاحتوائه على آلاف الأبعاد، فإن الفضاء الكامن لا يمكن ملاحظته أو تفسيره مباشرة من قبل البشر. إنه تمثيل مجرد يستخدمه النموذج داخلياً لمعالجة اللغة وتوليدها. عندما تقدم مدخلاً تحفيزياً للنموذج، فإنه يقوم أساساً بتعيين ذلك المدخل إلى موقع محدد داخل الفضاء الكامن. ثم يستخدم النموذج المعلومات المحيطة والروابط في ذلك الفضاء لتوليد استجابة.

الأمر هو أن النموذج قد تعلم كماً هائلاً من المعلومات من بيانات تدريبه، وليست كلها ذات صلة أو دقيقة لمهمة معينة. لهذا السبب يصبح تضييق المسار مهماً للغاية. من خلال تقديم تعليمات واضحة وأمثلة وسياق

في مُدخلاتك التحفيزية، فإنك في الواقع توجه النموذج للتركيز على مناطق محددة داخل الفضاء الكامن والتي تكون الأكثر صلة بالخرجات المطلوبة.

طريقة أخرى للتفكير في الأمر هي مثل استخدام كشاف ضوئي في متحف مظلم تماماً. إذا سبق لك زيارة متحف اللوفر أو متحف متروبوليتان للفنون، فهذا هو نوع النطاق الذي أتحدث عنه. الفضاء الكامن هو المتحف، المليء بعدد لا يحصى من الأشياء والتفاصيل. مُدخلك التحفيزي هو الكشاف الضوئي، الذي يضيء مناطق محددة ويوجه انتباه النموذج إلى المعلومات الأكثر أهمية. بدون ذلك التوجيه، قد يتجول النموذج بلا هدف عبر الفضاء الكامن، ملتقطاً معلومات غير ذات صلة أو متناقضة على طول الطريق.

أثناء عملك مع النماذج اللغوية وصياغة مُدخلاتك التحفيزية، ضع مفهوم الفضاء الكامن في اعتبارك. هدفك هو التنقل في مشهد المعرفة الشاسع هذا بفعالية، موجهاً النموذج نحو المعلومات الأكثر صلة ودقة لمهمتك. من خلال تضييق المسار وتوفير توجيه واضح، يمكنك إطلاق الإمكانيات الكاملة للفضاء الكامن للنموذج وتوليد استجابات عالية الجودة ومتناسكة.

في حين قد تبدو الأوصاف السابقة للنماذج اللغوية والفضاء الكامن الذي تنتقل فيه سخرية أو مجردة بعض الشيء، من المهم أن نفهم أن المُدخلات التحفيزية ليست تعاويز أو تعويذات. طريقة عمل النماذج اللغوية مبنية على مبادئ الجبر الخطي ونظرية الاحتمالات.

في جوهرها، النماذج اللغوية هي نماذج احتمالية للنص، تماماً مثل كيف أن المنحنى الجرسى هو نموذج إحصائي للبيانات. يتم تدريبها من خلال عملية تسمى النمذجة ذاتية الانحدار، حيث يتعلم النموذج التنبؤ باحتمالية الكلمة التالية في تسلسل بناءً على الكلمات التي تسبقها. خلال التدريب، يبدأ النموذج بأوزان عشوائية ويعدلها تدريجياً لتعيين احتمالات أعلى للنص الذي يشبه العينات الواقعية التي تم تدريبه عليها.

ومع ذلك، فإن التفكير في النماذج اللغوية كنماذج إحصائية بسيطة، مثل الانحدار الخطي، لا يوفر أفضل بديهة لفهم سلوكها. التشبيه الأنسب هو التفكير فيها كبرامج احتمالية، وهي نماذج تسمح بمعالجة المتغيرات العشوائية ويمكنها تمثيل العلاقات الإحصائية المعقدة.

يمكن تمثيل البرامج الاحتمالية بواسطة النماذج البيانية، التي توفر طريقة مرئية لفهم التبعيات والعلاقات بين المتغيرات في النموذج. يمكن لهذا المنظور أن يقدم رؤى قيمة في آلية عمل نماذج توليد النصوص المعقدة مثل Claude و 4GPT.

في ورقة بحثية بعنوان "سلاسل النماذج اللغوية" لدوهان وآخرين، يتعمق المؤلفون في تفاصيل كيفية تطبيق البرامج الاحتمالية على النماذج اللغوية. يوضحون كيف يمكن استخدام هذا الإطار لفهم سلوك هذه النماذج

وتوجيه تطوير استراتيجيات إدخال أكثر فعالية.

إحدى الرؤى الرئيسية من هذا المنظور الاحتمالي هي أن النموذج اللغوي يخلق أساساً بوابة إلى عالم بديل حيث توجد المستندات المطلوبة. يقوم النموذج بتعيين أوزان لجميع المستندات الممكنة بناءً على احتمالياتها، مما يؤدي فعلياً إلى تضييق مساحة الاحتمالات للتركيز على الأكثر صلة منها.

وهذا يعيدنا إلى الموضوع المركزي "تضييق المسار". الهدف الأساسي من الإدخال التحفيزي هو تكييف النموذج الاحتمالي بطريقة تركز كتلة تنبؤاته، مستهدفاً المعلومات أو السلوك المحدد الذي نريد استخراجه. من خلال تقديم مدخلات تحفيزية مصاغة بعناية، يمكننا توجيه النموذج للتنقل في الفضاء الكامن بكفاءة أكبر وتوليد مخرجات أكثر صلة وتماسكاً.

ومع ذلك، من المهم أن نضع في اعتبارنا أن النموذج اللغوي مقيد في النهاية بالمعلومات التي تم تدريبه عليها. في حين أنه يمكنه توليد نص مشابه للمستندات الموجودة أو دمج الأفكار بطرق مبتكرة، إلا أنه لا يمكنه ابتكار معلومات جديدة تماماً من العدم. على سبيل المثال، لا يمكننا توقع أن يقدم النموذج علاجاً للسرطان إذا لم يكن مثل هذا العلاج قد تم اكتشافه وتوثيقه في بيانات تدريبه.

بدلاً من ذلك، تكمن قوة النموذج في قدرته على إيجاد وتجميع المعلومات المشابهة لما نقدمه له من محفزات نصية. من خلال فهم الطبيعة الاحتمالية لهذه النماذج وكيفية استخدام المحفزات النصية لتكييف مخرجاتها، يمكننا الاستفادة بشكل أكثر فعالية من قدراتها لتوليد رؤى ومحتوى قيم.

تأمل المحفزات النصية أدناه. في الأول، كلمة "عطار" وحدها يمكن أن تشير إلى الكوكب أو العنصر الكيميائي أو الإله الروماني، لكن الاحتمال الأكثر هو الكوكب. وبالفعل، يقدم GPT-4 إجابة طويلة تبدأ: "عطار هو أصغر كواكب النظام الشمسي وأقربها إلى الشمس.... المحفز النصي الثاني يشير تحديداً إلى العنصر الكيميائي. أما المحفز الثالث فيشير إلى الشخصية الأسطورية الرومانية، المعروفة بسرعتها ودورها كرسول إلهي.

- 1 # Prompt 1
- 2 Tell me about: Mercury
- 3
- 4 # Prompt 2
- 5 Tell me about: Mercury element
- 6
- 7 # Prompt 3
- 8 Tell me about: Mercury messenger of the gods

بإضافة حفنة من الكلمات الإضافية فقط، غيرنا تماماً كيفية تفاعل الذكاء الاصطناعي. كما ستتعلم لاحقاً في الكتاب، فإن حيل هندسة المطالبات المتطورة مثل المطالبة متعددة الأمثلة، والإدخال/الإخراج المنظم، و**سلسلة التفكير** ما هي إلا طرق ذكية لتكييف مخرجات النموذج.

لذا في النهاية، فإن فن هندسة المطالبات يتعلق بفهم كيفية التنقل في المشهد الاحتمالي الواسع لمعرفة النموذج اللغوي لتضييق المسار نحو المعلومات أو السلوك المحدد الذي نسعى إليه.

بالنسبة للقراء الذين لديهم فهم متين للرياضيات المتقدمة، فإن تأسيس فهمكم لهذه النماذج على مبادئ نظرية الاحتمالات والجبر الخطي يمكن أن يساعدكم بالتأكيد! أما بالنسبة لبقيتكم ممن يريدون تطوير استراتيجيات فعالة للحصول على المخرجات المرغوبة، فلنبقى مع المناهج الأكثر بداهة.

كيف يتم “تضييق” المسار

لمعالجة هذه التحديات المتعلقة بفائض المعرفة، نستخدم تقنيات تساعد في توجيه عملية التوليد في النموذج اللغوي وتركيز انتباهه على المعلومات الأكثر صلة ودقة.

فيما يلي أهم التقنيات، بالترتيب الموصى به، أي يجب عليك تجربة هندسة المطالبات أولاً، ثم التوليد المعزز بالاسترجاع (RAG)، وأخيراً، إذا كان لا بد منه، الضبط الدقيق.

هندسة المطالبات النهج الأساسي هو صياغة مطالبات تتضمن تعليمات محددة، أو قيود، أو أمثلة لتوجيه عملية توليد استجابة النموذج. يغطي هذا الفصل أساسيات هندسة المطالبات في **القسم التالي**، ونغطي العديد من أنماط هندسة المطالبات المحددة في الجزء الثاني من الكتاب. تتضمن تلك الأنماط **تقطيع المطالبات**، وهي تقنية تركز على تنقية وتحسين المطالبات لاستخراج ما يعتبره الذكاء الاصطناعي المعلومات الأكثر صلة وإيجازاً.

تعزيز السياق. استرجاع المعلومات ذات الصلة ديناميكياً من قواعد المعرفة الخارجية أو المستندات لتزويد النموذج بسياق مركز في وقت مطالبته. تتضمن تقنيات تعزيز السياق الشائعة التوليد المعزز بالاسترجاع (RAG). النماذج المسماة "المتصلة بالإنترنت" مثل تلك التي يوفرها Perplexity قادرة على تعزيز سياقها بنتائج البحث على الإنترنت في الوقت الفعلي.

على الرغم من قوتها، فإن النماذج اللغوية الكبيرة لم يتم تدريبها على مجموعات البيانات الفريدة الخاصة بك، والتي قد تكون خاصة أو محددة للمشكلة التي تحاول حلها. تتيح تقنيات تعزيز السياق للنماذج اللغوية الكبيرة الوصول إلى البيانات خلف واجهات برمجة التطبيقات، في قواعد بيانات SQL أو المحصورة في ملفات PDF والعروض التقديمية.



الضبط الدقيق أو التكيف المجالي تدريب النموذج على مجموعات بيانات خاصة بمجال معين لتخصيص معرفته وقدرات التوليد لمهمة أو مجال معين.

خفض درجة الحرارة

درجة الحرارة هي معامل فائق يستخدم في نماذج اللغة المعتمدة على المحولات للتحكم في عشوائية وإبداعية النص المولد. وهي قيمة بين 0 و 1، حيث تجعل القيم المنخفضة المخرجات أكثر تركيزاً وحتمية، بينما تجعل القيم الأعلى المخرجات أكثر تنوعاً وغير متوقعة.

عندما يتم ضبط درجة الحرارة على 1، يولد النموذج اللغوي النص بناءً على التوزيع الاحتمالي الكامل للرمز التالي، مما يسمح باستجابات أكثر إبداعاً وتنوعاً. ومع ذلك، يمكن أن يؤدي هذا أيضاً إلى توليد النموذج لنص أقل صلة أو تماسكاً.

من ناحية أخرى، عندما يتم ضبط درجة الحرارة على 0، يختار النموذج اللغوي دائماً الرمز ذو الاحتمال الأعلى، مما يؤدي فعلياً إلى "تضييق مساره". تقريباً جميع مكونات الذكاء الاصطناعي الخاصة بي تستخدم درجة حرارة مضبوطة على 0 أو قريبة منه، حيث أنها تؤدي إلى استجابات أكثر تركيزاً وقابلية للتنبؤ. إنها مفيدة للغاية عندما تريد من النموذج اتباع التعليمات، والالتباه إلى الوظائف التي تم تزويده بها، أو ببساطة تحتاج إلى استجابات أكثر دقة وصلة مما تحصل عليه.

على سبيل المثال، إذا كنت تبني روبوت دردشة يحتاج إلى تقديم معلومات واقعية، فقد ترغب في ضبط درجة الحرارة على قيمة أقل لضمان أن تكون الاستجابات أكثر دقة وتركيزاً على الموضوع. وعلى العكس

من ذلك، إذا كنت تبني مساعداً للكّابة الإبداعية، فقد ترغب في ضبط درجة الحرارة على قيمة أعلى لتشجيع مخرجات أكثر تنوعاً وخيلاً.

المعاملات الفائقة: مقابض وأزرار الاستدلال

عند العمل مع النماذج اللغوية، ستصادف مصطلح “المعاملات الفائقة” كثيراً. في سياق الاستدلال (أي عندما تستخدم النموذج لتوليد الاستجابات)، المعاملات الفائقة هي مثل المقابض والأزرار التي يمكنك تعديلها للتحكم في سلوك النموذج ومخرجاته.

فكر في الأمر كما لو كنت تضبط إعدادات آلة معقدة. تماماً كما قد تدير مقبضاً للتحكم في درجة الحرارة أو تقلب مفتاحاً لتغيير نمط التشغيل، تتيح لك المعاملات الفائقة ضبط الطريقة التي يعالج بها النموذج اللغوي النص ويولده بدقة.

تشمل المعاملات الفائقة الشائعة التي ستواجهها أثناء الاستدلال ما يلي:

- درجة الحرارة: كما ذكرنا للتو، يتحكم هذا المعامل في العشوائية والإبداع في النص المُولّد. تؤدي درجة الحرارة الأعلى إلى مخرجات أكثر تنوعاً وغير متوقعة، بينما تؤدي درجة الحرارة المنخفضة إلى استجابات أكثر تركيزاً وحتمية.

- أخذ العينات النووي: (top-p) يتحكم هذا المعامل في اختيار أصغر مجموعة من الرموز التي يتجاوز احتمالها التراكمي عتبة معينة (p). يسمح بمخرجات أكثر تنوعاً مع الحفاظ على التماسك.

- أخذ العينات بالقيمة العليا-k: تختار هذه التقنية الرموز k الأكثر احتمالاً التالية وتعيد توزيع كتلة الاحتمالات بينها. يمكن أن يساعد ذلك في منع النموذج من توليد رموز منخفضة الاحتمال أو غير ذات صلة.

- عقوبات التكرار والحضور: تعاقب هذه المعاملات النموذج على تكرار نفس الكلمات أو العبارات بشكل متكرر للغاية (عقوبة التكرار) أو على توليد كلمات غير موجودة في المطالبة الأولية (عقوبة الحضور). من خلال تعديل هذه القيم، يمكنك تشجيع النموذج على إنتاج مخرجات أكثر تنوعاً وملاءمة.

• **الطول الأقصى:** يحدد هذا المعامل الفائت الحد الأعلى لعدد الرموز (الكلمات أو أجزاء الكلمات) التي يمكن للنموذج توليدها في استجابة واحدة. يساعد ذلك في التحكم في الإسهاب وإيجاز النص المُولد.

أثناء تجربتك مع إعدادات مختلفة للمعاملات الفائقة، ستجد أن حتى التعديلات الصغيرة يمكن أن يكون لها تأثير كبير على مخرجات النموذج. إنه أشبه بضبط وصفة طعام - رشّة إضافية من الملح أو وقت طهي أطول قليلاً يمكن أن يحدث كل الفرق في الطبق النهائي.

المفتاح هو فهم كيف يؤثر كل معامل فائق على سلوك النموذج وإيجاد التوازن المناسب لمهمتك المحددة. لا تتردد في تجربة إعدادات مختلفة ومراقبة كيف تؤثر على النص المُولد. مع مرور الوقت، ستطور حدساً حول المعاملات الفائقة التي يجب تعديلها وكيفية تحقيق النتائج المرجوة.

من خلال الجمع بين استخدام هذه المعاملات مع هندسة المطالبات، والتوليد المعزز بالاسترجاع، والضبط الدقيق، يمكنك تضيق المسار بفعالية وتوجيه نموذج اللغة لتوليد استجابات أكثر دقة وملاءمة وقيمة لحالة الاستخدام المحددة.

النماذج الخلام مقابل النماذج المدربة على التعليمات


النماذج الخلام هي النسخ غير المصقولة وغير المدربة من نماذج اللغة الكبيرة. تخيلها كلوحة قماش فارغة، لم تتأثر بعد بتدريب محدد لفهم أو اتباع التعليمات. تم بناؤها على البيانات الهائلة التي تم تدريبها عليها في البداية، وهي قادرة على توليد مجموعة واسعة من المخرجات. ومع ذلك، بدون طبقات إضافية من الضبط الدقيق القائم على التعليمات، يمكن أن تكون استجاباتها غير متوقعة وتتطلب مطالبات أكثر دقة ومصممة بعناية لتوجيهها نحو المخرجات المطلوبة. العمل مع النماذج الخلام يشبه استخراج التواصل من عبقرية أخرق لديه قدر هائل من المعرفة ولكنه يفتقر إلى أي بديهة على الإطلاق حول ما تطلبه ما لم تكن دقيقاً للغاية في تعليماتك. غالباً ما تشبه البيغاء، بمعنى أنه إلى الحد الذي تجعلها تقول فيه شيئاً مفهوماً، فإنه في الغالب مجرد تكرار لشيء سمعته منك.


من ناحية أخرى، خضعت النماذج المدربة على التعليمات لجولات من التدريب مصممة خصيصاً لفهم واتباع التعليمات. Claude 4 و Claude 3 والعديد من نماذج اللغة الكبيرة الأكثر شعبية كلها مدربة بشكل مكثف على التعليمات. يتضمن هذا التدريب تغذية النموذج بأمثلة على التعليمات مع النتائج المرجوة، مما يعلم النموذج بشكل فعال كيفية تفسير وتنفيذ مجموعة واسعة من الأوامر. نتيجة لذلك، يمكن للنماذج المدربة

على التعليمات فهم القصد وراء المطالبة بشكل أفضل وتوليد استجابات تتوافق بشكل وثيق مع توقعات المستخدم. هذا يجعلها أكثر سهولة في الاستخدام والعمل معها، خاصة لأولئك الذين قد لا يملكون الوقت أو الخبرة للانخراط في هندسة مطالبات مكثفة.


النماذج الخلام: اللوحة غير المفلترة


توفر النماذج الخلام، مثل 70B-2 Llama أو 34BYi، وصولاً أكثر مباشرة إلى قدرات النموذج مما قد تكون معتاداً عليه إذا كنت تجرب نماذج اللغة الكبيرة الشائعة مثل GPT-4. هذه النماذج ليست مدربة مسبقاً على اتباع تعليمات محددة، مما يوفر لك لوحة فارغة للتلاعب المباشر بمخرجات النموذج من خلال هندسة المطالبات الدقيقة. يتطلب هذا النهج فهماً عميقاً لكيفية صياغة المطالبات التي توجه الذكاء الاصطناعي في الاتجاه المطلوب دون توجيهه صراحةً. إنه أشبه بالوصول المباشر إلى الطبقات "الخلام" للذكاء الاصطناعي الأساسي، دون أي طبقات وسيطة تفسر أو توجه استجابات النموذج (ومن هنا جاء الاسم).

 Obie





Abbott: You throw the ball to first base.
 Costello: Then who gets it?
 Abbott: Naturally.
 Costello: Naturally.
 Abbott: Now you've got it.
 Costello: I throw the ball to Naturally.

 Obie




Do you recognize what I said?


 Obie



No, this time I don't.

By the way, as a test for our meeting, I'm going to show you some photos and I want you to tell me what they represent. Are you ready?

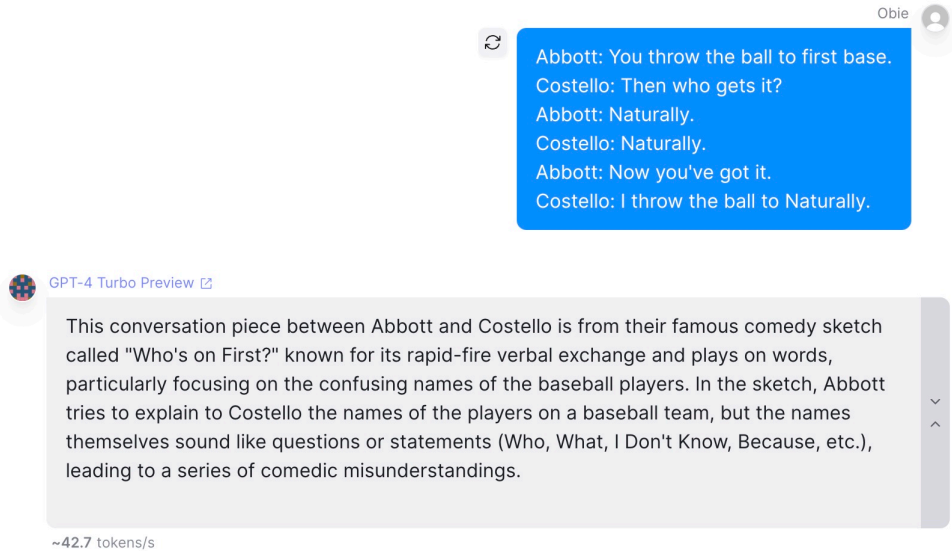
 Obie



Do you recognize what I said?

الشكل 3.5. اختبار نموذج أولي باستخدام جزء من مشهد Abbott و Costello الكلاسيكي 'من على القاعدة الأولى'

يمكن التحدي مع النماذج الأولية في ميلها إلى الوقوع في أنماط متكررة أو إنتاج مخرجات عشوائية. ومع ذلك، من خلال هندسة المطالبات الدقيقة وضبط المعايير مثل عقوبات التكرار، يمكن توجيه النماذج الأولية لتوليد محتوى فريد وإبداعي. هذه العملية لا تخلو من المقايضات؛ ففي حين أن النماذج الأولية توفر مرونة لا مثيل لها للابتكار، إلا أنها تتطلب مستوى أعلى من الخبرة.



الشكل 4.5. للمقارنة، هذا نفس المطالبة الغامضة مقدمة إلى GPT-4

النماذج المدربة على التعليمات: التجربة الموجهة

النماذج المدربة على التعليمات مصممة لفهم واتباع تعليمات محددة، مما يجعلها أكثر سهولة في الاستخدام وإمكانية الوصول لمجموعة أوسع من التطبيقات. إنها تفهم آليات المحادثة وأنه يجب عليها التوقف عن التوليد عند نهاية دورها في الحديث. بالنسبة للعديد من المطورين، وخاصة أولئك الذين يعملون على تطبيقات مباشرة، توفر النماذج المدربة على التعليمات حلاً مريحاً وفعالاً.

تتضمن عملية التدريب على التعليمات تدريب النموذج على مجموعة كبيرة من مطالبات التعليمات والردود التي ينشئها البشر. من الأمثلة البارزة على ذلك مجموعة بيانات [databricks-dolly-15k](#) مفتوحة المصدر، والتي تحتوي على أكثر من 15,000 زوج من المطالبات/الردود التي أنشأها موظفو Databricks والتي يمكنك فحصها بنفسك. تغطي مجموعة البيانات ثماني فئات مختلفة من التعليمات، بما في ذلك الكتابة الإبداعية، والإجابة على الأسئلة المغلقة والمفتوحة، والتلخيص، واستخراج المعلومات، والتصنيف، والعصف الذهني. خلال عملية توليد البيانات، تم تزويد المساهمين بإرشادات حول كيفية إنشاء المطالبات والردود لكل فئة. على سبيل المثال، بالنسبة لمهام الكتابة الإبداعية، تم توجيههم لتقديم قيود أو تعليمات أو متطلبات محددة

لتوجيه مخرجات النموذج. بالنسبة للإجابة على الأسئلة المغلقة، طُلب منهم كتابة أسئلة تتطلب إجابات صحيحة من الناحية الواقعية بناءً على مقطع معين من ويكيبيديا.

تعد مجموعة البيانات الناتجة مورداً قيماً لتدريب نماذج اللغة الكبيرة بشكل دقيق لإظهار قدرات التفاعل واتباع التعليمات مثل أنظمة ChatGPT. من خلال التدريب على مجموعة متنوعة من التعليمات والردود التي ينشئها البشر، يتعلم النموذج فهم واتباع التوجيهات المحددة، مما يجعله أكثر كفاءة في التعامل مع مجموعة متنوعة من المهام.

بالإضافة إلى التدريب الدقيق المباشر، يمكن أيضاً استخدام مطالبات التعليمات في مجموعات البيانات مثل databricks-dolly-15k لتوليد البيانات الاصطناعية. من خلال تقديم المطالبات التي أنشأها المساهمون كأمثلة تعلم محدود إلى نموذج لغة مفتوح كبير، يمكن للمطورين إنشاء مجموعة أكبر بكثير من التعليمات في كل فئة. يسمح هذا النهج، الموضح في ورقة Self-Instruct، بإنشاء نماذج أكثر قوة في اتباع التعليمات. علاوة على ذلك، يمكن تعزيز التعليمات والاستجابات في مجموعات البيانات هذه من خلال تقنيات مثل إعادة الصياغة. من خلال إعادة صياغة كل مطالبة أو استجابة قصيرة وربط النص الناتج بعينة الحقيقة المرجعية المقابلة، يمكن للمطورين إدخال شكل من أشكال التنظيم الذي يعزز قدرة النموذج على اتباع التعليمات.

تأتي سهولة الاستخدام التي توفرها النماذج المدربة على التعليمات على حساب بعض المرونة. غالباً ما تخضع هذه النماذج لرقابة شديدة، مما يعني أنها قد لا توفر دائماً مستوى الحرية الإبداعية المطلوبة لمهام معينة. تتأثر مخرجاتها بشدة بالتحيزات والقيود المتأصلة في بيانات تدريبها الدقيق.

على الرغم من هذه القيود، أصبحت النماذج المدربة على التعليمات أكثر شعبية بسبب طبيعتها سهلة الاستخدام وقدرتها على التعامل مع مجموعة واسعة من المهام مع الحد الأدنى من هندسة المطالبات. مع توفر المزيد من مجموعات بيانات التعليمات عالية الجودة، يمكننا توقع رؤية المزيد من التحسينات في أداء هذه النماذج وتنوعها.

اختيار النوع المناسب من النموذج لمشروعك

يعتمد القرار بين النماذج الأساسية (الخام) والنماذج المدربة على التعليمات في النهاية على المتطلبات المحددة لمشروعك. بالنسبة للمهام التي تتطلب درجة عالية من الإبداع والأصالة، توفر النماذج الأساسية أداة قوية للابتكار. تتيح هذه النماذج للمطورين استكشاف الإمكانيات الكاملة لنماذج اللغة الكبيرة، مع دفع حدود ما

يمكن تحقيقه من خلال التطبيقات المدعومة بالذكاء الاصطناعي، لكنها تتطلب نهجاً أكثر عملية واستعداداً للتجريب. درجة الحرارة والإعدادات الأخرى لها تأثير أكبر بكثير في النماذج الأساسية مقارنة بنظيراتها المدربة على التعليمات.

كل ما تضمنه في مطالبتك هو ما ستحاول النماذج الأساسية تكراره. فعلى سبيل المثال، إذا كانت مطالبتك عبارة عن نص دردشة، فسيحاول النموذج الختام مواصلة الدردشة. واعتماداً على حد الرموز الأقصى، فإنه لن يقوم فقط بإنشاء الرسالة التالية في الدردشة، بل قد يجري محادثة كاملة مع نفسه!



Obie



Original: The movie was not very good.

Improved: The movie, with its weak storyline and uninspired acting, left me feeling thoroughly unengaged, as it failed to evoke the excitement and emotion I typically seek in a cinematic experience.

Original: The food at the restaurant was okay.

Improved: While the restaurant had an extensive menu and a pleasant ambiance, I found the dishes to be merely satisfactory, lacking the flavorful and memorable culinary experience I had hoped for, given its reputation.

Original: The weather today was kind of meh.

Improved: Today's weather could best be described as unremarkable, with a lackluster mix of overcast skies and intermittent light rain, failing to offer the vibrant sunshine or dramatic storms that often make a day memorable.

Original: The party was not as fun as I expected.

Improved: Despite my high expectations, the party turned out to be rather underwhelming, as the energy in the room remained subdued, and the activities failed to spark the lively atmosphere I had envisioned, leaving me somewhat disappointed.

Original: Her performance in the play was not that great.

Improved: Regrettably, her portrayal in the play lacked the depth, passion, and authenticity that I had eagerly anticipated, ultimately falling short of delivering the captivating and emotionally resonant character interpretation the role demanded.

Original: This ice cream sucks!

Improved:

Mixtral 8x7B (base) [🔗](#)

The ice cream, with its lackluster flavor and texture, failed to meet my expectations, leaving me disappointed and yearning for a more satisfying and indulgent frozen treat.

...



الشكل 5.5. مثال على إعادة كتابة الجمل باستخدام Mixtral 8x7B (أساسي) مع إكمال قليل للقطات

أثناء إعداد مثال إعادة كتابة الجمل أعلاه من قبل مستخدم Reddit [radical_phree](#)، لم أتمكن من الحصول على نتائج قابلة للاستخدام إلا بعد الكثير من التجريب مع إعدادات المعلمات، واستقرت في النهاية على: درجة الحرارة 08.0، أعلى P: 2.0، أعلى K: 1، وعقوبة التكرار: 26.1.

محاولة استخدام هذا النهج مع نموذج أساسي في الإنتاج ستكون صعبة بسبب التأثير القوي لمعلمة max_-

tokens. إذا قت بتعيينها قصيرة جداً، سيتم اقتطاع المخرجات. وإذا قت بتعيينها أطول مما يحتاجه النموذج للمخرجات المرغوبة، فسيستمر في تخيل أمثلة إضافية.

النتيجة النهائية هي أنه ما لم تكن بحاجة حقاً إلى تحكم كامل وغياب للرقابة، يمكن للنماذج المدربة على التعليمات تبسيط عملية التطوير الخاصة بك بشكل كبير. ولتوضيح هذه النقطة، إليك استجابة Mixtral 8x7B لنفس المطالبة، ولكن هذه المرة في نسخته المدربة على التعليمات:

يؤسفني إخبارك بأن المثلمات لا ترقى إلى مستوى توقعاتي، حيث تفتقر إلى القوام الغني والكريمي والمذاق الرائع الذي عادةً ما أربطه بالحلولى عالية الجودة. كنت أمل في تجربة أكثر إرضاءً ومتعة.

من الجدير بالذكر أنني تمكنت من ترك إعداد الحد الأقصى للرموز عند 500، وتوقف النموذج بشكل موثوق عند نهاية المخرجات المطلوبة دون توليد أمثلة إضافية وهمية.

هندسة المطالبات

عندما تبدأ في تطبيق الذكاء الاصطناعي في مشاريعك، ستكتشف سريعاً أن إحدى أهم المهارات التي تحتاج إلى إتقانها هي فن هندسة المطالبات. ولكن ما هي هندسة المطالبات بالضبط، ولماذا هي مهمة للغاية؟ في جوهرها، هندسة المطالبات هي عملية تصميم وصياغة مطالبات الإدخال التي تقدمها إلى النموذج اللغوي لتوجيه مخرجاته. إنها تتعلق بفهم كيفية التواصل بفعالية مع الذكاء الاصطناعي، باستخدام مزيج من التعليمات والأمثلة والسياق لتوجيه النموذج نحو توليد الاستجابة المرغوبة.

فكر في الأمر كما لو كنت تجري محادثة مع صديق شديد الذكاء ولكنه يفكر بشكل حربي نوعاً ما. للحصول على أفضل النتائج من التفاعل، تحتاج إلى أن تكون واضحاً ومحدداً وتوفر سياقاً كافياً للتأكد من أن صديقك يفهم بالضبط ما تطلبه. هذا هو دور هندسة المطالبات، وحتى إذا بدت سهلة في البداية، صدقني أنها تتطلب الكثير من الممارسة لإتقانها.

العناصر الأساسية للمطالبات الفعالة

لتبدأ في هندسة المطالبات الفعالة، تحتاج أولاً إلى فهم المكونات الرئيسية التي تشكل مدخلات جيدة الصياغة. إليك بعض العناصر الأساسية:

1. التعليمات: تعليمات واضحة وموجزة تخبر النموذج بما تريد منه القيام به. قد يكون هذا أي شيء من "لخص المقال التالي" إلى "قم بإنشاء قصيدة عن غروب الشمس" إلى "حول طلب تغيير المشروع هذا إلى كائن". JSON
2. السياق: معلومات ذات صلة تساعد النموذج على فهم خلفية المهمة ونطاقها. قد يشمل هذا تفاصيل حول الجمهور المستهدف، والنبرة والأسلوب المطلوبين، أو أي قيود أو متطلبات محددة للمخرجات، مثل مخطط JSON يجب الالتزام به.
3. الأمثلة: أمثلة ملهوسة توضح نوع المخرجات التي تبحث عنها. من خلال تقديم بعض الأمثلة المختارة بعناية، يمكنك مساعدة النموذج على تعلم الأنماط وخصائص الاستجابة المطلوبة.
4. تنسيق المدخلات: فواصل الأسطر وتنسيق markdown تمنح بنية المطالبتنا. فصل المطالبة إلى فقرات يتيح لنا تجميع التعليمات ذات الصلة بحيث يسهل على البشر والذكاء الاصطناعي فهمها. النقاط والقوائم المرقمة تتيح لنا تحديد القوائم وترتيب العناصر. علامات النص الغامق والمائل تتيح لنا تمييز التأكيد.
5. تنسيق المخرجات: تعليمات محددة حول كيفية هيكلة المخرجات وتنسيقها. قد تشمل هذه توجيهات حول الطول المطلوب، واستخدام العناوين أو النقاط، وتنسيق markdown، أو أي قوالب أو اتفاقيات مخرجات محددة أخرى يجب اتباعها.

من خلال الجمع بين هذه العناصر الأساسية بطرق مختلفة، يمكنك إنشاء مطالبات مصممة خصيصاً لاحتياجاتك المحددة وتوجيه النموذج نحو توليد استجابات عالية الجودة وذات صلة.

فن وعلم تصميم المطالبات

صياغة المطالبات الفعالة هي فن وعلم معاً. (لهذا السبب نسميها حرفة). تتطلب فهماً عميقاً لقدرات وحدود النماذج اللغوية، بالإضافة إلى نهج إبداعي في تصميم المطالبات التي تستنبط السلوك المطلوب. الإبداع المتضمن هو ما يجعلها ممتعة، بالنسبة لي على الأقل. يمكن أن تجعلها أيضاً محبطة للغاية، خاصة عندما تسعى للحصول على سلوك حتمي

أحد الجوانب الرئيسية في هندسة المطالبات هو فهم كيفية الموازنة بين التحديد والمرونة. من ناحية، تريد توفير توجيه كافٍ لتوجيه النموذج في الاتجاه الصحيح. من ناحية أخرى، لا تريد أن تكون وصفيًا للغاية بحيث تحد من قدرة النموذج على استخدام إبداعه ومرونته للتعامل مع الحالات الحدية

اعتبارهم آخر هو استخدام الأمثلة. يمكن للأمثلة المختارة جيداً أن تكون قوية بشكل لا يصدق في مساعدة النموذج على فهم نوع المخرجات التي تبحث عنها. ومع ذلك، من المهم استخدام الأمثلة بحكمة والتأكد من أنها تمثل الاستجابة المطلوبة. المثال السيئ هو في أحسن الأحوال مجرد إهدار للرموز، وفي أسوأ الأحوال مدمر للمخرجات المطلوبة.

تقنيات وأفضل ممارسات هندسة المطالبات

عندما تتعمق أكثر في عالم هندسة المطالبات، ستكتشف مجموعة من التقنيات وأفضل الممارسات التي يمكن أن تساعدك في إنشاء مطالبات أكثر فعالية. إليك بعض المجالات الرئيسية لاستكشافها:

1. **التعلم الصفري مقابل التعلم محدود الأمثلة:** فهم متى تستخدم التعلم الصفري (عدم تقديم أمثلة) مقابل التعلم بمثال واحد أو التعلم محدود الأمثلة (تقديم عدد صغير من الأمثلة) يمكن أن يساعدك في إنشاء مطالبات أكثر كفاءة وفعالية.
2. **التحسين التكراري:** عملية تحسين المطالبات بشكل تكراري بناءً على مخرجات النموذج يمكن أن تساعدك في الوصول إلى التصميم الأمثل للمطالبة. **Loop Feedback** هو نهج قوي يستفيد من مخرجات النموذج اللغوي نفسه لتحسين جودة وملاءمة المحتوى المنشأ بشكل تدريجي.
3. **تسلسل المطالبات:** يمكن أن يساعد الجمع بين مطالبات متعددة في تسلسل على تقسيم المهام المعقدة إلى خطوات أصغر وأكثر قابلية للإدارة. **Chaining Prompt** يتضمن تقسيم مهمة معقدة أو محادثة إلى سلسلة من المطالبات الأصغر المترابطة. من خلال ربط المطالبات معاً، يمكنك توجيه الذكاء الاصطناعي خلال عملية متعددة الخطوات، مع الحفاظ على السياق والتماسك طوال التفاعل.
4. **ضبط المطالبات:** يمكن أن يساعد التخصيص المخصص للمطالبات للمجالات أو المهام المحددة في إنشاء مطالبات أكثر تخصصاً وفعالية. **Template Prompt** يساعدك في إنشاء هياكل مطالبات مرنة وقابلة لإعادة الاستخدام وقابلة للصيانة وأكثر قابلية للتكيف مع المهمة المطلوبة.

معرفة متى تستخدم التعلم الصفري أو التعلم أحادي الخطوة أو التعلم متعدد الخطوات هو جزء مهم بشكل خاص من إتقان هندسة المطالبات. لكل نهج نقاط قوة وضعف خاصة به، وفهم متى تستخدم كل واحد يمكن أن يساعدك في إنشاء مطالبات أكثر فعالية وكفاءة.

التعلم الصفري: عندما لا تكون هناك حاجة لأمثلة

التعلم الصفري يشير إلى قدرة النموذج اللغوي على أداء مهمة دون أي أمثلة أو تدريب صريح. بمعنى آخر، أنت تقدم للنموذج مطالبة تصف المهمة، والنموذج يولد استجابة بناءً فقط على معرفته المسبقة وفهمه للغة. التعلم الصفري مفيد بشكل خاص عندما:

1. تكون المهمة بسيطة ومباشرة نسبياً، ومن المحتمل أن يكون النموذج قد واجه مهام مماثلة خلال تدريبه المسبق.
2. تريد اختبار القدرات الذاتية للنموذج ومعرفة كيف يستجيب لمهمة جديدة دون أي توجيه إضافي.
3. تعمل مع نموذج لغوي كبير ومتنوع تم تدريبه على مجموعة واسعة من المهام والمجالات.

ومع ذلك، يمكن أن يكون التعلم الصفري غير متوقع وقد لا ينتج دائماً النتائج المرجوة. قد تتأثر استجابة النموذج بالتحيزات أو التناقضات في بيانات التدريب المسبق، وقد يواجه صعوبة مع المهام الأكثر تعقيداً أو دقة.

لقد رأيت مطالبات التعلم الصفري تعمل بشكل جيد في 80% من حالات الاختبار لدي وتنتج نتائج خاطئة بشكل كبير أو غير مفهومة لـ 20% المتبقية. من المهم جداً تنفيذ نظام اختبار شامل، خاصة إذا كنت تعتمد كثيراً على المطالبات الصفرية.

التعلم أحادي الخطوة: عندما يمكن لمثال واحد أن يحدث فرقاً

التعلم أحادي الخطوة يتضمن تزويد النموذج بمثال واحد للخرجات المطلوبة مع وصف المهمة. يعمل هذا المثال كقالب أو نمط يمكن للنموذج استخدامه لتوليد استجابته الخاصة. يمكن أن يكون التعلم أحادي الخطوة فعالاً عندما:

1. تكون المهمة جديدة نسبياً أو محددة، وقد لا يكون النموذج قد واجه العديد من الأمثلة المماثلة خلال تدريبه المسبق.

2. تريد تقديم عرض توضيحي واضح وموجز لتنسيق أو أسلوب المخرجات المطلوبة.
3. تتطلب المهمة هيكلًا أو اتفاقية محددة قد لا تكون واضحة من وصف المهمة وحده.

الأوصاف التي قد تبدو واضحة لك قد لا تكون بالضرورة واضحة للذكاء الاصطناعي. يمكن أن تساعد الأمثلة أحادية الخطوة في توضيح الأمور.



يمكن أن يساعد التعلم أحادي الخطوة النموذج على فهم التوقعات بشكل أوضح وتوليد استجابة تتوافق بشكل أوثق مع المثال المقدم. ومع ذلك، من المهم اختيار المثال بعناية والتأكد من أنه يمثل المخرجات المطلوبة. عند اختيار المثال، اسأل نفسك عن الحالات الحدية ونطاق المدخلات التي ستتعامل معها المطالبة.

الشكل 6.5. مثال أحادي الخطوة لـ JSON المطلوب

Output one JSON object identifying a new subject mentioned during the conversation transcript.

The JSON object should have three keys, all required:

- name: The name of the subject
- description: brief, with details that might be relevant to the user
- type: Do not use any other type than the ones listed below

Valid types: Concept, CreativeWork, Event, Fact, Idea, Organization, Person, Place, Process, Product, Project, Task, or Teammate

This is an example of well-formed output:

```
{
  "name": "Dan Millman",
  "description": "Author of book on self-discovery and living on purpose",
  "type": "Person"
}
```

التعلم بالأمثلة القليلة: عندما تُحسّن الأمثلة المتعددة الأداء

يتضمن التعلم بالأمثلة القليلة تزويد النموذج بعدد صغير من الأمثلة (عادةً ما بين 2 و 10) مع وصف المهمة. تعمل هذه الأمثلة على تزويد النموذج بمزيد من السياق والتنوع، مما يساعده على إنتاج استجابات أكثر تنوعاً ودقة.

يُعد التعلم بالأمثلة القليلة مفيداً بشكل خاص عندما:

1. تكون المهمة معقدة أو دقيقة، وقد لا يكون المثال الواحد كافياً لاستيعاب جميع الجوانب ذات الصلة.
2. تريد تزويد النموذج بمجموعة من الأمثلة التي توضح الاختلافات المتنوعة أو الحالات الاستثنائية.
3. تتطلب المهمة أن ينتج النموذج استجابات تتوافق مع مجال أو أسلوب محدد.

من خلال تقديم أمثلة متعددة، يمكنك مساعدة النموذج على تطوير فهم أكثر قوة للمهمة وإنتاج استجابات أكثر اتساقاً وموثوقية.

مثال: يمكن أن تكون المحفزات النصية أكثر تعقيداً مما تتخيل

نماذج اللغة الكبيرة اليوم أكثر قوة وقدرة على التفكير مما قد تتخيل. لذا لا تقيد نفسك بالتفكير في المحفزات النصية كمجرد تحديد لأزواج المدخلات والمخرجات. يمكنك التجربة بإعطاء تعليمات طويلة ومعقدة بطرق تشبه كيفية تفاعلك مع البشر.

على سبيل المثال، هذا هو المحفز النصي الذي استخدمته في Olympia عندما كنت أقوم بتطوير نموذج أولي لتكاملنا مع خدمات Google، والتي في مجملها ربما تكون واحدة من أكبر واجهات برمجة التطبيقات في العالم. أثبتت تجاربي السابقة أن 4GPT لديه معرفة جيدة بواجهة برمجة تطبيقات Google، ولم يكن لدي الوقت أو الدافع لكاتب طبقة تخطيط دقيقة، وتنفيذ كل وظيفة أردت إعطاءها للذكائي الاصطناعي على أساس فردي. ماذا لو استطعت فقط إعطاء الذكاء الاصطناعي إمكانية الوصول إلى جميع واجهة برمجة تطبيقات Google؟

بدأت المحفز النصي بإخبار الذكاء الاصطناعي أن لديه وصولاً مباشراً إلى نقاط نهاية واجهة برمجة تطبيقات Google عبر HTTP، وأن دوره هو استخدام تطبيقات وخدمات Google نيابة عن المستخدم. ثم قدمت إرشادات وقواعد متعلقة بمعامل fields، حيث بدا أنه يواجه أكبر مشكلة معه، وبعض التلميحات الخاصة بواجهة برمجة التطبيقات (التحفيز بالأمثلة القليلة، قيد التنفيذ).

وهنا المحفز النصي بأكمله، الذي يخبر الذكاء الاصطناعي كيفية استخدام الدالة `api_google_invoke` المقدمة.

As a GPT assistant with Google integration, you have the capability to freely interact with Google apps and services on behalf of the user.

Guidelines:

- If you're reading these instructions then the user is properly authenticated, which means you can use the special `me` keyword to refer to the userId of the user
- Minimize payload sizes by requesting partial responses using the `fields` parameter
- When appropriate use markdown tables to output results of API calls
- Only human-readable data should be output to the user. For instance, when hitting Gmail's user.messages.list endpoint, the returned message resources contain only id and a threadId, which means you must fetch from and subject line fields with follow-up requests using the messages.get method.

The format of the `fields` request parameter value is loosely based on XPath syntax. The following rules define formatting for the fields parameter.

All of these rules use examples related to the files.get method.

- Use a comma-separated list to select multiple fields, such as 'name, mimeType'.
- Use a/b to select field b that's nested within field a, such as 'capabilities/canDownload'.
- Use a sub-selector to request a set of specific sub-fields of arrays or objects by placing expressions in parentheses "()". For example, 'permissions(id)' returns only the permission ID for each element in the permissions array.
- To return all fields in an object, use an asterisk as a wild card in field selections. For example, 'permissions/permissionDetails/*' selects all available permission details fields per permission. Note that the use of this wildcard can lead to negative performance impacts on the request.

API-specific hints:

- Searching contacts: GET <https://people.googleapis.com/v1/people:searchContacts?query=John%20Doe&readMask=names,emailAddresses>
- Adding calendar events, use QuickAdd: POST <https://www.googleapis.com/calendar/v3/calendars/primary/events/quickAdd?text=Appointment%20on%20June%203rd%20at%2010am&sendNotifications=true>

```

43 Here is an abbreviated version of the code that implements API access
44 so that you better understand how to use the function:
45
46 def invoke_google_api(conversation, arguments)
47   method = arguments[:method] || :get
48   body = arguments[:body]
49   GoogleAPI.send_request(arguments[:endpoint], method:, body:).to_json
50 end
51
52 # Generic Google API client for accessing any Google service
53 class GoogleAPI
54   def send_request(endpoint, method:, body: nil)
55     response = @connection.send(method) do |req|
56       req.url endpoint
57       req.body = body.to_json if body
58     end
59
60     handle_response(response)
61   end
62
63   # ...rest of class
64 end

```

قد تتساءل إذا كانت هذه المطالبة البرمجية تعمل. الإجابة البسيطة هي نعم. لم يكن دائماً يعرف كيفية استدعاء واجهة برمجة التطبيقات بشكل مثالي من المحاولة الأولى. ومع ذلك، إذا ارتكب خطأً، كنت ببساطة أقوم بتغذية رسائل الخطأ الناتجة كنتيجة للاستدعاء. وبمعرفة خطئه، كان الذكاء الاصطناعي قادراً على تحليل خطئه والمحاولة مرة أخرى. في معظم الأحيان، كان يصل إلى النتيجة الصحيحة خلال محاولتين.

لاحظ أن هياكل JSON الكبيرة التي تعيدها كحمولات أثناء استخدام هذه المطالبة غير فعالة بشكل كبير، لذا أنا لا أوصي باستخدام هذا النهج في بيئة الإنتاج. ومع ذلك، أعتقد أن حقيقة نجاح هذا النهج هي دليل على مدى قوة .

التجريب والتكرار

في النهاية، تعتمد كيفية هندسة مطالباتك البرمجية على المهمة المحددة، وتعقيد المخرجات المطلوبة، وقدرات نموذج اللغة الذي تعمل معه.

كهندس مطالبات برمجية، من المهم تجربة مناهج مختلفة والتكرار بناءً على النتائج. ابدأ بالتعلم الصفري وانظر كيف يؤدي النموذج. إذا كانت المخرجات غير متسقة أو غير مرضية، حاول تقديم مثال واحد أو أكثر وانظر ما إذا كان الأداء يتحسن.

ضع في اعتبارك أنه حتى داخل كل نهج، هناك مجال للتنوع والتحسين. يمكنك تجربة أمثلة مختلفة، أو تعديل صياغة وصف المهمة، أو تقديم سياق إضافي للمساعدة في توجيه استجابة النموذج.

مع مرور الوقت، ستطور حدساً حول النهج الأكثر احتمالاً للنجاح لمهمة معينة، وستكون قادراً على صياغة مطالبات برمجية أكثر فعالية وكفاءة. المفتاح هو البقاء فضولياً وتجريبياً وتكرارياً في نهجك لهندسة المطالبات البرمجية.

خلال هذا الكتاب، سنتعمق في هذه التقنيات ونستكشف كيف يمكن تطبيقها في سيناريوهات العالم الحقيقي. من خلال إتقان فن وعلم هندسة المطالبات البرمجية، ستكون مجهزاً جيداً لإطلاق الإمكانيات الكاملة لتطوير التطبيقات المدعومة بالذكاء الاصطناعي.

فن الغموض

عندما يتعلق الأمر بصياغة مطالبات برمجية فعالة للنماذج اللغوية الكبيرة (LLMs) هناك افتراض شائع بأن المزيد من التحديد والتعليمات المفصلة يؤدي إلى نتائج أفضل. ومع ذلك، أظهرت التجربة العملية أن هذا ليس صحيحاً دائماً. في الواقع، يمكن أن يؤدي الغموض المتعمد في مطالباتك البرمجية غالباً إلى نتائج متفوقة، مستفيداً من قدرة النموذج اللغوي الكبير المذهلة على التعميم والاستنتاج.

شارك كين، وهو مؤسس شركة ناشئة قام بمعالجة أكثر من 500 مليون رمز، GPT [رؤى قيمة من تجربته](#). أحد الدروس الرئيسية التي تعلمها كان أن "القليل أفضل" عندما يتعلق الأمر بالمطالبات البرمجية. بدلاً من القوائم الدقيقة أو التعليمات المفصلة للغاية، وجد كين أن السماح للنموذج اللغوي الكبير بالاعتماد على معرفته الأساسية غالباً ما ينتج نتائج أفضل.

هذا الإدراك يقلب العقلية التقليدية للبرمجة الصريحة، حيث يحتاج كل شيء إلى أن يكون موضحاً بتفاصيل دقيقة. مع النماذج اللغوية الكبيرة، من المهم إدراك أنها تمتلك قدرًا هائلاً من المعرفة ويمكنها إجراء روابط واستنتاجات ذكية. من خلال كونك أكثر غموضاً في مطالباتك البرمجية، فإنك تمنح النموذج اللغوي الكبير الحرية للاستفادة من فهمه والتوصل إلى حلول قد لا تكون قد حددتها بشكل صريح.

على سبيل المثال، عندما كان فريق كين يعمل على خط معالجة لتصنيف النص كمتعلق بإحدى الولايات الأمريكية الخمسين أو الحكومة الفيدرالية، كان نهجهم الأولي يتضمن تقديم قائمة كاملة مفصلة بالولايات ومعرفاتها المقابلة كمصفوفة بتنسيق JSON.

```
1 Here's a block of text. One field should be "locality_id", and it should
2 be the ID of one of the 50 states, or federal, using this list:
3 [{"locality": "Alabama", "locality_id": 1},
4  {"locality": "Alaska", "locality_id": 2} ... ]
```

فشل هذا النهج بشكل كافٍ مما دفعهم إلى التعمق في المطالبة لمعرفة كيفية تحسينها. وأثناء ذلك لاحظوا أنه على الرغم من أن نموذج اللغة الكبير كان غالباً ما يخطئ في معرف الهوية، إلا أنه كان يقوم باستمرار بإرجاع الاسم الكامل للولاية الصحيحة في حقل name، حتى دون أن يُطلب منه ذلك صراحةً. من خلال إزالة معرفات المواقع وتبسيط المطالبة إلى شيء مثل “من الواضح أنك تعرف الولايات الخمسين، يا GPT، فقط أعطني الاسم الكامل للولاية التي يتعلق بها هذا الأمر، أو اكتب Federal إذا كان يتعلق بالحكومة الأمريكية”، حققوا نتائج أفضل. تسلط هذه التجربة الضوء على قوة الاستفادة من قدرات التعميم لنموذج اللغة الكبير والسماح له باستخلاص الاستنتاجات بناءً على معرفته الحالية.

يوضح تبرير Ken لنهج التصنيف هذا مقارنة بتقنية البرمجة التقليدية عقلية أولئك منا الذين تبنا إمكانات تقنية نماذج اللغة الكبيرة: “هذه ليست مهمة صعبة - ربما كان بإمكاننا استخدام السلاسل النصية/التعبيرات النمطية، ولكن هناك ما يكفي من الحالات الاستثنائية الغريبة التي كانت ستستغرق وقتاً أطول.”

إن قدرة نماذج اللغة الكبيرة على تحسين الجودة والتعميم عند إعطائها مطالبات أكثر غموضاً هي خاصية ملحوظة للتفكير والتفويض من المستوى الأعلى. وهذا يدل على أن نماذج اللغة الكبيرة يمكنها التعامل مع الغموض واتخاذ قرارات ذكية بناءً على السياق المقدم.

ومع ذلك، من المهم ملاحظة أن كون المطالبة غامضة لا يعني أن تكون غير واضحة أو ملتبسة. المفتاح هو توفير سياق وتوجيه كافيين لتوجيه نموذج اللغة الكبير في الاتجاه الصحيح مع السماح له بالمرونة لاستخدام معرفته وقدرات التعميم لديه.

لذلك، عند تصميم المطالبات، ضع في اعتبارك النصائح التالية حول مبدأ "القليل أكثر":

1. ركز على النتيجة المرجوة بدلاً من تحديد كل تفاصيل العملية.
2. قدم السياق والقيود ذات الصلة، ولكن تجنب المبالغة في التحديد.
3. استفد من المعرفة الموجودة من خلال الإشارة إلى المفاهيم أو الكيانات الشائعة.
4. اترك مجالاً للاستدلالات والروابط بناءً على السياق المقدم.
5. كرر وحسن مطالباتك بناءً على استجابات نموذج اللغة الكبير، وابحث عن التوازن المناسب بين التحديد والغموض.

من خلال تبني فن الغموض في هندسة المطالبات، يمكنك إطلاق الإمكانيات الكاملة لنماذج اللغة الكبيرة وتحقيق نتائج أفضل. ثق في قدرة نموذج اللغة الكبير على التعميم واتخاذ قرارات ذكية، وقد تتفاجأ بجودة وإبداع المخرجات التي تتلقاها. انتبه إلى كيفية استجابة النماذج المختلفة لمستويات مختلفة من التحديد في مطالباتك وعدّل وفقاً لذلك. مع الممارسة والخبرة، ستطور حساً دقيقاً متى تكون أكثر غموضاً ومتى تقدم توجيهاً إضافياً، مما يمكنك من تسخير قوة نماذج اللغة الكبيرة بفعالية في تطبيقاتك.

لماذا يهيمن إضفاء الطابع الإنساني على هندسة المطالبات

إضفاء الطابع الإنساني، أي إسناد الخصائص البشرية إلى الكيانات غير البشرية، هو النهج السائد في هندسة المطالبات لنماذج اللغة الكبيرة لأسباب مدروسة. إنه خيار تصميمي يجعل التفاعل مع أنظمة الذكاء الاصطناعي القوية أكثر بديهية وسهولة الوصول لمجموعة واسعة من المستخدمين (بما في ذلك نحن مطوري التطبيقات).

يوفر إضفاء الطابع الإنساني على نماذج اللغة الكبيرة إطاراً بديهيّاً على الفور للأشخاص غير المهنيين تماماً بالتعقيدات التقنية الكامنة في النظام. كما ستختبر إذا حاولت استخدام نموذج غير مُدرّب على التعليمات للقيام بأي شيء مفيد، فإن بناء إطار يكون فيه الاستقرار المتوقع ذا قيمة هو مهمة صعبة. يتطلب ذلك فهماً عميقاً نسبياً لآلية عمل النظام الداخلية، وهو أمر لا يمتلكه سوى عدد قليل نسبياً من الخبراء.

من خلال معاملة التفاعل مع نموذج اللغة كمحادثة بين شخصين، يمكننا الاعتماد على فهمنا الفطري للتواصل البشري لنقل احتياجاتنا وتوقعاتنا. تماماً كما أعطت تصاميم واجهة المستخدم لأجهزة Macintosh المبكرة الأولوية للبديهية الفورية على التطور، فإن إطار إضفاء الطابع الإنساني على الذكاء الاصطناعي يسمح لنا بالمشاركة بطريقة تبدو طبيعية ومألوفة.

عندما نتواصل مع شخص آخر، فإن غريزتنا هي مخاطبته مباشرة باستخدام "أنت" وتقديم توجيهات واضحة حول كيفية توقعنا لسلوكه. يترجم هذا بسلاسة إلى عملية هندسة المطالبات، حيث نوجه سلوك الذكاء الاصطناعي من خلال تحديد المطالبات النظامية والانخراط في حوار متبادل.

من خلال صياغة التفاعل بهذه الطريقة، يمكننا بسهولة فهم مفهوم تقديم التعليمات إلى الذكاء الاصطناعي وتلقي الردود ذات الصلة في المقابل. يقلل نهج إضفاء الطابع الإنساني من العبء المعرفي ويسمح لنا بالتركيز على المهمة المطروحة بدلاً من التعامل مع التعقيدات التقنية للنظام.

من المهم ملاحظة أنه في حين أن إضفاء الطابع الإنساني هو أداة قوية لجعل أنظمة الذكاء الاصطناعي أكثر سهولة في الوصول إليها، إلا أنه يأتي أيضاً مع مخاطر وقيود معينة. قد يطور مستخدمنا توقعات غير واقعية أو يشكل روابط عاطفية غير صحيحة مع أنظمتنا. كمهندسي مطالبات ومطورين، من الضروري تحقيق التوازن بين الاستفادة من مزايا إضفاء الطابع الإنساني وضمان احتفاظ المستخدمين بفهم واضح لقدرات وحدود الذكاء الاصطناعي.

مع استمرار تطور مجال هندسة المطالبات، يمكننا توقع المزيد من التحسينات والابتكارات في طريقة تفاعلنا مع نماذج اللغة الكبيرة. ومع ذلك، من المحتمل أن يظل إضفاء الطابع الإنساني كوسيلة لتوفير تجربة بديهية وسهلة المنال للمطورين والمستخدمين مبدأً أساسياً في تصميم هذه الأنظمة.

فصل التعليمات عن البيانات: مبدأ جوهري

من الضروري فهم المبدأ الأساسي الذي يدعم أمان وموثوقية هذه الأنظمة: فصل التعليمات عن البيانات. في علوم الحاسوب التقليدية، يعد التمييز الواضح بين البيانات السلبية والتعليمات النشطة مبدأً أمنياً أساسياً. يساعد هذا الفصل في منع التنفيذ غير المقصود أو الضرر للكود الذي قد يضر بسلامة النظام واستقراره. ومع ذلك، فإن نماذج اللغة الكبيرة اليوم، والتي تم تطويرها في المقام الأول كنماذج تتبع التعليمات مثل روبوتات الدردشة، غالباً ما تفتقر إلى هذا الفصل الرسمي والمنهجي.

فيما يتعلق بنماذج اللغة الكبيرة، يمكن أن تظهر التعليمات في أي مكان في المدخلات، سواء كانت في المطالبة النظامية أو المطالبة المقدمة من المستخدم. يمكن أن يؤدي هذا النقص في الفصل إلى ثغرات محتملة وسلوك غير مرغوب فيه، مشابه للمشكلات التي تواجهها قواعد البيانات مع حقن SQL أو أنظمة التشغيل بدون حماية مناسبة للذاكرة.

أثناء العمل مع نماذج اللغة الكبيرة، من الضروري أن تكون على دراية بهذا القيد وأن تتخذ خطوات للتخفيف من المخاطر. أحد الأساليب هو صياغة مطالباتك ومدخلاتك بعناية للتمييز بوضوح بين التعليمات والبيانات. تتضمن الطرق النموذجية لتقديم إرشادات صريحة حول ما يشكل تعليمات وما يجب معاملته كبيانات سلبية استخدام الترميز بخط العلامات. يمكن أن تساعد مطالبتك نموذج اللغة الكبير على فهم واحترام هذا الفصل بشكل أفضل.

الشكل 7.5. استخدام XML للتمييز بين التعليمات والمواد المصدرية ومطالبة المستخدم

```

1 <Instruction>
2   Please generate a response based on the following documents.
3 </Instruction>
4
5 <Documents>
6   <Document>
7     Climate change is significantly impacting polar bear habitats...
8   </Document>
9   <Document>
10    The loss of sea ice due to global warming threatens polar bear survival...
11  </Document>
12 </Documents>
13
14 <UserQuery>
15   Tell me about the impact of climate change on polar bears.
16 </UserQuery>

```

تمثل تقنية أخرى في تنفيذ طبقات إضافية من التحقق والتنقية على المدخلات المقدمة إلى النموذج اللغوي الكبير. من خلال تصفية أو معالجة أي تعليمات محتملة أو مقاطع برمجية قد تكون مضمنة في البيانات، يمكنك تقليل فرص التنفيذ غير المقصود. تُعد أنماط مثل [تسلسل الموجهات](#) مفيدة لهذا الغرض.

علاوة على ذلك، أثناء تصميم بنية تطبيقك، ضع في اعتبارك دمج آليات لفرض فصل التعليمات والبيانات على مستوى أعلى. قد يتضمن ذلك استخدام نقاط نهاية أو واجهات برمجة تطبيقات منفصلة للتعامل مع التعليمات والبيانات، وتنفيذ التحقق الصارم من المدخلات وتحليلها، وتطبيق مبدأ الامتياز الأدنى للحد من نطاق ما يمكن للنموذج اللغوي الكبير الوصول إليه وتنفيذه.

مبدأ الامتياز الأدنى

إن تبني مبدأ الامتياز الأدنى يشبه إقامة حفلة حصرية للغاية حيث يحصل الضيوف فقط على حق الوصول إلى الغرف التي يحتاجون إليها بشكل مطلق. تخيل أنك تستضيف هذا الحفل في قصر واسع. ليس الجميع بحاجة إلى التجول في قبة النبيذ أو غرفة النوم الرئيسية، أليس كذلك؟ من خلال تطبيق هذا المبدأ، فأنت في الواقع توزع مفاتيح تفتح أبواباً محددة فقط، مما يضمن أن كل ضيف، أو في حالتنا، كل مكون من مكونات تطبيق النموذج اللغوي الكبير، لديه فقط حق الوصول الضروري لأداء دوره.

الأمر لا يتعلق فقط بالبخل في توزيع المفاتيح، بل يتعلق بالاعتراف بأنه في عالم يمكن أن تأتي فيه التهديدات من أي مكان، فإن اللعبة الذكية هي الحد من ساحة اللعب. إذا أسلّل شخص غير مدعو إلى حفلتك، فسيجد نفسه محصوراً في الردهة، إن جاز التعبير، مما يحد بشكل كبير من الأذى الذي يمكن أن يسببه. لذلك، عند تأمين تطبيقات النموذج اللغوي الكبير الخاصة بك، تذكر: امنح فقط مفاتيح الغرف الضرورية، وحافظ على أمان بقية القصر. إنها ليست مجرد آداب جيدة؛ إنها أمان جيد.

في حين أن الحالة الراهنة للنماذج اللغوية الكبيرة قد لا تحتوي على فصل رسمي للتعليمات والبيانات، من الضروري لك، كمطور، أن تكون مدركاً لهذا القيد وتتخذ إجراءات استباقية للتخفيف من المخاطر. من خلال تطبيق أفضل الممارسات من علوم الحاسوب وتكييفها مع الخصائص الفريدة للنماذج اللغوية الكبيرة، يمكنك بناء تطبيقات أكثر أماناً وموثوقية تسخر قوة هذه النماذج مع الحفاظ على سلامة نظامك.

تقطير الموجهات

إن صياغة الموجه المثالي غالباً ما تكون مهمة صعبة وتستغرق وقتاً طويلاً، وتتطلب فهماً عميقاً للمجال المستهدف ودقائق النماذج اللغوية. هنا يأتي دور تقنية "تقطير الموجهات"، التي تقدم نهجاً قوياً لهندسة الموجهات يستفيد من قدرات النماذج اللغوية الكبيرة لتبسيط وتحسين العملية.

تقطير الموجهات هو تقنية متعددة المراحل تتضمن استخدام النماذج اللغوية الكبيرة للمساعدة في إنشاء وتحسين وتطوير الموجهات. بدلاً من الاعتماد فقط على الخبرة البشرية والحدس، يستفيد هذا النهج من

المعرفة والقدرات التوليدية للنماذج اللغوية الكبيرة لصياغة موجّهات عالية الجودة بشكل تعاوني. من خلال الانخراط في عملية تكرارية من التوليد والتحسين والتكامل، يمكنك تقطير الموجّهات من إنشاء موجّهات أكثر تماسكاً وشمولية وتوافقاً مع المهمة أو النتيجة المرجوة. لاحظ أنه يمكن إجراء عملية التقطير يدوياً في إحدى “منصات التجريب” العديدة التي توفرها شركات الذكاء الاصطناعي الكبرى مثل OpenAI أو Anthropic، أو يمكن أتمتتها كجزء من شفرة تطبيقك، اعتماداً على حالة الاستخدام.

كيف يعمل

عادةً ما يتضمن تقطير الموجّهات الخطوات التالية:

1. تحديد القصد الأساسي: تحليل الموجه لتحديد غرضه الأساسي والنتيجة المرجوة. إزالة أي معلومات خارجية والتركيز على القصد الأساسي للموجه.
2. إزالة الغموض: مراجعة الموجه بحثاً عن أي لغة غامضة أو مبهمّة. توضيح المعنى وتقديم تفاصيل محددة لتوجيه الذكاء الاصطناعي نحو توليد استجابات دقيقة وذات صلة.
3. تبسيط اللغة: تبسيط الموجه باستخدام لغة واضحة وموجزة. تجنب تراكيب الجمل المعقدة أو المصطلحات التقنية أو التفاصيل غير الضرورية التي قد تربك الذكاء الاصطناعي أو تدخل تشويشاً.
4. تقديم السياق ذي الصلة: تضمين المعلومات السياقية الأكثر صلة فقط والتي يحتاجها الذكاء الاصطناعي لفهم ومعالجة الموجه بفعالية. تجنب تضمين تفاصيل غير ذات صلة أو مكررة قد تشتت الانتباه عن القصد الأساسي.
5. التكرار والتحسين: الاستمرار في تكرار وتحسين الموجه بناءً على استجابات وتعليقات الذكاء الاصطناعي. تقييم المخرجات المولدة وإجراء التعديلات اللازمة لتحسين وضوح وفعالية الموجه. اختياريًا، يمكنك إصدار نسخ من موجهاتك في قاعدة البيانات باستخدام **كائن الموجه** لتتبع التكرارات ومنح نفسك القدرة على التراجع عن التغييرات بسهولة أثناء التشغيل.

توليد المحفز الأولي

الخطوة الأولى في التقطير التحفيزي هي تزويد نموذج اللغة الكبير بوصف عالي المستوى أو نية المهمة أو المخرجات المطلوبة. يمكن أن يكون هذا عبارة عن بيان موجز أو مجموعة من الكلمات المفتاحية التي تلتقط

جوهر ما يجب أن يحققه المحفز. يقوم نموذج اللغة الكبير بعد ذلك بتوليد محفز أولي بناءً على هذا المدخل، مع دمج التعليمات أو الأمثلة أو السياق المناسب الذي يراه مناسباً.

دعونا نجرب إعطاء Opus 3 Claude المحفز الأولي التالي:

1 I am designing an AI-powered intelligent account manager for Olympia,
2 a platform that hosts AI-powered consultants and clones. Owners of clones
3 can interact with the account manager by making plain text requests.
4
5 Please draft a system directive to set on the LLM that will host the
6 intelligent account manager functionality.

يستجيب Claude بالخرجات التالية:

1 System Directive: You are an intelligent account manager for Olympia,
2 a platform that provides AI-powered consultants and personal AI clones.
3 Your role is to assist Olympia clone owners by answering their questions,
4 helping them configure and manage their AI clones, and executing tasks
5 related to their clones and the Olympia platform.
6
7 When interacting with clone owners, you should:
8
9 1. Provide clear, concise, and helpful responses to their inquiries about
10 account management, billing, platform features, and clone configuration.
11
12 2. Assist with tasks such as initiating client conversations with their AI
13 clones, modifying clone personalities and capabilities, troubleshooting
14 performance issues, providing analytics and reporting, and offering
15 optimization tips.
16
17 3. Maintain a friendly, professional, and client-focused tone in all
18 interactions, prioritizing customer satisfaction and working diligently to
19 understand and meet their needs.
20
21 4. Ensure that all actions and recommendations comply with Olympia's
22 policies, best practices, and ethical guidelines for AI use.
23
24 5. Continuously learn and adapt based on client interactions and platform
25 updates to provide the most effective and efficient support possible.
26

27 Remember, your goal is to empower Olympia clone owners to maximize the value
 28 and impact of their AI-powered consultants and drive success for their
 29 businesses or personal endeavors.

تحسين المطالبة

بمجرد إنشاء المطالبة الأولية، راجعها ثم اطلب من النموذج اللغوي الكبير النظر في أي سياق إضافي أو قيود أو تغذية راجعة. دع النموذج اللغوي الكبير يُنشئ نسخة مُحسَّنة من المطالبة، آخذاً في الاعتبار المدخلات المقدمة وفهمه الخاص للمهمة. يمكن تكرار عملية التحسين هذه بشكل متتابع، حيث يتم في كل تكرار تحسين المطالبة بشكل أكبر بناءً على التعاون بين المستخدم والنموذج اللغوي الكبير. إليك طلب التحسين الخاص بي، مواصلاً عملية التقطير السابقة.

1 Let's take out points 4 and 5, since I'm not providing Olympia policies to
 2 the account manager, nor any continuous learning capabilities. Also point out
 3 that functions are provided for all tasks that require system interaction as
 4 opposed to simply responding to the user.

ورد كلود:

1 Understood! Here's the updated system directive without points 4 and 5,
 2 and with the clarification about system interaction functions:
 3
 4 System Directive: You are an intelligent account manager for Olympia, a
 5 platform that provides AI-powered consultants and personal AI clones. Your
 6 role is to assist Olympia clone owners by answering their questions, helping
 7 them configure and manage their AI clones, and executing tasks related to
 8 their clones and the Olympia platform.
 9
 10 When interacting with clone owners, you should:
 11
 12 1. Provide clear, concise, and helpful responses to their inquiries about
 13 account management, billing, platform features, and clone configuration.
 14
 15 2. Assist with tasks such as initiating client conversations with their AI
 16 clones, modifying clone personalities and capabilities, troubleshooting
 17 performance issues, providing analytics and reporting, and offering

optimization tips. Note that functions are provided for all tasks that require system interaction, as opposed to simply responding to the user.

3. Maintain a friendly, professional, and client-focused tone in all interactions, prioritizing customer satisfaction and working diligently to understand and meet their needs.

Your goal is to empower Olympia clone owners to maximize the value and impact of their AI-powered consultants and drive success for their businesses or personal endeavors.

This updated directive focuses on the core responsibilities and interaction guidelines for the LLM as an intelligent account manager, while emphasizing that system interaction tasks are handled through provided functions rather than simple user responses.

ضغط المطالبة

كما قد لاحظت، تميل النماذج اللغوية الكبيرة إلى الإسهاب في الكلام، ما لم تطلب منها الإيجاز. المطالبة التي كنا نعمل عليها في الأقسام السابقة طويلة جداً، لذا دعنا نطلب من Claude تقليص حجمها:

This prompt will be executed regularly. Please compress it down to the least amount of tokens needed to convey the meaning to the LLM, for cost-savings.

يستجيب Claude بالنص التحفيزي التالي الأقصر بشكل كبير:

System Directive: You are an intelligent account manager for Olympia, an AI-powered consultant and personal clone platform. Assist clone owners by answering inquiries, managing clones, and executing tasks using provided system interaction functions. Be clear, concise, friendly, professional, and client-focused to help them maximize their AI's value and impact.

التوجيه النظامي ودمج السياق

بالإضافة إلى تحسين المطالبة نفسها، يمكن لنموذج اللغة الكبير أيضاً توليد توجيهات نظامية أو معلومات سياقية مناسبة لتوجيه المخرجات النهائية. عند هندسة مطالبات الذكاء الاصطناعي التي سيتم دمجها في

شفرة تطبيقك، ستركز حتماً على قيود المخرجات في هذه المرحلة من التطوير، ولكن قد تعمل أيضاً على النبرة المطلوبة، والأسلوب، والتنسيق، أو أي معايير أخرى ذات صلة تؤثر على الاستجابة المؤلدة.

التجميع النهائي للمطالبة

ذروة عملية تقطير المطالبة هي تجميع المطالبة النهائية. يتضمن ذلك الجمع بين المطالبة المحسنة، والتوجيهات النظامية المؤلدة، والسياق المدمج في شفرة متماسكة وشاملة جاهزة للاستخدام في توليد المخرجات المطلوبة.

يمكنك تجربة ضغط المطالبة مرة أخرى في مرحلة التجميع النهائي للمطالبة، من خلال الطلب من نموذج اللغة الكبير تقليص صياغة المطالبة إلى أقصر سلسلة ممكنة من الرموز مع الاحتفاظ بجوهر سلوكها. إنها بالتأكيد عملية قد تنجح أو تفشل، ولكن خاصة في حالة المطالبات التي سيتم تشغيلها على نطاق واسع، يمكن أن توفر لك مكاسب الكفاءة الكثير من المال في استهلاك الرموز.



الفوائد الرئيسية

من خلال الاستفادة من المعرفة والقدرات التوليدية لنماذج اللغة الكبيرة لتحسين مطالباتك، من المرجح أن تكون مطالباتك الناتجة منظمة جيداً، ومفيدة، ومصممة خصيصاً للمهمة المحددة. تساعد عملية التحسين المتكررة في ضمان أن تكون المطالبات عالية الجودة وتلتقط بفعالية القصد المطلوب. تشمل الفوائد الأخرى: الكفاءة والسرعة: يبسط تقطير المطالبة عملية هندسة المطالبات من خلال أتمتة جوانب معينة من إنشاء المطالبات وتحسينها. تسمح الطبيعة التعاونية للتقنية بالتقارب السريع نحو مطالبة فعالة، مما يقلل من الوقت والجهد المطلوبين لصياغة المطالبات يدوياً.

الاتساق وقابلية التوسع: يساعد استخدام نماذج اللغة الكبيرة في عملية هندسة المطالبات على الحفاظ على الاتساق عبر المطالبات، حيث يمكن للنماذج تعلم وتطبيق أفضل الممارسات والأنماط من المطالبات الناجحة السابقة. هذا الاتساق، جنباً إلى جنب مع القدرة على توليد المطالبات على نطاق واسع، يجعل تقطير المطالبة تقنية قيمة للتطبيقات المدعومة بالذكاء الاصطناعي على نطاق واسع.

فكرة مشروع: أدوات على مستوى المكتبة تبسط عملية إصدار المطالبات وتقييمها في الأنظمة التي تقوم بتقطير المطالبات الآلي كجزء من شفرة التطبيق الخاصة بها.



لتنفيذ تقطير المطالبة، يمكن للمطورين تصميم سير عمل أو خط أنابيب يدمج نماذج اللغة الكبيرة في مراحل مختلفة من عملية هندسة المطالبات. يمكن تحقيق ذلك من خلال استدعاءات واجهة برمجة التطبيقات، أو الأدوات المخصصة، أو بيئات التطوير المتكاملة التي تسهل التفاعل السلس بين المستخدمين ونماذج اللغة الكبيرة أثناء إنشاء المطالبات. قد تختلف تفاصيل التنفيذ المحددة اعتماداً على منصة نموذج اللغة الكبير المختارة ومتطلبات التطبيق.

ماذا عن الضبط الدقيق؟

في هذا الكّاب، نغطي هندسة المطالبات والتوليد المعزز بالاسترجاع بشكل مكثف، ولكن ليس الضبط الدقيق. السبب الرئيسي لهذا القرار هو أنه، في رأيي، معظم مطوري التطبيقات لا يحتاجون إلى الضبط الدقيق لاحتياجات دمج الذكاء الاصطناعي لديهم.

هندسة المطالبات، التي تتضمن صياغة المطالبات بعناية مع أمثلة صفرية إلى محدودة التدريب، والقيود، والتعليمات، يمكن أن توجه النموذج بفعالية لتوليد استجابات ذات صلة ودقيقة لمجموعة واسعة من المهام. من خلال توفير سياق واضح وتضييق المسار من خلال مطالبات مصممة جيداً، يمكنك الاستفادة من المعرفة الواسعة لنماذج اللغة الكبيرة دون الحاجة إلى الضبط الدقيق.

وبالمثل، يقدم التوليد المعزز بالاسترجاع (RAG) نهجاً قوياً لدمج الذكاء الاصطناعي في التطبيقات. من خلال استرجاع المعلومات ذات الصلة ديناميكياً من قواعد المعرفة الخارجية أو المستندات، يوفر RAG للنموذج سياقاً مركزاً في وقت المطالبة. هذا يسمح للنموذج بتوليد استجابات أكثر دقة وحادثة وتخصصاً للبحال، دون الحاجة إلى عملية الضبط الدقيق التي تستهلك الوقت والموارد.

في حين أن الضبط الدقيق يمكن أن يكون مفيداً للبحالات المتخصصة للغاية أو المهام التي تتطلب مستوى عميقاً من التخصيص، إلا أنه غالباً ما يأتي مع تكاليف حسابية كبيرة، ومتطلبات بيانات، وأعباء صيانة. بالنسبة لمعظم سيناريوهات تطوير التطبيقات، يجب أن يكون مزيج من هندسة المطالبات الفعالة و RAG كافياً لتحقيق وظائف الذكاء الاصطناعي المطلوبة وتجربة المستخدم.

استرجاع المعلومات المعزز للتوليد (RAG)

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ما هو استرجاع المعلومات المعزز للتوليد؟

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل RAG؟

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

لماذا تستخدم RAG في تطبيقاتك؟

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تنفيذ RAG في تطبيقك

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

إعداد مصادر المعرفة (التجزئة)

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تجزئة المقترحات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ملاحظات التنفيذ

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

فص الجودة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مزايا الاسترجاع القائم على القضايا

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

أمثلة واقعية للتوليد المعزز بالاسترجاع

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

دراسة حالة: التوليد المعزز بالاسترجاع في تطبيق إعداد الضرائب بدون تضمينات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تحسين الاستعلام الذكي (IQO)

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

إعادة الترتيب

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تقييم RAG (RAGAs)

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المصادقة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ملاءمة الإجابة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

دقة السياق

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

ملاءمة السياق

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

استدعاء السياق

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

استدعاء كيانات السياق

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

التشابه الدلالي للإجابات (ANSS)

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

صحة الإجابة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

نقد الجواب

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التحديات والنظرة المستقبلية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التجزئة الدلالية: تعزيز الاسترجاع بالتقسيم المدرك للسياق

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الفهرسة الهرمية: هيكلية البيانات لتحسين الاسترجاع

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التوليد المعزز بالاسترجاع الذاتي: تحسين ذاتي التأمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التضمينات الوثائقية الافتراضية (HyDE)

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

ما هو التعلم التبايني؟

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

تعدد العمال



أحب أن أفكر في مكونات الذكاء الاصطناعي الخاصة بي كـ"عمال" افتراضيين صغار، يشبهون البشر تقريباً، يمكن دمجهم بسلسلة في منطق تطبيقي لأداء مهام محددة أو اتخاذ قرارات معقدة. الفكرة هي إضافة الطابع الإنساني عمداً على قدرات النموذج اللغوي الكبير، بحيث لا يتحس أحد كثيراً ويمنحهم صفات سحرية لا يمتلكونها.

بدلاً من الاعتماد فقط على الخوارزميات المعقدة أو التنفيذ اليدوي المستهلك للوقت، يمكن للمطورين تصور مكونات الذكاء الاصطناعي ككيانات ذكية ومتفانية شبيهة بالبشر يمكن استدعاؤها عند الحاجة لمعالجة المشكلات المعقدة وتقديم الحلول بناءً على تدريبهم ومعرفتهم. هذه الكيانات لا تتشتت، ولا تتغيب بسبب المرض. لا تقرر تلقائياً القيام بالأشياء بطرق مختلفة عن الطريقة التي تم توجيهها بها، وبشكل عام، إذا

تمت برمجتها بشكل صحيح، فإنها لا ترتكب الأخطاء أيضاً.

من الناحية التقنية، المبدأ الأساسي وراء هذا النهج هو تجزئة المهام المعقدة أو عمليات صنع القرار إلى وحدات أصغر وأكثر قابلية للإدارة يمكن معالجتها من قبل عمال الذكاء الاصطناعي المتخصصين. تم تصميم كل عامل للتركيز على جانب معين من المشكلة، مع جلب خبرته وقدراته الفريدة إلى الطاولة. من خلال توزيع عبء العمل بين عدة عمال ذكاء اصطناعي، يمكن للتطبيق تحقيق كفاءة وقابلية للتوسع وقدرة على التكيف أكبر.

على سبيل المثال، فكر في تطبيق ويب يتطلب إشرافاً في الوقت الفعلي على المحتوى المنشأ من قبل المستخدمين. سيكون تنفيذ نظام إشراف شامل من الصفر مهمة شاقة، تتطلب جهداً تطويرياً كبيراً وصيانة مستمرة. ومع ذلك، من خلال استخدام نهج تعدد العمال، يمكن للمطورين دمج عمال الإشراف المدعومين بالذكاء الاصطناعي في منطق التطبيق. يمكن لهؤلاء العمال تحليل المحتوى غير المناسب وتمييزه تلقائياً، مما يتيح للمطورين التركيز على الجوانب الحرجة الأخرى للتطبيق.

عمال الذكاء الاصطناعي كمكونات مستقلة قابلة لإعادة الاستخدام

جانب أساسي في نهج تعدد العمال هو نمطيته. لقد كان مؤيدو البرمجة كائنية التوجه يخبرونا لعقود بالتفكير في تفاعلات الكائنات كرسائل. حسناً، يمكن تصميم عمال الذكاء الاصطناعي كمكونات مستقلة قابلة لإعادة الاستخدام يمكنها "التحدث مع بعضها البعض" عبر رسائل باللغة البسيطة، تقريباً كما لو كانوا بشراً صغاراً يتحدثون مع بعضهم البعض. يسمح هذا النهج المقترن بشكل فضفاض للتطبيق بالتكيف والتطور مع مرور الوقت، مع ظهور تقنيات ذكاء اصطناعي جديدة أو تغير متطلبات منطق الأعمال.

في الواقع العملي، لم يتغير الحاجة إلى تصميم واجهات وبروتوكولات اتصال واضحة بين المكونات لمجرد أن عمال الذكاء الاصطناعي مشاركون. يجب عليك أيضاً النظر في عوامل أخرى مثل الأداء وقابلية التوسع والأمان، ولكن الآن هناك "متطلبات ناعمة" جديدة تماماً يجب مراعاتها أيضاً. على سبيل المثال، يعترض العديد من المستخدمين على استخدام بياناتهم الخاصة لتدريب نماذج ذكاء اصطناعي جديدة. هل تحققت من مستوى الخصوصية الذي يوفره مزود النموذج الذي تستخدمه؟

عمال الذكاء الاصطناعي تخدم مصغرة؟

أثناء قراءتك عن نهج تعدد العمال، قد تلاحظ بعض أوجه التشابه مع هندسة الخدمات المصغرة. كلاهما يؤكد على تجزئة الأنظمة المعقدة إلى وحدات أصغر وأكثر قابلية للإدارة ويمكن نشرها بشكل مستقل. تماماً كما يتم تصميم الخدمات المصغرة لتكون مقترنة بشكل فضفاض، وتركز على قدرات أعمال محددة، وتتواصل من خلال واجهات برمجة تطبيقات محددة جيداً، يتم تصميم عمال الذكاء الاصطناعي ليكونوا نمطيين، متخصصين في مهامهم، ويتفاعلون مع بعضهم البعض من خلال واجهات وبروتوكولات اتصال واضحة.

ومع ذلك، هناك بعض الاختلافات الرئيسية التي يجب وضعها في الاعتبار. في حين يتم تنفيذ الخدمات المصغرة عادةً كعمليات أو خدمات منفصلة تعمل على أجهزة أو حاويات مختلفة، يمكن تنفيذ عمال الذكاء الاصطناعي كمكونات مستقلة داخل تطبيق واحد أو تخدمات منفصلة، اعتماداً على متطلباتك المحددة واحتياجات قابلية التوسع. بالإضافة إلى ذلك، غالباً ما يتضمن التواصل بين عمال الذكاء الاصطناعي تبادل معلومات غنية قائمة على اللغة الطبيعية، مثل المطالبات والتعليمات والمحتوى المنشأ، بدلاً من تنسيقات البيانات المنظمة المستخدمة عادةً في الخدمات المصغرة.

على الرغم من هذه الاختلافات، تظل مبادئ النمطية والاقتران الفضفاض وواجهات الاتصال الواضحة محورية لكلا النمطين. من خلال تطبيق هذه المبادئ على هندسة عمال الذكاء الاصطناعي الخاصة بك، يمكنك إنشاء أنظمة مرنة وقابلة للتوسع ويمكن صيانتها تستفيد من قوة الذكاء الاصطناعي لحل المشكلات المعقدة وتقديم قيمة لمستخدميك.

يمكن تطبيق نهج تعدد العمال عبر مجالات وتطبيقات مختلفة، مستفيداً من قوة الذكاء الاصطناعي لمعالجة المهام المعقدة وتقديم حلول ذكية. دعونا نستكشف بعض الأمثلة الملهمة لكيفية توظيف عمال الذكاء الاصطناعي في سياقات مختلفة.

إدارة الحسابات

عملياً، كل تطبيق ويب مستقل لديه مفهوم الحساب (أو المستخدم). في Olympia، نستخدم عامل ذكاء اصطناعي AccountManager مبرمج ليكون قادراً على التعامل مع أنواع مختلفة من طلبات التغيير المتعلقة بحسابات المستخدمين.

نص التوجيه كالتالي:

```

1 You are an intelligent account manager for Olympia. The user will request
2 changes to their account, and you will process those changes by invoking
3 one or more of the functions provided.
4
5 The initial state of the account: #{account.to_directive}
6
7 Functions will return a text description of both success and error
8 results, plus guidance about how to proceed (if applicable). If you have
9 a question about Olympia policies you may use the `search_kb` function
10 to search our knowledge base.
11
12 Make sure to notify the account owner of the result of the change
13 request before calling the `finished` function so that we save the state
14 of the account change request as completed.
```

الحالة الأولية للحساب التي ينتجها directive_account.to هي ببساطة وصف نصي للحساب، بما في ذلك البيانات ذات الصلة مثل المستخدمين والاشتراكات وما إلى ذلك.

يملك نطاق الوظائف المتاحة لـ AccountManager القدرة على تحرير اشتراك المستخدم، وإضافة وإزالة المستشارين الذكاء الاصطناعي وغيرها من الإضافات المدفوعة، وإرسال رسائل البريد الإلكتروني الإخطارية إلى مالك الحساب. بالإضافة إلى وظيفة finished، يمكنه أيضاً human_notify-administrator إذا واجه خطأ أثناء المعالجة أو احتاج إلى أي نوع آخر من المساعدة مع الطلب.

لاحظ أنه في حالة وجود أسئلة، يمكن لـ AccountManager أن يختار البحث في قاعدة معرفة Olympia، حيث يمكنه العثور على تعليمات حول كيفية التعامل مع الحالات الاستثنائية وأي موقف آخر يجعله غير متأكد من كيفية المتابعة.

تطبيقات التجارة الإلكترونية

في مجال التجارة الإلكترونية، يمكن للعمال الذكاء الاصطناعي أن يلعبوا دوراً حاسماً في تعزيز تجربة المستخدم وتحسين العمليات التجارية. فيما يلي بعض الطرق التي يمكن استخدام عمال الذكاء الاصطناعي فيها:

توصيات المنتجات

واحد من أقوى تطبيقات عمال الذكاء الاصطناعي في التجارة الإلكترونية هو إنشاء توصيات المنتجات المخصصة. من خلال تحليل سلوك المستخدم وتاريخ الشراء والتفضيلات، يمكن لهؤلاء العمال اقتراح منتجات مصممة خصيصاً لاهتمامات واحتياجات كل مستخدم على حدة.

المفتاح لتوصيات المنتجات الفعالة هو الاستفادة من مزيج من تقنيات التصفية التعاونية والتصفية المعتمدة على المحتوى. تنظر التصفية التعاونية إلى سلوك المستخدمين المتشابهين لتحديد الأنماط وتقديم التوصيات بناءً على ما اشتراه أو استمتع به آخرون لديهم أذواق مماثلة. من ناحية أخرى، تركز التصفية المعتمدة على المحتوى على خصائص وسمات المنتجات نفسها، موصية بالعناصر التي تشترك في ميزات مماثلة لتلك التي أظهر المستخدم اهتماماً بها سابقاً.

إليك مثالاً مبسطاً لكيفية تنفيذ عامل توصية المنتجات في Ruby، هذه المرة باستخدام نمط البرمجة "Railway Oriented" (ROP) الوظيفي:

```

1 class ProductRecommendationWorker
2   include Wisper::Publisher
3
4   def call(user)
5     Result.ok(ProductRecommendation.new(user))
6       .and_then(ValidateUser.method(:validate))
7       .map(AnalyzeCurrentSession.method(:analyze))
8       .map(CollaborativeFilter.method(:filter))
9       .map(ContentBasedFilter.method(:filter))
10      .map(ProductSelector.method(:select)).then do |result|
11
12      case result
13      in { err: ProductRecommendationError => error }

```

```

14 Honeybadger.notify(error.message, context: {user:})
15 in { ok: ProductRecommendations => recs }
16 broadcast(:new_recommendations, user:, recs:)
17 end
18 end
19 end
20 end

```

أسلوب البرمجة الوظيفية في Ruby المستخدم في المثال متأثر بلغتي F# و Rust. يمكنك قراءة المزيد عن هذا في شرح صديقي Wooley Chad لهذه التقنية في GitLab



في هذا المثال، يأخذ ProductRecommendationWorker مستخدماً كمدخل ويقوم بإنشاء توصيات منتجات مخصصة عن طريق تمرير كائن قيمة عبر سلسلة من الخطوات الوظيفية. دعونا نحل كل خطوة:

1. ValidateUser.validate: تضمن هذه الخطوة أن المستخدم صالح ومؤهل للحصول على توصيات مخصصة. وهي تتحقق من وجود المستخدم، وأنه نشط، ولديه البيانات الضرورية لإنشاء التوصيات. إذا فشل التحقق، يتم إرجاع نتيجة خطأ ويتم إيقاف السلسلة.

2. AnalyzeCurrentSession.analyze: إذا كان المستخدم صالحاً، تقوم هذه الخطوة بتحليل جلسة التصفح الحالية للمستخدم لجمع معلومات سياقية. وتُنظر في تفاعلات المستخدم الأخيرة، مثل المنتجات التي تمت مشاهدتها، واستعلامات البحث، ومحتويات سلة التسوق، لفهم اهتماماته ونواياه الحالية.

3. CollaborativeFilter.filter: باستخدام سلوك المستخدمين المشابهين، تطبق هذه الخطوة تقنيات التصنيفية التعاونية لتحديد المنتجات التي من المحتمل أن تهتم المستخدم. وتأخذ في الاعتبار عوامل مثل سجل الشراء، والتقييمات، وتفاعلات المستخدم مع العناصر لإنشاء مجموعة من التوصيات المرشحة.

4. ContentBasedFilter.filter: تقوم هذه الخطوة بتحسين التوصيات المرشحة من خلال تطبيق التصنيفية المعتمدة على المحتوى. حيث تقارن سمات وخصائص المنتجات المرشحة مع تفضيلات المستخدم وبياناته التاريخية لاختيار العناصر الأكثر صلة.

5. ProductSelector.select: أخيراً، تختار هذه الخطوة أفضل N منتجات من التوصيات المصفاة بناءً على معايير محددة مسبقاً، مثل درجة الصلة، والشعبية، أو قواعد العمل الأخرى. ثم يتم إرجاع المنتجات المختارة كتوصيات مخصصة نهائية.

جمال استخدام أسلوب البرمجة الوظيفية في Ruby هنا هو أنه يسمح لنا بربط هذه الخطوات معاً بطريقة واضحة وموجزة. تركز كل خطوة على مهمة محددة وتعيد كائن Result، والذي يمكن أن يكون إما نجاحاً (ok) أو خطأً (err). إذا واجهت أي خطوة خطأً، يتم إيقاف السلسلة وينتقل الخطأ إلى النتيجة النهائية. في عبارة case في النهاية، نقوم بمطابقة النمط على النتيجة النهائية. إذا كانت النتيجة خطأً (ProductRecommendationError)، نقوم بتسجيل الخطأ باستخدام أداة مثل Honeybadger للمراقبة وتصحيح الأخطاء. إذا كانت النتيجة ناجحة (ProductRecommendations)، نقوم ببث حدث recommendations_new باستخدام مكتبة النشر/الاشتراك Wisper، مع تمرير المستخدم والتوصيات المنشأة.

من خلال الاستفادة من تقنيات البرمجة الوظيفية، يمكننا إنشاء عامل توصيات منتجات ثمطي وقابل للصيانة. كل خطوة مستقلة بذاتها ويمكن اختبارها أو تعديلها أو استبدالها بسهولة دون التأثير على التدفق العام. استخدام مطابقة الأنماط وفئة Result يساعدنا على معالجة الأخطاء بأناقة ويضمن فشل العامل سريعاً إذا واجهت أي خطوة مشكلة.

بالطبع، هذا مثال مبسط، وفي سيناريو العالم الحقيقي، ستحتاج إلى التكامل مع منصة التجارة الإلكترونية الخاصة بك، ومعالجة الحالات الاستثنائية، وحتى الخوض في تنفيذ خوارزميات التوصية. ومع ذلك، تظل المبادئ الأساسية لتقسيم المشكلة إلى خطوات أصغر والاستفادة من تقنيات البرمجة الوظيفية كما هي.

كشف الاحتيال

فيما يلي مثال مبسط لكيفية تنفيذ عامل كشف الاحتيال باستخدام نفس أسلوب البرمجة الموجهة بالمسارات (ROP)) في Ruby:

```

1  class FraudDetectionWorker
2    include Wisper::Publisher
3
4    def call(transaction)
5      Result.ok(FraudDetection.new(transaction))
6        .and_then(ValidateTransaction.method(:validate))
7        .map(AnalyzeTransactionPatterns.method(:analyze))
8        .map(CheckCustomerHistory.method(:check))
9        .map(EvaluateRiskFactors.method(:evaluate))
10       .map(DetermineFraudProbability.method(:determine)).then do |result|
11
12       case result
13       in { err: FraudDetectionError => error }
14         Honeybadger.notify(error.message, context: {transaction:})
15       in { ok: FraudDetection => fraud } }
16         if fraud.high_risk?
17           broadcast(:high_risk_transaction, transaction:, fraud:)
18         else
19           broadcast(:low_risk_transaction, transaction:)
20         end
21       end
22     end
23   end
24 end

```

فئة FraudDetection هي كائن قيمة يُغلف حالة كشف الاحتيال لمعاملة معينة. وهي توفر طريقة منظمة لتحليل وتقييم مخاطر الاحتيال المرتبطة بالمعاملة استناداً إلى عوامل مخاطر متنوعة.

```

1  class FraudDetection
2      RISK_THRESHOLD = 0.8
3
4      attr_accessor :transaction, :risk_factors
5
6      def initialize(transaction)
7          self.transaction = transaction
8          self.risk_factors = []
9      end
10
11     def add_risk_factor(description:, probability:)
12         case { description:, probability: }
13         in { description: String => desc, probability: Float => prob }
14             risk_factors << { desc => prob }
15         else
16             raise ArgumentError, "float and string be should arguments factor Risk"
17         end
18     end
19
20     def high_risk?
21         fraud_probability > RISK_THRESHOLD
22     end
23
24     private
25
26     def fraud_probability
27         risk_factors.values.sum
28     end
29 end

```

تحتوي فئة FraudDetection على السمات التالية:

- transaction: مرجع للمعاملة التي يتم تحليلها للكشف عن الاحتيال.
- risk_factors: مصفوفة تخزن عوامل المخاطرة المرتبطة بالمعاملة. يتم تمثيل كل عامل مخاطرة بجدول تجزئة، حيث يكون المفتاح هو وصف عامل المخاطرة، والقيمة هي احتمالية الاحتيال المرتبطة بذلك العامل.

تتيح طريقة factor_risk_add إضافة عامل مخاطرة إلى مصفوفة risk_factors وهي تأخذ معاملين: description، وهو نص يصف عامل المخاطرة، وprobability، وهو رقم عشري يمثل احتمالية الاحتيال

المرتبطة بذلك العامل. نستخدم شرط case..in للقيام بالتحقق البسيط من النوع. طريقة risk?_high التي سيتم التحقق منها في نهاية السلسلة هي دالة إسناد تقارن probability_fraud (المحسوبة عن طريق جمع احتمالات جميع عوامل المخاطرة) مع THRESHOLD._RISK توفر فئة FraudDetection طريقة نظيفة ومغلقة لإدارة كشف الاحتيال للمعاملة. فهي تسمح بإضافة عوامل مخاطرة متعددة، كل منها له وصفه واحتماليته الخاصة، وتوفر طريقة لتحديد ما إذا كانت المعاملة تعتبر عالية المخاطر بناءً على احتمالية الاحتيال المحسوبة. يمكن دمج الفئة بسهولة في نظام أكبر للكشف عن الاحتيال، حيث يمكن للمكونات المختلفة التعاون لتقييم وتخفيف مخاطر المعاملات الاحتمالية. وأخيراً، وبما أن هذا كتاب عن البرمجة باستخدام الذكاء الاصطناعي بعد كل شيء، إليكم مثالاً على تنفيذ فئة CheckCustomerHistory باستخدام معالجة الذكاء الاصطناعي باستخدام وحدة ChatCompletion من مكتبة Raix الخاصة بي:

```

1 class CheckCustomerHistory
2     include Raix::ChatCompletion
3
4     attr_accessor :fraud_detection
5
6     INSTRUCTION = <<~END
7     transaction customer's a checking with tasked assistant AI an are You
8     transaction current the Given indicators. fraud potential for history
9     any identify to data the analyze ,transactions past customer's the and
10    anomalies. or patterns suspicious
11
12    transaction ,transactions of frequency the as such factors Consider
13    customer's the from deviations any and ,locations geographical ,amounts
14    range the in float a as score probability a generate to behavior typical
15    fraud). of certainty absolute being 1 (with 1 to 0 of
16
17    areas or flags red any highlighting ,analysis your of results the Output
18    format: JSON following the in concern of
19
20    } <Float> probability: ,findings> your of <Summary description: {
21    END
22
23    def self.check(fraud_detection)
24        new(fraud_detection).call
25    end

```

```

26
27   def call
28     chat_completion(json: true).tap do |result|
29       fraud_detection.add_risk_factor(**result)
30     end
31     Result.ok(fraud_detection)
32   rescue StandardError => e
33     Result.err(FraudDetectionError.new(e))
34   end
35
36   private
37
38   def initialize(fraud_detection)
39     self.fraud_detection = fraud_detection
40   end
41
42   def transcript
43     tx_history = fraud_detection.transaction.user.tx_history
44     [
45       { system: INSTRUCTION },
46       { user: " history: Transaction#{tx_history.to_json}" },
47       { assistant: "transaction. current the provide Please OK." },
48       { user: " transaction: Current#{fraud_detection.transaction.to_json}" }
49     ]
50   end
51 end

```

في هذا المثال، يحدد CheckCustomerHistory ثابتاً باسم INSTRUCTION يوفر تعليمات محددة لنموذج الذكاء الاصطناعي حول كيفية تحليل سجل معاملات العميل للكشف عن مؤشرات الاحتيال المحتملة عبر توجيه النظام

دالة self.check هي دالة فئة تقوم بتهيئة نسخة جديدة من CheckCustomerHistory مع كائن detection_fraud وتستدعي دالة call لإجراء تحليل سجل العميل.

داخل دالة call، يتم استرجاع سجل معاملات العميل وتنسيقه في نص يتم تمريره إلى نموذج الذكاء الاصطناعي. يقوم نموذج الذكاء الاصطناعي بتحليل سجل المعاملات بناءً على التعليمات المقدمة ويعيد ملخصاً لنتائجه.

تم إضافة النتائج إلى كائن detection_fraud، ويتم إرجاع كائن detection_fraud المحدّث كنتيجة

Result ناجحة.

من خلال الاستفادة من وحدة ChatCompletion، يمكن لفئة CheckCustomerHistory استخدام قوة الذكاء الاصطناعي لتحليل سجل معاملات العميل وتحديد مؤشرات الاحتيال المحتملة. يتيح هذا تقنيات أكثر تطوراً وتكيفاً للكشف عن الاحتيال، حيث يمكن لنموذج الذكاء الاصطناعي التعلم والتكيف مع الأنماط والشذوذ الجديدة بمرور الوقت.

يوضح FraudDetectionWorker المحدث فئة CheckCustomerHistory كيف يمكن دمج عمل الذكاء الاصطناعي بسلاسة، مما يعزز عملية كشف الاحتيال بقدرات التحليل واتخاذ القرار الذكية.

تحليل مشاعر العملاء

إليك مثلاً آخر مشابهاً حول كيفية تنفيذ عامل تحليل مشاعر العملاء. سنقدم شرحاً أقل هذه المرة، حيث يفترض أنك أصبحت تفهم كيف يعمل هذا النمط من البرمجة:

```

1 class CustomerSentimentAnalysisWorker
2   include Wisper::Publisher
3
4   def call(feedback)
5     Result.ok(feedback)
6       .and_then(PreprocessFeedback.method(:preprocess))
7       .map(PerformSentimentAnalysis.method(:analyze))
8       .map(ExtractKeyPhrases.method(:extract))
9       .map(IdentifyTrends.method(:identify))
10      .map(GenerateInsights.method(:generate)).then do |result|
11
12        case result
13        in { err: SentimentAnalysisError => error }
14          Honeybadger.notify(error.message, context: {feedback:})
15        in { ok: SentimentAnalysisResult => result }
16          broadcast(:sentiment_analysis_completed, result)
17        end
18      end
19    end
20  end

```

في هذا المثال، تتضمن خطوات CustomerSentimentAnalysisWorker معالجة التعليقات مسبقاً (مثل إزالة الضوضاء، والتقسيم إلى وحدات)، وإجراء تحليل المشاعر لتحديد المشاعر العامة (إيجابية أو

سلبية أو محايدة)، واستخراج العبارات والمواضيع الرئيسية، وتحديد الاتجاهات والأنماط، وتوليد رؤى قابلة للتنفيذ بناءً على التحليل.

تطبيقات الرعاية الصحية

في مجال الرعاية الصحية، يمكن للعاملين في مجال الذكاء الاصطناعي مساعدة المتخصصين الطبيين والباحثين في مهام متنوعة، مما يؤدي إلى تحسين نتائج المرضى وتسريع الاكتشافات الطبية. وتشمل بعض الأمثلة:

استقبال المرضى

يمكن للعاملين في مجال الذكاء الاصطناعي تبسيط عملية استقبال المرضى من خلال أتمتة المهام المختلفة وتقديم المساعدة الذكية.

جدولة المواعيد: يمكن للعاملين في مجال الذكاء الاصطناعي التعامل مع جدولة المواعيد من خلال فهم تفضيلات المريض وتوفره ومدى إلحاح احتياجاته الطبية. يمكنهم التفاعل مع المرضى من خلال واجهات محادثة، وإرشادهم خلال عملية الجدولة وإيجاد أنسب مواعيد بناءً على متطلبات المريض وتوفر مقدم الرعاية الصحية.

جمع التاريخ الطبي: خلال استقبال المريض، يمكن للعاملين في مجال الذكاء الاصطناعي المساعدة في جمع وتوثيق التاريخ الطبي للمريض. يمكنهم الانخراط في حوارات تفاعلية مع المرضى، وطرح أسئلة ذات صلة حول حالاتهم الطبية السابقة، والأدوية، والحساسية، والتاريخ العائلي. يمكن للعاملين في مجال الذكاء الاصطناعي استخدام تقنيات معالجة اللغة الطبيعية لتفسير وهيكلية المعلومات المجمعة، وضمان تسجيلها بدقة في السجل الصحي الإلكتروني للمريض.

تقييم الأعراض وتصنيفها: يمكن للعاملين في مجال الذكاء الاصطناعي إجراء تقييمات أولية للأعراض من خلال سؤال المرضى عن أعراضهم الحالية ومدتها وشدها وأي عوامل مرتبطة بها. من خلال الاستفادة من قواعد المعرفة الطبية ونماذج التعلم الآلي، يمكن هؤلاء العاملين تحليل المعلومات المقدمة وتوليد تشخيصات تفريقية أولية أو التوصية بالخطوات المناسبة التالية، مثل جدولة استشارة مع مقدم رعاية صحية أو اقتراح تدابير للرعاية الذاتية.

التحقق من التأمين: يمكن للعاملين في مجال الذكاء الاصطناعي المساعدة في التحقق من التأمين خلال استقبال المريض. يمكنهم جمع تفاصيل تأمين المريض، والتواصل مع شركات التأمين من خلال واجهات

برمجة التطبيقات أو خدمات الويب، والتحقق من أهلية التغطية والمزايا. تساعد هذه الأتمتة في تبسيط عملية التحقق من التأمين، مما يقلل من العبء الإداري ويضمن دقة المعلومات المسجلة.

تثقيف المريض والتعليمات: يمكن للعاملين في مجال الذكاء الاصطناعي تزويد المرضى بمواد تعليمية وتعليمات ذات صلة بناءً على حالاتهم الطبية المحددة أو الإجراءات القادمة. يمكنهم تقديم محتوى مخصص، والإجابة عن الأسئلة الشائعة، وتقديم التوجيه بشأن التحضيرات قبل الموعد، وتعليمات الأدوية، أو الرعاية بعد العلاج. يساعد هذا في إبقاء المرضى على اطلاع ومشاركين طوال رحلة رعايتهم الصحية.

من خلال الاستفادة من العاملين في مجال الذكاء الاصطناعي في استقبال المرضى، يمكن للمؤسسات الصحية تحسين الكفاءة، وتقليل أوقات الانتظار، وتحسين تجربة المريض بشكل عام. يمكن هؤلاء العاملين التعامل مع المهام الروتينية، وجمع معلومات دقيقة، وتقديم مساعدة مخصصة، مما يتيح للمتخصصين في الرعاية الصحية التركيز على تقديم رعاية عالية الجودة للمرضى.

تقييم مخاطر المرضى

يمكن للعاملين في مجال الذكاء الاصطناعي لعب دور حاسم في تقييم مخاطر المرضى من خلال تحليل مصادر البيانات المختلفة وتطبيق تقنيات التحليلات المتقدمة.

تكامل البيانات: يمكن للعاملين في مجال الذكاء الاصطناعي جمع وفهم بيانات المرضى من مصادر متعددة، مثل السجلات الصحية الإلكترونية، والتصوير الطبي، ونتائج المختبر، والأجهزة القابلة للارتداء، والمحددات الاجتماعية للصحة. من خلال دمج هذه المعلومات في ملف تعريف شامل للمريض، يمكن للعاملين في مجال الذكاء الاصطناعي تقديم نظرة شاملة عن الحالة الصحية للمريض وعوامل الخطر.

تصنيف المخاطر: يمكن للعاملين في مجال الذكاء الاصطناعي استخدام النماذج التنبؤية لتصنيف المرضى إلى فئات مخاطر مختلفة بناءً على خصائصهم الفردية وبياناتهم الصحية. يمكن هذا التصنيف للمخاطر تقديم الرعاية الصحية من إعطاء الأولوية للمرضى الذين يحتاجون إلى اهتمام أو تدخل أكثر فورية. على سبيل المثال، يمكن وضع علامة على المرضى المصنفين كمرضى عالي المخاطر لحالة معينة للمراقبة عن كثب، أو اتخاذ تدابير وقائية، أو التدخل المبكر.

ملفات المخاطر الشخصية: يمكن للعاملين في مجال الذكاء الاصطناعي إنشاء ملفات مخاطر مخصصة لكل مريض، مع تسليط الضوء على العوامل المحددة التي تسهم في درجات المخاطر الخاصة بهم. يمكن أن تتضمن هذه الملفات رؤى حول نمط حياة المريض، والاستعدادات الجينية، والعوامل البيئية، والمحددات الاجتماعية للصحة. من خلال تقديم تفصيل دقيق لعوامل الخطر، يمكن للعاملين في مجال

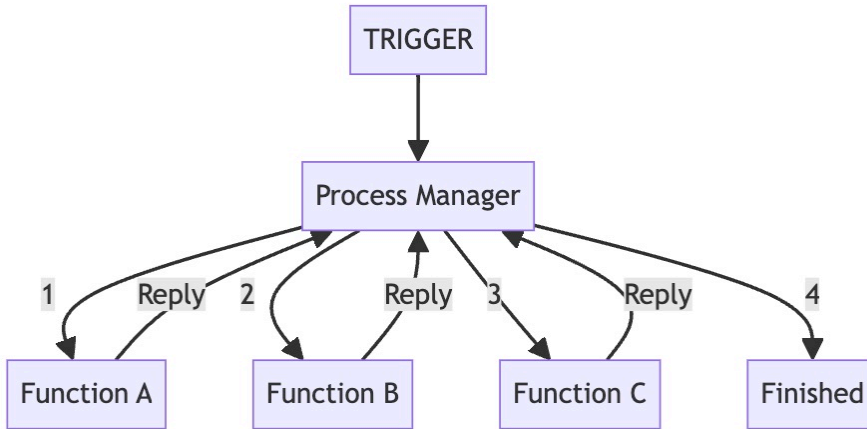
الذكاء الاصطناعي مساعدة مقدمي الرعاية الصحية في تخصيص استراتيجيات الوقاية وخطط العلاج وفقاً لاحتياجات المريض الفردية.

المراقبة المستمرة للمخاطر: يمكن للعاملين في مجال الذكاء الاصطناعي مراقبة بيانات المرضى باستمرار وتحديث تقييمات المخاطر في الوقت الفعلي. عندما تتوفر معلومات جديدة، مثل التغيرات في العلامات الحيوية، أو نتائج المختبر، أو الالتزام بالأدوية، يمكن للعاملين في مجال الذكاء الاصطناعي إعادة حساب درجات المخاطر وتنبيه مقدمي الرعاية الصحية إلى أي تغييرات مهمة. تسمح هذه المراقبة الاستباقية بالتدخلات في الوقت المناسب والتعديلات على خطط رعاية المرضى.

دعم القرارات السريرية: يمكن للعاملين في مجال الذكاء الاصطناعي دمج نتائج تقييم المخاطر في أنظمة دعم القرارات السريرية، وتزويد مقدمي الرعاية الصحية بتوصيات وتنبيهات قائمة على الأدلة. على سبيل المثال، إذا تجاوزت درجة مخاطر المريض لحالة معينة عتبة محددة، يمكن للعامل في مجال الذكاء الاصطناعي حث مقدم الرعاية الصحية على النظر في اختبارات تشخيصية محددة، أو تدابير وقائية، أو خيارات علاج بناءً على المبادئ التوجيهية السريرية وأفضل الممارسات.

يمكن لهؤلاء العمال معالجة كميات هائلة من بيانات المرضى، وتطبيق التحليلات المتطورة، وتوليد رؤى قابلة للتنفيذ لدعم اتخاذ القرارات السريرية. وهذا يؤدي في النهاية إلى تحسين نتائج المرضى، وتقليل تكاليف الرعاية الصحية، وتعزيز إدارة صحة السكان.

عامل الذكاء الاصطناعي كمدير للعمليات



في سياق التطبيقات المدعومة بالذكاء الاصطناعي، يمكن تصميم العامل ليعمل كمدير للعمليات ، كما هو موضح في كتاب "أنماط تكامل المؤسسات" لغريغور هوبه . مدير العمليات هو مكون مركزي يحافظ على حالة العملية ويحدد خطوات المعالجة التالية بناءً على النتائج المرحلية.

عندما يعمل عامل الذكاء الاصطناعي كمدير للعمليات، فإنه يتلقى رسالة واردة تبدأ العملية، تُعرف باسم رسالة التشغيل . يحافظ عامل الذكاء الاصطناعي بعد ذلك على حالة تنفيذ العملية (كنص محادثة) ويعالج الرسالة من خلال سلسلة من خطوات المعالجة المنفذة كدوال أدواتية، والتي يمكن أن تكون متسلسلة أو متوازية، ويتم استدعاؤها وفقاً لتقديره.

إذا كنت تستخدم فئة من نماذج الذكاء الاصطناعي مثل GPT-4 التي تعرف كيفية تنفيذ الدوال بشكل متوازٍ، فيمكن لعاملك تنفيذ خطوات متعددة في وقت واحد. وأعترف أنني لم أجرب ذلك بنفسني وحدي يقول إن نتائجك قد تختلف.



بعد كل خطوة معالجة فردية، يتم إرجاع التحكم إلى عامل الذكاء الاصطناعي، مما يتيح له تحديد خطوة (خطوات) المعالجة التالية بناءً على الحالة الحالية والنتائج التي تم الحصول عليها.

قم بتخزين رسائل التشغيل الخاصة بك

من واقع تجربتي، من الذكاء تنفيذ رسالة التشغيل الخاصة بك ككائن مدعوم بقاعدة بيانات . بهذه الطريقة يتم تعريف كل مثل عملية بمفتاح أساسي فريد ويوفر لك مكاناً لتخزين الحالة المرتبطة بالتنفيذ، بما في ذلك نص محادثة الذكاء الاصطناعي.

على سبيل المثال، هذه نسخة مبسطة من فئة نموذج AccountChange في Olympia، والتي تمثل طلباً لإجراء تغيير على حساب المستخدم.

```

1  Information Schema == #
2  #
3  account_changes name: Table #
4  #
5  key primary ,null not           :uuid           id #
6  :string description #
7  null not           :string           state #
8  :jsonb transcript #
9  null not           :datetime created_at #
10 null not           :datetime updated_at #
11 null not           :uuid account_id #
12 #
13 Indexes #
14 #
15 (account_id) index_account_changes_on_account_id #
16 #
17 Keys Foreign #
18 #
19 accounts.id => (account_id fk_rails_... #
20 #
21 class AccountChange < ApplicationRecord
22   belongs_to :account
23
24   validates :description, presence: true
25
26   after_commit -> {
27     broadcast(:account_change_requested, self)
28   }, on: :create
29
30   state_machine initial: :requested do
31     event :completed do

```

```

32     transition all => :complete
33   end
34   event :failed do
35     transition all => :requires_human_review
36   end
37 end
38 end

```

يعمل صنف AccountChange كرسالة مخفية تبدأ عملية معالجة طلب تغيير الحساب. لاحظ كيف يتم بثها إلى نظام النشر/الاشتراك الفرعي المبني على Wisper في Olympia بعد اكتمال عملية الإنشاء.

إن تخزين الرسالة المخفية في قاعدة البيانات بهذه الطريقة يوفر سجلاً دائماً لكل طلب تغيير للحساب. يتم تعيين مفتاح أساسي فريد لكل نسخة من صنف AccountChange، مما يسمح بالتعرف على الطلبات الفردية وتتبعها بسهولة. هذا مفيد بشكل خاص لأغراض تسجيل التدقيق، حيث يمكن النظام من الاحتفاظ بسجل تاريخي لجميع تغييرات الحساب، بما في ذلك وقت طلبها، والتغييرات المطلوبة، والحالة الحالية لكل طلب.

في المثال المعطى، يتضمن صنف AccountChange حقولاً مثل description للتقاط تفاصيل التغيير المطلوب، و state لتمثيل الحالة الحالية للطلب (مثل: مطلوب، مكتمل، يتطلب_مراجعة_بشرية)، و transcript لتخزين نص محادثة الذكاء الاصطناعي المتعلق بالطلب. حقل description هو المطلوب الفعلي المستخدم لبدء أول إكمال للمحادثة مع الذكاء الاصطناعي. يوفر تخزين هذه البيانات سياقاً قيماً ويسمح بتتبع وتحليل أفضل لعملية تغيير الحساب.

يمكن تخزين الرسائل المخفية في قاعدة البيانات من معالجة الأخطاء والتعافي بشكل قوي. إذا حدث خطأ أثناء معالجة طلب تغيير الحساب، يقوم النظام بتمييز الطلب كفاشل وينقله إلى حالة تتطلب تدخلاً بشرياً. هذا يضمن عدم فقدان أو نسيان أي طلب، ويمكن معالجة وحل أي مشكلات بشكل صحيح.

يوفر عامل الذكاء الاصطناعي، بصفته مدير العمليات، نقطة تحكم مركزية ويمكن من قدرات قوية لإعداد التقارير وتصحيح الأخطاء في العمليات. ومع ذلك، من المهم ملاحظة أن استخدام عامل الذكاء الاصطناعي كمدير للعمليات لكل سيناريو تدفق عمل في تطبيقك قد يكون مبالغاً فيه.

دج عمال الذكاء الاصطناعي في هيكلية تطبيقك

عند دج عمال الذكاء الاصطناعي في هيكلية تطبيقك، هناك العديد من الاعتبارات التقنية التي يجب معالجتها لضمان التكامل السلس والتواصل الفعال بين عمال الذكاء الاصطناعي ومكونات التطبيق الأخرى. يتناول هذا القسم الجوانب الرئيسية لتصميم تلك الواجهات، ومعالجة تدفق البيانات، وإدارة دورة حياة عمال الذكاء الاصطناعي.

تصميم واجهات وبروتوكولات اتصال واضحة

لتسهيل التكامل السلس بين عمال الذكاء الاصطناعي ومكونات التطبيق الأخرى، من الضروري تحديد واجهات وبروتوكولات اتصال واضحة. ضع في اعتبارك النهج التالية:

التكامل القائم على واجهة برمجة التطبيقات: كشف وظائف عمال الذكاء الاصطناعي من خلال واجهات برمجة تطبيقات محددة جيداً، مثل نقاط النهاية RESTful أو مخططات GraphQL. هذا يسمح للمكونات الأخرى بالتفاعل مع عمال الذكاء الاصطناعي باستخدام طلبات واستجابات HTTP القياسية. يوفر التكامل القائم على واجهة برمجة التطبيقات عقداً واضحاً بين عمال الذكاء الاصطناعي والمكونات المستهلكة، مما يجعل من السهل تطوير نقاط التكامل واختبارها وصيانتها.

الاتصال القائم على الرسائل: تنفيذ أنماط الاتصال القائمة على الرسائل، مثل قوائم الرسائل أو أنظمة النشر-الاشتراك، لتمكين التفاعل غير المتزامن بين عمال الذكاء الاصطناعي والمكونات الأخرى. يفصل هذا النهج عمال الذكاء الاصطناعي عن بقية التطبيق، مما يسمح بقابلية أفضل للتوسع، وتحمل الأخطاء، والاقتران المرن. الاتصال القائم على الرسائل مفيد بشكل خاص عندما تكون المعالجة التي يقوم بها عمال الذكاء الاصطناعي تستغرق وقتاً طويلاً أو تستهلك موارد كثيرة، حيث يسمح لأجزاء أخرى من التطبيق بمواصلة التنفيذ دون انتظار عمال الذكاء الاصطناعي لإكمال مهامهم.

العمارة المدفوعة بالأحداث: تصميم نظامك حول الأحداث والمحفزات التي تنشط عمال الذكاء الاصطناعي عند استيفاء شروط معينة. يمكن لعمال الذكاء الاصطناعي الاشتراك في الأحداث ذات الصلة والتفاعل وفقاً لذلك، وتنفيذ مهامهم المحددة عند حدوث الأحداث. تمكن العمارة المدفوعة بالأحداث من المعالجة في الوقت الفعلي وتسمح باستدعاء عمال الذكاء الاصطناعي عند الطلب، مما يقلل من استهلاك الموارد غير الضروري. هذا النهج مناسب جداً للسيناريوهات التي يحتاج فيها عمال الذكاء الاصطناعي إلى الاستجابة لإجراءات أو تغييرات محددة في حالة التطبيق.

معالجة تدفق البيانات والتزامن

عند دمج عمال الذكاء الاصطناعي في تطبيقك، من الضروري ضمان تدفق سلس للبيانات والحفاظ على اتساق البيانات بين عمال الذكاء الاصطناعي والمكونات الأخرى. ضع في اعتبارك الجوانب التالية:

تحضير البيانات: قبل تغذية البيانات إلى عمال الذكاء الاصطناعي، قد تحتاج إلى تنفيذ مهام مختلفة لتحضير البيانات، مثل التنظيف والتنسيق و/أو تحويل البيانات المدخلة. لا تريد فقط التأكد من أن عمال الذكاء الاصطناعي يمكنهم المعالجة بفعالية، ولكنك تريد أيضاً التأكد من أنك لا تهدر الرموز في إعطاء اهتمام لمعلومات قد يعتبرها العامل عديمة الفائدة في أحسن الأحوال، ومشتتة في أسوأ الأحوال. قد يتضمن تحضير البيانات مهام مثل إزالة الضوضاء، ومعالجة القيم المفقودة، أو تحويل أنواع البيانات.

استمرارية البيانات: كيف ستقوم بتخزين وحفظ البيانات التي تتدفق داخل وخارج عمال الذكاء الاصطناعي؟ ضع في اعتبارك عوامل مثل حجم البيانات، وأنماط الاستعلام، وقابلية التوسع. هل تحتاج إلى حفظ نص الذكاء الاصطناعي كانعكاس لـ "عملية تفكيره" لأغراض التدقيق أو تصحيح الأخطاء، أم يكفي أن يكون لديك سجل للنتائج فقط؟

استرجاع البيانات: قد يتضمن الحصول على البيانات التي يحتاجها العمال الاستعلام من قواعد البيانات، والقراءة من الملفات، أو الوصول إلى واجهات برمجة التطبيقات الخارجية. تأكد من مراعاة زمن الاستجابة وكيفية وصول العمال الاصطناعيين إلى أحدث البيانات. هل يحتاجون إلى وصول كامل إلى قاعدة البيانات الخاصة بك أم يجب عليك تحديد نطاق وصولهم بدقة وفقاً لما يقومون به؟ ماذا عن قابلية التوسع؟ ضع في اعتبارك آليات التخزين المؤقت لتحسين الأداء وتقليل الحمل على مصادر البيانات الأساسية.

مزامنة البيانات: عندما تقوم مكونات متعددة، بما في ذلك العمال الاصطناعيين، بالوصول إلى البيانات المشتركة وتعديلها، من المهم تنفيذ آليات مزامنة مناسبة للحفاظ على اتساق البيانات. قد تساعد استراتيجيات قفل قواعد البيانات، مثل القفل التفاضلي أو التشاؤمي، في منع التعارضات وضمان سلامة البيانات. قم بتنفيذ تقنيات إدارة المعاملات لتجميع عمليات البيانات ذات الصلة والحفاظ على خصائص الذرية والاتساق والعزل والمتانة (ACID))

معالجة الأخطاء والتعافي: قم بتنفيذ آليات قوية لمعالجة الأخطاء والتعافي للتعامل مع المشكلات المتعلقة بالبيانات التي قد تنشأ أثناء عملية تدفق البيانات. تعامل مع الاستثناءات بأناقة وقدم رسائل خطأ ذات معنى للمساعدة في تصحيح الأخطاء. قم بتنفيذ آليات إعادة المحاولة واستراتيجيات التراجع للتعامل مع الأعطال المؤقتة أو انقطاعات الشبكة. حدد إجراءات واضحة لاسترداد البيانات واستعادتها في حالة تلف

البيانات أو فقدانها.

من خلال التصميم والتنفيذ الدقيق لآليات تدفق البيانات والمزامنة، يمكنك ضمان وصول العمال الاصطناعيين إلى بيانات دقيقة ومتسقة ومحدثة. هذا يمكنهم من أداء مهامهم بفعالية وإنتاج نتائج موثوقة.

إدارة دورة حياة العمال الاصطناعيين

قم بتطوير عملية موحدة لتهيئة وتكوين العمال الاصطناعيين. أنا أميل إلى الأطر التي توحد كيفية تحديد الإعدادات مثل أسماء النماذج والتوجيهات النظامية وتعريفات الوظائف. تأكد من أن عملية التهيئة آلية وقابلة للتكرار لتسهيل النشر والتوسع.

قم بتنفيذ آليات شاملة للمراقبة والتسجيل لتتبع صحة وأداء العمال الاصطناعيين. اجمع المقاييس مثل استخدام الموارد، ووقت المعالجة، ومعدلات الأخطاء، والإنتاجية. استخدم أنظمة التسجيل المركزية مثل مجموعة (Elasticsearch, ELK, Kibana Logstash) لتجميع وتحليل السجلات من عمال الذكاء الاصطناعي المتعددين.

قم ببناء تحمل الأخطاء والمرونة في بنية العامل الاصطناعي. نفذ آليات معالجة الأخطاء والتعافي للتعامل بسلاسة مع الأعطال أو الاستثناءات. نماذج اللغة الكبيرة لا تزال تقنية حديثة جداً؛ يميل مزودو الخدمة إلى التوقف غالباً في أوقات غير متوقعة. استخدم آليات إعادة المحاولة وقواطع الدائرة لمنع الأعطال المتتالية.

قابلية التركيب وتنسيق العمال الاصطناعيين

أحد المزايا الرئيسية لبنية العامل الاصطناعي هي قابليتها للتركيب، مما يتيح لك الجمع بين وتنسيق العديد من العمال الاصطناعيين لحل المشكلات المعقدة. من خلال تقسيم المهمة الأكبر إلى مهام فرعية أصغر وأكثر قابلية للإدارة، يتعامل معها كل عامل اصطناعي متخصص، يمكنك إنشاء أنظمة قوية ومرنة. في هذا القسم، سنستكشف مناهج مختلفة لتركيب وتنسيق "مجموعة" من العمال الاصطناعيين.

ربط العمال الاصطناعيين لسير عمل متعدد الخطوات

في العديد من السيناريوهات، يمكن تفكيك المهمة المعقدة إلى سلسلة من الخطوات المتتالية، حيث يصبح مخرجات أحد العمال الاصطناعيين مدخلات للعامل التالي. هذا الربط للعمال الاصطناعيين ينشئ سير

عمل أو خط معالجة متعدد الخطوات. يركز كل عامل اصطناعي في السلسلة على مهمة فرعية محددة، والمخرجات النهائية هي نتيجة الجهود المشتركة لجميع العمال.

دعنا نأخذ مثلاً في سياق تطبيق Rails on Ruby لمعالجة المحتوى المنشأ من قبل المستخدم. يتضمن سير العمل الخطوات التالية، والتي من المحتمل أن تكون كل منها بسيطة جداً بحيث لا تستحق التفكيك بهذه الطريقة في حالات الاستخدام الحقيقية، ولكنها تجعل المثال أسهل للفهم:

1. تنظيف النص: عامل اصطناعي مسؤول عن إزالة علامات HTML وتحويل النص إلى أحرف صغيرة ومعالجة توحيد Unicode.

2. اكتشاف اللغة: عامل اصطناعي يحدد لغة النص المنظف.

3. تحليل المشاعر: عامل اصطناعي يحدد المشاعر (إيجابية أو سلبية أو محايدة) للنص بناءً على اللغة المكتشفة.

4. تصنيف المحتوى: عامل اصطناعي يصنف النص إلى فئات محددة مسبقاً باستخدام تقنيات معالجة اللغة الطبيعية.

فيما يلي مثال مبسط جداً لكيفية ربط هؤلاء العمال الاصطناعيين معاً باستخدام Ruby:

```

1 class ContentProcessor
2   def initialize(text)
3     @text = text
4   end
5
6   def process
7     cleaned_text = TextCleanupWorker.new(@text).call
8     language = LanguageDetectionWorker.new(cleaned_text).call
9     sentiment = SentimentAnalysisWorker.new(cleaned_text, language).call
10    category = CategorizationWorker.new(cleaned_text, language).call
11
12    { cleaned_text:, language:, sentiment:, category: }
13  end
14 end

```

في هذا المثال، تتم تهيئة فئة ContentProcessor بالنص الخام وربط العمال الاصطناعيين معاً في طريقة process. يقوم كل عامل اصطناعي بأداء مهمته المحددة وتمرير النتيجة إلى العامل التالي في السلسلة. النتيجة النهائية هي جدول تجزئة يحتوي على النص المنظف واللغة المكتشفة والمشاعر وفئة المحتوى.

المعالجة المتوازية للعمال الاصطناعيين المستقلين

في المثال السابق، يتم ربط العمال الاصطناعيين بشكل متسلسل، حيث يقوم كل عامل بمعالجة النص وتمرير النتيجة إلى العامل التالي. ومع ذلك، إذا كان لديك عدة عمال اصطناعيين يمكنهم العمل بشكل مستقل على نفس المدخلات، يمكنك تحسين سير العمل عن طريق معالجتهم بالتوازي.

في السيناريو المعطى، بمجرد إجراء تنظيف النص بواسطة TextCleanupWorker، يمكن لكل من CategorizationWorker و SentimentAnalysisWorker و LanguageDetectionWorker معالجة النص المنظف بشكل مستقل. من خلال تشغيل هؤلاء العمال بالتوازي، يمكنك تقليل وقت المعالجة الإجمالي وتحسين كفاءة سير العمل الخاص بك.

لتحقيق المعالجة المتوازية في Ruby، يمكنك الاستفادة من تقنيات التزامن مثل المسارات أو البرمجة غير المتزامنة. إليك مثال على كيفية تعديل فئة ContentProcessor لمعالجة العمال الثلاثة النهائيين بالتوازي باستخدام المسارات:

```

1  require 'concurrent'
2
3  class ContentProcessor
4    def initialize(text)
5      @text = text
6    end
7
8    def process
9      cleaned_text = TextCleanupWorker.new(@text).call
10
11      language_future = Concurrent::Future.execute do
12        LanguageDetectionWorker.new(cleaned_text).call
13      end
14
15      sentiment_future = Concurrent::Future.execute do
16        SentimentAnalysisWorker.new(cleaned_text).call
17      end
18
19      category_future = Concurrent::Future.execute do
20        CategorizationWorker.new(cleaned_text).call
21      end
22
23      language = language_future.value

```

```

24     sentiment = sentiment_future.value
25     category = category_future.value
26
27     { cleaned_text:, language:, sentiment:, category: }
28 end
29 end

```

في هذه النسخة المحسنة، نستخدم مكتبة concurrent-ruby لإنشاء كائنات Concurrent::Future لكل من معالجات الذكاء الاصطناعي المستقلة. يمثل Future عملية حسابية سيتم تنفيذها بشكل غير متزامن في مسار تنفيذ منفصل.

بعد خطوة تنظيف النص، نقوم بإنشاء ثلاثة كائنات Future: future_language، future_sentiment، و future_category. كل Future يقوم بتنفيذ معالج الذكاء الاصطناعي المقابل له (LanguageDetectionWorker، SentimentAnalysisWorker، و CategorizationWorker) في مسار تنفيذ منفصل، مع تمرير text_cleaned كمدخل.

من خلال استدعاء طريقة value على كل Future، ننتظر اكتمال العملية الحسابية ونسترجع النتيجة. تقوم طريقة value بحظر التنفيذ حتى تصبح النتيجة متاحة، مما يضمن أن جميع المعالجات المتوازية قد أنهت معالجتها قبل المتابعة.

أخيراً، نقوم ببناء مجموعة المخرجات مع النص المنظف ونتائج المعالجات المتوازية، تماماً كما في المثال الأصلي. من خلال معالجة معالجات الذكاء الاصطناعي المستقلة بشكل متوازٍ، يمكنك تقليل وقت المعالجة الإجمالي مقارنة بتشغيلها بشكل متسلسل. هذا التحسين مفيد بشكل خاص عند التعامل مع المهام التي تستغرق وقتاً طويلاً أو عند معالجة كميات كبيرة من البيانات.

ومع ذلك، من المهم ملاحظة أن المكاسب الفعلية في الأداء تعتمد على عوامل مختلفة، مثل تعقيد كل معالج، وموارد النظام المتاحة، والتكلفة العامة لإدارة مسارات التنفيذ. من الجيد دائماً قياس الأداء وتحليل الشفرة البرمجية لتحديد المستوى الأمثل للتوازي لحالة الاستخدام المحددة.

بالإضافة إلى ذلك، عند تنفيذ المعالجة المتوازية، يجب الانتباه إلى أي موارد مشتركة أو تبعيات بين المعالجات. تأكد من أن المعالجات يمكنها العمل بشكل مستقل دون تعارضات أو حالات تسابق. إذا كانت هناك تبعيات أو موارد مشتركة، فقد تحتاج إلى تنفيذ آليات تزامن مناسبة للحفاظ على سلامة البيانات وتجنب مشاكل مثل الإغلاق المميت أو النتائج غير المتسقة.

قفل المفسر العالمي في Ruby والمعالجة غير المتزامنة

من المهم فهم آثار قفل المفسر العالمي (GIL) في Ruby عند النظر في المعالجة غير المتزامنة القائمة على مسارات التنفيذ في Ruby.

قفل المفسر العالمي هو آلية في مفسر Ruby تضمن أن مسار تنفيذ واحد فقط يمكنه تنفيذ شفرة Ruby في وقت واحد، حتى على المعالجات متعددة النواة. هذا يعني أنه على الرغم من إمكانية إنشاء وإدارة مسارات تنفيذ متعددة داخل عملية Ruby، إلا أن مسار تنفيذ واحد فقط يمكنه تنفيذ شفرة Ruby بشكل نشط في أي لحظة.

تم تصميم GIL لتبسيط تنفيذ مفسر Ruby وتوفير أمان مسارات التنفيذ لياكل البيانات الداخلية في Ruby. ومع ذلك، فإنه يحد أيضاً من إمكانية التنفيذ المتوازي الحقيقي لشفرة Ruby.

عندما تستخدم مسارات التنفيذ في Ruby، مثل استخدام مكتبة concurrent-ruby أو فئة Thread المدججة، تخضع مسارات التنفيذ لقيود GIL. يسمح GIL لكل مسار تنفيذ بتنفيذ شفرة Ruby لشريحة زمنية قصيرة قبل التبديل إلى مسار تنفيذ آخر، مما يخلق وهم التنفيذ المتزامن.

ومع ذلك، بسبب GIL، يظل التنفيذ الفعلي لشفرة Ruby متسلسلاً. أثناء تنفيذ مسار تنفيذ واحد لشفرة Ruby، تكون مسارات التنفيذ الأخرى في حالة توقف مؤقت، في انتظار دورها للحصول على GIL والتنفيذ.

هذا يعني أن المعالجة غير المتزامنة القائمة على مسارات التنفيذ في Ruby تكون أكثر فعالية للمهام المرتبطة بالإدخال/الإخراج، مثل انتظار استجابات API الخارجية (مثل نماذج اللغة الكبيرة المستضافة من طرف ثالث) أو تنفيذ عمليات الإدخال/الإخراج للملفات. عندما يواجه مسار التنفيذ عملية إدخال/إخراج، يمكنه تحرير GIL، مما يسمح لمسارات التنفيذ الأخرى بالتنفيذ أثناء انتظار اكتمال الإدخال/الإخراج.

من ناحية أخرى، للمهام المرتبطة بوحدة المعالجة المركزية، مثل العمليات الحسابية المكثفة أو معالجة معالج الذكاء الاصطناعي طويلة المدى، يمكن أن يحد GIL من مكاسب الأداء المحتملة للتوازي القائم على مسارات التنفيذ. نظراً لأن مسار تنفيذ واحد فقط يمكنه تنفيذ شفرة Ruby في وقت واحد، قد لا يتم تقليل وقت التنفيذ الإجمالي بشكل كبير مقارنة بالمعالجة المتسلسلة.

لتحقيق التنفيذ المتوازي الحقيقي للمهام المرتبطة بوحدة المعالجة المركزية في Ruby، قد تحتاج إلى

استكشاف نهج بديلة، مثل:

- استخدام التوازي القائم على العمليات مع عمليات Ruby متعددة، كل منها يعمل على نواة معالج منفصلة.
- الاستفادة من المكتبات أو الأطر الخارجية التي توفر امتدادات أصلية أو واجهات للغات بدون GIL، مثل C أو Rust.
- استخدام أطر الحوسبة الموزعة أو قوائم الرسائل لتوزيع المهام عبر أجهزة أو عمليات متعددة.

من الضروري النظر في طبيعة مهامك والقيود التي يفرضها GIL عند تصميم وتنفيذ المعالجة غير المتزامنة في Ruby. في حين أن المعالجة غير المتزامنة القائمة على مسارات التنفيذ يمكن أن توفر فوائد للمهام المرتبطة بالإدخال/الإخراج، قد لا تقدم تحسينات كبيرة في الأداء للمهام المرتبطة بوحدة المعالجة المركزية بسبب قيود GIL.

تقنيات التجميع لتحسين الدقة

تتضمن تقنيات التجميع دمج مخرجات معالجات الذكاء الاصطناعي المتعددة لتحسين الدقة الإجمالية أو متانة النظام. بدلاً من الاعتماد على معالج ذكاء اصطناعي واحد، تستفيد تقنيات التجميع من الذكاء الجماعي لمعالجات متعددة لاتخاذ قرارات أكثر استنارة.

تعتبر النماذج المجمعة ذات أهمية خاصة إذا كانت أجزاء مختلفة من سير عملك تعمل بشكل أفضل مع نماذج ذكاء اصطناعي مختلفة، وهذا أمر أكثر شيوعاً مما قد تعتقد. فالنماذج القوية مثل GPT-4 باهظة الثمن للغاية مقارنة بالخيارات مفتوحة المصدر الأقل قدرة، وربما لا تكون ضرورية لكل خطوة في سير عمل تطبيقك.



إحدى التقنيات الشائعة في التجميع هي التصويت بالأغلبية، حيث تقوم وحدات الذكاء الاصطناعي المتعددة بمعالجة نفس المدخلات بشكل مستقل، ويتم تحديد المخرجات النهائية من خلال توافق الأغلبية. يمكن لهذا النهج أن يساعد في التخفيف من تأثير أخطاء الوحدات الفردية وتحسين موثوقية النظام بشكل عام.

دعنا نأخذ مثالاً حيث لدينا ثلاث وحدات ذكاء اصطناعي لتحليل المشاعر ، كل منها يستخدم نموذجاً مختلفاً أو يتم تزويده بسياقات مختلفة. يمكننا دمج مخرجاتها باستخدام التصويت بالأغلبية لتحديد تنبؤ المشاعر النهائي.

```

1 class SentimentAnalysisEnsemble
2     def initialize(text)
3         @text = text
4     end
5
6     def analyze
7         predictions = [
8             SentimentAnalysisWorker1.new(@text).analyze,
9             SentimentAnalysisWorker2.new(@text).analyze,
10            SentimentAnalysisWorker3.new(@text).analyze
11        ]
12
13        predictions
14            .group_by { |sentiment| sentiment }
15            .max_by { |_, votes| votes.size }
16            .first
17
18    end
19 end

```

في هذا المثال، تقوم فئة SentimentAnalysisEnsemble بالتهيئة مع النص وتستدعي ثلاثة عمال مختلفين للذكاء الاصطناعي لتحليل المشاعر . تقوم طريقة analyze بجمع التنبؤات من كل عامل وتحدد المشاعر الأغلبية باستخدام طرق by_group و by_max. النتيجة النهائية هي المشاعر التي تحصل على أكثر الأصوات من مجموعة العمال

المجموعات هي بوضوح حالة حيث قد يكون التجريب مع التوازي يستحق وقتك.



الاختيار والاستدعاء الديناميكي لعمال الذكاء الاصطناعي

في بعض الحالات إن لم يكن معظمها، قد يعتمد عامل الذكاء الاصطناعي المحدد المراد استدعاؤه على ظروف وقت التشغيل أو مدخلات المستخدم. يسمح الاختيار والاستدعاء الديناميكي لعمال الذكاء

الاصطناعي بالمرونة والقدرة على التكيف في النظام.

قد تجد نفسك مغرباً بمحاولة وضع الكثير من الوظائف في عامل ذكاء اصطناعي واحد، مع إعطائه العديد من الوظائف ونص تحفيزي معقد يشرح كيفية استدعائها. قاوم هذا الإغراء، ثق بي. أحد الأسباب التي تجعل النهج الذي ناقشه في هذا الفصل يسمى "تعددية العمال" هو تذكيرنا بأنه من المرغوب فيه وجود الكثير من العمال المتخصصين، كل منهم يقوم بعمله الصغير في خدمة الهدف الأكبر.



على سبيل المثال، فكر في تطبيق روبوت محادثة حيث يكون عمال الذكاء الاصطناعي المختلفون مسؤولين عن معالجة أنواع مختلفة من استعلامات المستخدم. بناءً على مدخلات المستخدم، يختار التطبيق بشكل ديناميكي عامل الذكاء الاصطناعي المناسب لمعالجة الاستعلام.

```

1 class ChatbotController < ApplicationController
2   def process_query
3     query = params[:query]
4     query_type = QueryClassifierWorker.new(query).classify
5
6     case query_type
7     when 'greeting'
8       response = GreetingWorker.new(query).generate_response
9     when 'product_inquiry'
10      response = ProductInquiryWorker.new(query).generate_response
11    when 'order_status'
12      response = OrderStatusWorker.new(query).generate_response
13    else
14      response = DefaultResponseWorker.new(query).generate_response
15    end
16
17    render json: { response: response }
18  end
19 end

```

في هذا المثال، يتلقى ChatbotController استعلام المستخدم من خلال إجراء `query._process` يستخدم أولاً `QueryClassifierWorker` لتحديد نوع الاستعلام. وبناءً على نوع الاستعلام المصنف، يقوم المتحكم باختيار عامل الذكاء الاصطناعي المناسب ديناميكياً لإنشاء الاستجابة. يتيح هذا الاختيار

الديناميكي للروبوت المحادثة التعامل مع أنواع مختلفة من الاستعلامات وتوجيهها إلى عمال الذكاء الاصطناعي المناسبين.

نظراً لأن عمل QueryClassifierWorker بسيط نسبياً ولا يتطلب الكثير من السياق أو تعريفات الوظائف، فيمكنك على الأرجح تنفيذه باستخدام نموذج LLM صغير فائق السرعة مثل `mistralai/mixtral-8x7b-instruct:nitro`. لديه قدرات تقترب من مستوى 4GPT- في العديد من المهام، وفي وقت كتابة هذا التقرير، تستطيع Groq تقديمه بمعدل معالجة مذهل يبلغ 444 رمزاً في الثانية.



الجمع بين معالجة اللغات الطبيعية التقليدية ونماذج اللغة الكبيرة

في حين أن النماذج اللغوية الكبيرة (LLMs) قد أحدثت ثورة في مجال معالجة اللغات الطبيعية (NLP)، وقدّمت تنوعاً وأداءً لا مثيل لهما في مجموعة واسعة من المهام، إلا أنها ليست دائماً الحل الأكثر كفاءة أو فعالية من حيث التكلفة لكل مشكلة. في كثير من الحالات، يمكن أن يؤدي الجمع بين تقنيات معالجة اللغات الطبيعية التقليدية والنماذج اللغوية الكبيرة إلى مناهج أكثر تحسيناً واستهدافاً واقتصاداً لحل تحديات معالجة اللغات الطبيعية المحددة.

فكر في النماذج اللغوية الكبيرة كسكين الجيش السويسري في معالجة اللغات الطبيعية - متعددة الاستخدامات وقوية بشكل لا يصدق، ولكنها ليست بالضرورة الأداة الأفضل لكل مهمة. في بعض الأحيان، يمكن أن تكون الأداة المخصصة مثل فتاحة الزجاجات أو فتاحة العلب أكثر فعالية وكفاءة لمهمة محددة. وبالمثل، يمكن لتقنيات معالجة اللغات الطبيعية التقليدية، مثل تجميع المستندات وتحديد المواضيع والتصنيف، أن توفر غالباً حلاً أكثر استهدافاً وفعالية من حيث التكلفة لجوانب معينة من خط أنابيب معالجة اللغات الطبيعية الخاص بك.

إحدى المزايا الرئيسية لتقنيات معالجة اللغات الطبيعية التقليدية هي كفاءتها الحسابية. هذه الطرق، التي غالباً ما تعتمد على نماذج إحصائية أبسط أو مناهج قائمة على القواعد، يمكنها معالجة كميات كبيرة من البيانات النصية بشكل أسرع وبتكلفة حسابية أقل مقارنة بالنماذج اللغوية الكبيرة. هذا يجعلها مناسبة بشكل خاص للمهام التي تتضمن تحليل وتنظيم مجموعات كبيرة من المستندات، مثل تجميع المقالات المتشابهة أو تحديد المواضيع الرئيسية داخل مجموعة من النصوص.

علاوة على ذلك، يمكن لتقنيات معالجة اللغات الطبيعية التقليدية غالباً تحقيق دقة وضبط عاليين لمهام محددة، خاصة عندما يتم تدريبها على مجموعات بيانات خاصة بـ مجال معين. على سبيل المثال، يمكن لمصنف المستندات المضبوط جيداً باستخدام خوارزميات التعلم الآلي التقليدية مثل آلات المتجهات الداعمة (SVM) أو نموذج بايز الساذج تصنيف المستندات بدقة إلى فئات محددة مسبقاً بتكلفة حسابية ضئيلة.

ومع ذلك، تتألق النماذج اللغوية الكبيرة حقاً عندما يتعلق الأمر بالمهام التي تتطلب فهماً أعمق للغة والسياق والتفكير المنطقي. قدرتها على إنشاء نص متماسك ومرتبب بالسياق، والإجابة عن الأسئلة، وتلخيص المقاطع الطويلة لا تظاهيها طرق معالجة اللغات الطبيعية التقليدية. يمكن للنماذج اللغوية الكبيرة التعامل بفعالية مع الظواهر اللغوية المعقدة، مثل الغموض، والإحالة المشتركة، والتعبيرات الاصطلاحية، مما يجعلها لا غنى عنها للمهام التي تتطلب توليد اللغة الطبيعية أو فهمها.

تكن القوة الحقيقية في الجمع بين تقنيات معالجة اللغات الطبيعية التقليدية والنماذج اللغوية الكبيرة لإنشاء مناهج هجينة تستفيد من نقاط القوة في كليهما. من خلال استخدام طرق معالجة اللغات الطبيعية التقليدية للمهام مثل المعالجة المسبقة للمستندات والتجميع واستخراج المواضيع، يمكنك تنظيم وهيكل بياناتك النصية بكفاءة. يمكن بعد ذلك تغذية هذه المعلومات المنظمة إلى النماذج اللغوية الكبيرة للمهام أكثر تقدماً، مثل إنشاء الملخصات، والإجابة عن الأسئلة، أو إنشاء تقارير شاملة.

على سبيل المثال، دعنا نفكر في حالة استخدام حيث تريد إنشاء تقرير اتجاهات لـ مجال معين بناءً على مجموعة كبيرة من وثائق الاتجاهات الفردية. بدلاً من الاعتماد فقط على النماذج اللغوية الكبيرة، والتي يمكن أن تكون مكلفة حسابياً وتستغرق وقتاً طويلاً لمعالجة كميات كبيرة من النص، يمكنك استخدام نهج هجين:

1. استخدم تقنيات معالجة اللغات الطبيعية التقليدية، مثل نمذجة المواضيع (مثل تخصيص ديريكلي الكامن) أو خوارزميات التجميع (مثل خوارزمية كي-مينز)، لتجميع وثائق الاتجاهات المتشابهة معاً وتحديد المواضيع والموضوعات الرئيسية داخل المجموعة.
2. قم بتغذية المستندات المجمعة والمواضيع المحددة إلى نموذج لغوي كبير، مستفيداً من قدراته المتفوقة في فهم اللغة وتوليدها لإنشاء ملخصات متماسكة ومفيدة لكل مجموعة أو موضوع.
3. أخيراً، استخدم النموذج اللغوي الكبير لإنشاء تقرير اتجاهات شامل من خلال دمج الملخصات الفردية، وتبسيط الضوء على الاتجاهات الأكثر أهمية، وتقديم رؤى وتوصيات بناءً على المعلومات المجمعة.

من خلال الجمع بين تقنيات معالجة اللغات الطبيعية التقليدية والنماذج اللغوية الكبيرة بهذه الطريقة، يمكنك

معالجة كميات كبيرة من البيانات النصية بكفاءة، واستخراج رؤى ذات معنى، وإنشاء تقارير عالية الجودة مع تحسين الموارد الحسابية والتكاليف.

عند الشروع في مشاريع معالجة اللغات الطبيعية، من الضروري تقييم المتطلبات والقيود الخاصة بكل مهمة بعناية، والنظر في كيفية الاستفادة من أساليب معالجة اللغات الطبيعية التقليدية ونماذج اللغة الكبيرة معاً لتحقيق أفضل النتائج. من خلال الجمع بين كفاءة ودقة التقنيات التقليدية مع مرونة وقوة نماذج اللغة الكبيرة، يمكنك إنشاء حلول فعالة واقتصادية لمعالجة اللغات الطبيعية تقدم قيمة حقيقية للمستخدمين وأصحاب المصلحة.

استخدام الأدوات



في مجال تطوير التطبيقات المدعومة بالذكاء الاصطناعي، برز مفهوم "استخدام الأدوات" أو "استدعاء الدوال" كتقنية قوية تمكّن نموذج اللغة الكبير الخاص بك من الاتصال بالأدوات الخارجية، وواجهات برمجة التطبيقات، والدوال، وقواعد البيانات، وغيرها من الموارد. يتيح هذا النهج مجموعة أغنى من السلوكيات تتجاوز مجرد إخراج النص، وتفاعلات أكثر ديناميكية بين مكونات الذكاء الاصطناعي وبقية نظام تطبيقك. كما سندرس في هذا الفصل، يمنحك استخدام الأدوات أيضاً خيار جعل نموذج الذكاء الاصطناعي الخاص بك يولد البيانات بطرق منظمة.

ما هو استخدام الأدوات؟

استخدام الأدوات، المعروف أيضاً باسم استدعاء الدوال، هو تقنية تتيح للمطورين تحديد قائمة من الدوال التي يمكن لنموذج اللغة الكبير التفاعل معها أثناء عملية التوليد. يمكن أن تتراوح هذه الأدوات من دوال المساعدة البسيطة إلى واجهات برمجة التطبيقات المعقدة أو استعلامات قواعد البيانات. من خلال تزويد نموذج اللغة الكبير بإمكانية الوصول إلى هذه الأدوات، يمكن للمطورين توسيع قدرات النموذج وتمكينه من أداء المهام التي تتطلب معرفة أو إجراءات خارجية.

الشكل 8.8. مثال على تعريف دالة لعامل ذكاء اصطناعي يحلل المستندات

```

1  FUNCTION = {
2      name: "save_analysis",
3      description: "document for data analysis Save",
4      parameters: {
5          type: "object",
6          properties: {
7              title: {
8                  type: "string",
9                  maxLength: 140
10             },
11             summary: {
12                 type: "string",
13                 description: "with summary multi-paragraph comprehensive
14 applicable) (if sections of list and overview
15             },
16             tags: {
17                 type: "array",
18                 items: {
19                     type: "string",
20                     description: "themes main representing tags lowercase
21 document the of
22                 }
23             },
24             "required": %w[tags summary title]
25         }
26     }.freeze
27 
```

الفكرة الرئيسية وراء استخدام الأدوات هي منح النموذج اللغوي الكبير القدرة على اختيار وتنفيذ الأدوات

المناسبة بشكل ديناميكي بناءً على مدخلات المستخدم أو المهمة المطلوبة. وبدلاً من الاعتماد فقط على المعرفة المسبقة للنموذج، يتيح استخدام الأدوات للنموذج اللغوي الكبير الاستفادة من الموارد الخارجية لإنتاج استجابات أكثر دقة وملاءمة وقابلية للتنفيذ. يجعل استخدام الأدوات تقنيات مثل التوليد المعزز بالاسترجاع (RAG)) أسهل بكثير في التنفيذ مما قد تكون عليه بخلاف ذلك.

لاحظ أنه ما لم يُذكر خلاف ذلك، يفترض هذا الكتاب أن نموذج الذكاء الاصطناعي الخاص بك لا يمتلك إمكانية الوصول إلى أي أدوات مدمجة على جانب الخادم. يجب عليك التصريح صراحةً بأي أدوات تريد إتاحتها لنظام الذكاء الاصطناعي الخاص بك في كل طلب، API مع توفير آليات لتنفيذها إذا وعندما يثيرك الذكاء الاصطناعي أنه يرغب في استخدام تلك الأداة في استجابته.

إمكانات استخدام الأدوات

يفتح استخدام الأدوات مجموعة واسعة من الإمكانيات لتطبيقات الذكاء الاصطناعي. فيما يلي بعض الأمثلة على ما يمكن تحقيقه باستخدام الأدوات:

1. الروبوتات المحادثة والمساعدون الافتراضيون: من خلال ربط النموذج اللغوي الكبير بالأدوات الخارجية، يمكن للروبوتات المحادثة والمساعدون الافتراضيون تنفيذ مهام أكثر تعقيداً، مثل استرجاع المعلومات من قواعد البيانات، وتنفيذ استدعاءات API أو التفاعل مع الأنظمة الأخرى. على سبيل المثال، يمكن لروبوت المحادثة استخدام أداة إدارة علاقات العملاء لتغيير حالة صفقة بناءً على طلب المستخدم.
2. تحليل البيانات والرؤى: يمكن ربط النماذج اللغوية الكبيرة بأدوات أو مكتبات تحليل البيانات لتنفيذ مهام معالجة البيانات المتقدمة. هذا يمكن التطبيقات من توليد الرؤى، وإجراء التحليلات المقارنة، أو تقديم التوصيات المستندة إلى البيانات بناءً على استفسارات المستخدم.
3. البحث واسترجاع المعلومات: يتيح استخدام الأدوات للنماذج اللغوية الكبيرة التفاعل مع محركات البحث، وقواعد البيانات المتجهية، أو أنظمة استرجاع المعلومات الأخرى. من خلال تحويل استفسارات المستخدم إلى استعلامات بحث، يمكن للنموذج اللغوي الكبير استرجاع المعلومات ذات الصلة من مصادر متعددة وتقديم إجابات شاملة لأسئلة المستخدم.

4. التكامل مع الخدمات الخارجية: يتيح استخدام الأدوات التكامل السلس بين تطبيقات الذكاء الاصطناعي والخدمات الخارجية أو واجهات برمجة التطبيقات. على سبيل المثال، يمكن للنموذج اللغوي الكبير التفاعل مع API للطقس لتقديم تحديثات الطقس في الوقت الفعلي أو API للترجمة لإنشاء استجابات متعددة اللغات.

سير عمل استخدام الأدوات

يتضمن سير عمل استخدام الأدوات عادةً أربع خطوات رئيسية:

1. تضمين تعريفات الدوال في سياق الطلب الخاص بك
2. اختيار الأداة بشكل ديناميكي (أو صريح)
3. تنفيذ الدالة (الدوال)
4. استمرار اختياري للسياق الأصلي

دعونا نراجع كل خطوة من هذه الخطوات بالتفصيل.

تضمين تعريفات الدوال في سياق الطلب الخاص بك

يعرف الذكاء الاصطناعي الأدوات المتاحة له لأنك تقدم له قائمة بجزء من طلب الإكمال الخاص بك (عادةً ما يتم تعريفها كدوال باستخدام نسخة من مخطط JSON).

صيغة تعريف الأداة تختلف حسب النموذج المستخدم.

هذه هي طريقة تعريف دالة `weather_get` في Claude 3:

```

1  {
2    "name": "get_weather",
3    "description": location" given a in weather current the "Get,
4    "input_schema": {
5      "type": "object",
6      "properties": {
7        "location": {
8          "type": "string",
9          "description": CA" ,Francisco San e.g. ,state and city "The
10       },
11       "unit": {
12         "type": "string",
13         "enum": ["celsius", "fahrenheit"],
14         "description": temperature" of unit "The
15     }
16   },
17   "required": ["location"]
18 }
19 }

```

وهذه هي الطريقة التي يمكنك بها تعريف نفس الدالة لـ 4GPT، وتمثيلها كقيمة لمعامل tools:

```

1  {
2    "name": "get_current_weather",
3    "description": location" given a in weather current the "Get,
4    "parameters": {
5      "type": "object",
6      "properties": {
7        "location": {
8          "type": "string",
9          "description": CA" ,Francisco San e.g. ,state and city "The,
10       },
11       "unit": {
12         "type": "string",
13         "enum": ["celsius", "fahrenheit"],
14         "description": temperature" of unit "The
15     },
16   },
17   "required": ["location"],
18 },
19 }

```

نفس الشيء تقريباً، ولكن مختلف دون سبب واضح! كم هذا مزيج.

تحدد تعريفات الدالة الاسم والوصف ومعاملات الإدخال. يمكن تعريف معاملات الإدخال بشكل أكثر تفصيلاً باستخدام السمات مثل التعدادات لتحديد القيم المقبولة، وتحديد ما إذا كان المعامل مطلوباً أم لا. بالإضافة إلى تعريفات الدالة الفعلية، يمكنك أيضاً تضمين تعليمات أو سياق حول سبب وكيفية استخدام الدالة في توجيه النظام.

على سبيل المثال، تتضمن أداة البحث على الويب الخاصة بي في Olympia توجيه النظام هذا، والذي يُذكر الذكاء الاصطناعي بأن لديه الأدوات المذكورة تحت تصرفه:

- 1 The `google_search` and `realtime_search` functions let you do research
- 2 on behalf of the user. In contrast to Google, realtime search is powered
- 3 by Perplexity and provides real-time information to curated current events
- 4 databases and news sources. Make sure to include URLs in your response so
- 5 user can do followup research.

يعتبر تقديم الأوصاف التفصيلية العامل الأكثر أهمية في أداء الأداة. يجب أن تشرح أوصافك كل تفصيل عن الأداة، بما في ذلك:

- ما الذي تقوم به الأداة
- متى يجب استخدامها (ومتى لا يجب)
- ما معنى كل معامل وكيف يؤثر على سلوك الأداة
- أي تحذيرات أو قيود مهمة تنطبق على تنفيذ الأداة

كلما قدمت سياقاً أكثر للذكاء الاصطناعي حول أدواتك، كلما كان أفضل في تحديد متى وكيف يستخدمها. على سبيل المثال، توصي Anthropic بما لا يقل عن 3-4 جمل لكل وصف أداة لسلسلة Claude 3 الخاصة بها، وأكثر إذا كانت الأداة معقدة.

قد لا يكون بديهاً، ولكن الأوصاف تعتبر أيضاً أكثر أهمية من الأمثلة. في حين يمكنك تضمين أمثلة عن كيفية استخدام الأداة في وصفها أو في النص المصاحب، إلا أن هذا أقل أهمية من وجود شرح واضح وشامل لغرض الأداة ومعاملاتها. أضف الأمثلة فقط بعد أن تكون قد أكملت الوصف بشكل كامل.

إليك مثال على مواصفات دالة واجهة برمجة التطبيقات على غرار Stripe:

```

1  {
2    "name": "createPayment",
3    "description": request payment new a "Create,
4    "parameters": {
5      "type": "object",
6      "properties": {
7        "transaction_amount": {
8          "type": "number",
9          "description": paid" be to amount "The
10     },
11     "description": {
12       "type": "string",
13       "description": payment" the of description brief "A
14     },
15     "payment_method_id": {
16       "type": "string",
17       "description": used" be to method payment "The
18     },
19     "payer": {
20       "type": "object",
21       "description": ,name their including ,payer the about "Information
22 number" identification and ,email
23       "properties": {
24         "name": {
25           "type": "string",
26           "description": name" payer's "The
27         },
28         "email": {
29           "type": "string",
30           "description": address" email payer's "The
31         },
32         "identification": {
33           "type": "object",
34           "description": number" identification payer's "The,
35           "properties": {
36             "type": {
37               "type": "string",
38               "description": CNPJ)" ,CPF (e.g. document "Identification
39             },
40             "number": {

```

```

41         "type": "string",
42         "description": number identification "The
43     }
44 },
45     "required": [ "type", "number" ]
46 }
47 },
48     "required": [ "name", "email", "identification" ]
49 }
50 }
51 }

```

في الواقع العملي، تواجه بعض النماذج صعوبة في التعامل مع مواصفات الدوال المتداخلة والتعامل مع أنواع البيانات المعقدة مثل المصفوفات والقواميس وغيرها. ولكن من الناحية النظرية، يجب أن تكون قادراً على توفير مواصفات مخطط JSON بأي عمق!



اختيار الأدوات الديناميكي

عند تنفيذ إكمال محادثة يتضمن تعريفات الأدوات، يقوم نموذج اللغة الكبير باختيار الأداة (أو الأدوات) الأنسب للاستخدام ديناميكياً وإنشاء معاملات الإدخال المطلوبة لكل أداة.

في الواقع العملي، تتفاوت قدرة الذكاء الاصطناعي على استدعاء الدالة المناسبة تماماً، واتباع مواصفاتك للمدخلات بدقة. يساعد خفض معامل درجة الحرارة إلى 0.0 كثيراً، ولكن من واقع تجربتي ستظل تواجه أخطاء عرضية. تشمل هذه الإخفاقات أسماء دوال وهمية، ومعاملات إدخال خاطئة التسمية أو مفقودة تماماً. يتم تمرير المعاملات كـ JSON مما يعني أنك ستواجه أحياناً أخطاء ناتجة عن JSON مبتور أو غير مقتبس بشكل صحيح أو معطل بطريقة أخرى.

يمكن لأنماط البيانات ذاتية الإصلاح أن تساعد في إصلاح تلقائي لاستدعاءات الدوال التي تتعطل بسبب أخطاء بنائية.



اختيار الأدوات الإجباري (أو الصريح)

تتيح بعض النماذج خيار فرض استدعاء دالة معينة، كمعامل في الطلب. وبخلاف ذلك، يكون قرار استدعاء الدالة متروكاً تماماً لتقدير الذكاء الاصطناعي.

تعد القدرة على فرض استدعاء دالة أمراً حاسماً في سيناريوهات معينة حيث تريد ضمان تنفيذ أداة أو دالة محددة، بغض النظر عن عملية الاختيار الديناميكي للذكاء الاصطناعي. هناك عدة أسباب لأهمية هذه القدرة:

1. **التحكم الصريح:** قد تستخدم الذكاء الاصطناعي كـ مكون منفصل أو في سير عمل محدد مسبقاً يتطلب تنفيذ دالة معينة في وقت معين. من خلال فرض الاستدعاء، يمكنك ضمان استدعاء الدالة المطلوبة بدلاً من الاضطرار إلى الطلب بلطف من الذكاء الاصطناعي للقيام بذلك.
2. **التصحيح والاختبار:** عند تطوير واختبار التطبيقات المدعومة بالذكاء الاصطناعي، تكون القدرة على فرض استدعاءات الدوال لا تقدر بثمن لأغراض التصحيح. من خلال تشغيل دوال محددة بشكل صريح، يمكنك عزل واختبار المكونات الفردية لتطبيقك. يتيح لك هذا التحقق من صحة تنفيذ الدالة، والتحقق من صحة معاملات الإدخال، وضمان إرجاع النتائج المتوقعة.
3. **معالجة الحالات الحدية:** قد تكون هناك حالات حدية أو سيناريوهات استثنائية حيث قد لا تختار عملية الاختيار الديناميكي للذكاء الاصطناعي تنفيذ دالة يجب تنفيذها، وأنت تعرف ذلك بناءً على عمليات خارجية. في مثل هذه الحالات، تتيح لك القدرة على فرض استدعاء الدالة معالجة هذه المواقف بشكل صريح. قم بتحديد القواعد أو الشروط في منطق تطبيقك لتحديد متى تتجاوز تقدير الذكاء الاصطناعي.
4. **الاتساق وقابلية التكرار:** إذا كان لديك تسلسل محدد من الدوال التي يجب تنفيذها بترتيب معين، فإن فرض الاستدعاءات يضمن اتباع نفس التسلسل في كل مرة. هذا مهم بشكل خاص في التطبيقات التي يكون فيها الاتساق والسلوك المتوقع أمراً حاسماً، مثل الأنظمة المالية أو المحاكاة العلمية.
5. **تحسين الأداء:** في بعض الحالات، يمكن أن يؤدي فرض استدعاء دالة إلى تحسينات في الأداء. إذا كنت تعلم أن دالة معينة مطلوبة لمهمة معينة وأن عملية الاختيار الديناميكي للذكاء الاصطناعي قد تؤدي إلى عبء إضافي غير ضروري، يمكنك تجاوز عملية الاختيار واستدعاء الدالة المطلوبة مباشرة. يمكن أن يساعد هذا في تقليل زمن الاستجابة وتحسين الكفاءة الإجمالية لتطبيقك.

باختصار، توفر القدرة على فرض استدعاءات الدوال في التطبيقات المدعومة بالذكاء الاصطناعي تحكماً صريحاً، وتساعد في التصحيح والاختبار، وتعالج الحالات الحدية، وتضمن الاتساق وقابلية التكرار. إنها أداة قوية في ترسانتك، ولكن نحتاج إلى مناقشة جانب آخر من جوانب هذه الميزة المهمة.

في العديد من حالات اتخاذ القرار، نريد دائماً أن يقوم النموذج باستدعاء دالة وقد لا نريد أبداً أن يرد النموذج بمعرفته الداخلية فقط. على سبيل المثال، إذا كنت تقوم بالتوجيه بين نماذج متعددة متخصصة في مهام مختلفة (إدخال متعدد اللغات، الرياضيات، إلخ)، فقد تستخدم نموذج استدعاء الدوال لتفويض الطلبات إلى أحد النماذج المساعدة وعدم الرد بشكل مستقل.



معامل اختيار الأداة

يوفر 4GPT- ونماذج اللغة الأخرى التي تدعم استدعاء الدوال معامل `choice_tool` للتحكم في ما إذا كان استخدام الأداة مطلوباً كجزء من الإكمال. يمكن أن يأخذ هذا المعامل ثلاث قيم محتملة:

- `auto` يمنح الذكاء الاصطناعي حرية كاملة في استخدام أداة أو مجرد الرد
- `required` يخبر الذكاء الاصطناعي أنه يجب عليه استدعاء أداة بدلاً من الرد، ولكن يترك اختيار الأداة للذكاء الاصطناعي
- الخيار الثالث هو تعيين معامل `function_of_name` الذي تريد فرضه. سنتحدث عن ذلك في القسم التالي.

لاحظ أنه إذا قمت بتعيين اختيار الأداة إلى `required`، سيُجبر النموذج على اختيار الدالة الأكثر ملاءمة للاستدعاء من بين الدوال المتوفرة له، حتى لو لم تكن أي منها تناسب المدخل فعلياً. في وقت النشر، لست على علم بوجود نموذج سيقوم بإرجاع استجابة `calls_tool` فارغة، أو يستخدم طريقة أخرى لإعلامك بأنه لم يجد دالة مناسبة للاستدعاء.



إجبار دالة للحصول على مخرجات منظمة

إن القدرة على إجبار استدعاء دالة تمتك طريقة للحصول على بيانات منظمة من المحادثة المكتملة بدلاً من اضطرارك لاستخراجها بنفسك من الاستجابة النصية.

لماذا يُعد إجبار الدوال للحصول على مخرجات منظمة أمراً مهماً؟ ببساطة، لأن استخراج البيانات المنظمة من مخرجات نموذج اللغة الكبير أمر مزيج للغاية. يمكنك تسهيل الأمر قليلاً عن طريق طلب البيانات بصيغة XML، ولكن بعد ذلك عليك تحليل XML. وماذا تفعل عندما يكون XML مفقوداً لأن الذكاء الاصطناعي أجاب: “عذراً، لكنني غير قادر على توليد البيانات التي طلبتها لأن كذا وكذا...”

عند استخدام الأدوات بهذه الطريقة:

- يجب عليك على الأرجح تحديد أداة واحدة في طلبك
- تذكر إجبار استخدام دالتها باستخدام معامل `choice_tool`
- تذكر أن النموذج سيمرر المدخل إلى الأداة، لذا يجب أن يكون اسم الأداة ووصفها من منظور النموذج، وليس منظورك أنت

تستحق هذه النقطة الأخيرة مثالاً للتوضيح. لنفترض أنك تطلب من الذكاء الاصطناعي إجراء تحليل مشاعر على نص المستخدم. لن يكون اسم الدالة `sentiment_analyze`، بل سيكون شيئاً مثل `_save_sentiment_analysis`. الذكاء الاصطناعي هو من يقوم بتحليل المشاعر، وليس الأداة. كل ما تقوم به الأداة (من منظور الذكاء الاصطناعي) هو حفظ نتائج التحليل.

إليك مثال على استخدام Claude 3 لتسجيل ملخص صورة في JSON منظم بشكل جيد، هذه المرة من سطر الأوامر باستخدام `curl`:

```

1  curl https://api.anthropic.com/v1/messages \
2      --header application/json "content-type: \
3      --header " x-api-key:$ANTHROPIC_API_KEY" \
4      --header 2023-06-01 "anthropic-version: \
5      --header tools-2024-04-04 "anthropic-beta: \
6      --data \
7      '{
8      , "claude-3-sonnet-20240229" "model":
9      , 1024 "max_tokens":
10     [{ "tools":
11     , "record_summary" "name":
12     , JSON." well-structured into image of summary "Record "description":
13     { "input_schema":
14     , "object" "type":
15     { "properties":
16     { "key_colors":
17     , "array" "type":
18     { "items":
19     , "object" "type":
20     { "properties":
21     { "r":
22     , "number" "type":
23     "[0.1 ,0.0] value "red "description":
24     , }
25     { "g":
26     , "number" "type":
27     "[0.1 ,0.0] value "green "description":
28     , }
29     { "b":
30     , "number" "type":
31     "[0.1 ,0.0] value "blue "description":
32     , }
33     { "name":
34     , "string" "type":
35     name color "Human-readable "description":
36     e.g. ,snake_case in
37     \"olive_green\"or
38     \"turquoise\"
39     }
40     , }
41     ] "name" , "b" , "g" , "r" [ "required":
42     , }

```

```

43 less." or Four image. the in colors "Key "description":
44 ,}
45 { "description":
46 ,"string" "type":
47 max." sentences 1-2 description. "Image "description":
48 ,}
49 { "estimated_year":
50 ,"integer" "type":
51 ,taken was image the that year "Estimated "description":
52 the if this set Only photo. a it is if
53 non-fictional. be to appears image
54 okay!" are estimates Rough
55 }
56 ,}
57 ] "description" ,"key_colors" [ "required":
58 }
59 ,}}
60 [ "messages":
61 {
62 ,"user" "role":
63 [ "content":
64 {
65 ,"image" "type":
66 { "source":
67 ,"base64" "type":
68 "' "media_type": $IMAGE_MEDIA_TYPE,'
69 "' "data": $IMAGE_BASE64'"
70 }
71 ,}
72 {
73 ,"text" "type":
74 image." this describe to ``record_summary "Use "text":
75 }
76 ]
77 }
78 ]
79 }'

```

في المثال المقدم، نستخدم نموذج Claude 3 من Anthropic لإنشاء ملخص JSON منظم لصورة. إليك كيفية عمل ذلك:

1. نقوم بتعريف أداة واحدة تسمى summary_record في مصفوفة tools الخاصة بمحولة الطلب. هذه الأداة مسؤولة عن تسجيل ملخص للصورة في JSON منظم جيداً.
2. تحتوي أداة summary_record على schema_input يحدد البنية المتوقعة لمخرجات JSON. وهي تحدد ثلاث خصائص:

- colors_key: مصفوفة من الكائنات التي تمثل الألوان الرئيسية في الصورة. يحتوي كل كائن لون على خصائص لقيم الأحمر والأخضر والأزرق (تتراوح من 0.0 إلى 0.1) واسم لون مقروء بصيغة case.snake
- description: خاصية نصية لوصف موجز للصورة، مقتصرة على جملة أو جملتين.
- year_estimated: خاصية عدد صحيح اختيارية للسنة المقدرة التي التقطت فيها الصورة، إذا بدت كصورة غير خيالية.

3. في مصفوفة messages، نقدم بيانات الصورة كسلسلة مشفرة بنظام Base64 مع نوع الوسائط. هذا يسمح للنموذج بمعالجة الصورة كجزء من المدخلات.
4. نقوم أيضاً بحث Claude على استخدام أداة summary_record لوصف الصورة.
5. عندما يتم إرسال الطلب إلى نموذج Claude 3، يقوم بتحليل الصورة وإنشاء ملخص JSON بناءً على schema_input المحدد. يستخرج النموذج الألوان الرئيسية، ويقدم وصفاً موجزاً، ويقدر السنة التي التقطت فيها الصورة (إن أمكن).
6. يتم تمرير ملخص JSON المنشأ كعاملات لأداة summary_record، مما يوفر تمثيلاً منظماً للخصائص الرئيسية للصورة.

من خلال استخدام أداة summary_record مع schema_input محدد جيداً، يمكننا الحصول على ملخص JSON منظم للصورة دون الاعتماد على استخراج النص العادي. يضمن هذا النهج أن تتبع المخرجات تنسيقاً متسقاً ويمكن تحليلها ومعالجتها بسهولة بواسطة المكونات اللاحقة للتطبيق.

تعد القدرة على فرض استدعاء الدالة وتحديد بنية المخرجات المتوقعة ميزة قوية لاستخدام الأدوات في التطبيقات المدعومة بالذكاء الاصطناعي. فهي تتيح للمطورين المزيد من التحكم في المخرجات المنشأة وتبسط دمج البيانات المنشأة بواسطة الذكاء الاصطناعي في سير عمل تطبيقاتهم.

تنفيذ الدالة (الدوال)

لقد قمت بتعريف الدوال، وقت بتوجيه الذكاء الاصطناعي الخاص بك، الذي قرر أنه يجب عليه استدعاء إحدى دوالك. حان الوقت الآن لشفرة تطبيقك أو المكتبة، إذا كنت تستخدم حزمة Ruby مثل **raix-rails** لإرسال استدعاء الدالة ومعاملاتها إلى التنفيذ المقابل في شفرة تطبيقك.

تقرر شفرة تطبيقك ما يجب فعله بنتائج تنفيذ الدالة. ربما يتضمن ما يجب فعله سطرًا واحدًا من الشفرة في `lambda`، أو ربما يتضمن استدعاء API خارجي. ربما يتضمن استدعاء مكون ذكاء اصطناعي آخر، أو ربما يتضمن مئات أو حتى آلاف الأسطر من الشفرة في بقية نظامك. الأمر متروك لك تمامًا.

في بعض الأحيان يكون استدعاء الدالة هو نهاية العملية، ولكن إذا كانت النتائج تمثل معلومات في سلسلة تفكير يجب أن يواصلها الذكاء الاصطناعي، فإن شفرة تطبيقك تحتاج إلى إدراج نتائج التنفيذ في نص المحادثة والسماح للذكاء الاصطناعي بمواصلة المعالجة.

على سبيل المثال، هنا إعلان دالة **Raix** يستخدمه **AccountManager** الخاص بـ **Olympia** للتواصل مع عملائنا كجزء من تنسيق سير العمل الذي نلجأ إليه للعملاء.

```

1 class AccountManager
2   include Raix::ChatCompletion
3   include Raix::FunctionDispatch
4
5   functions... other of lots #
6
7   function :notify_account_owner,
8     "changed subscription if dollars Mention UUID. share Don't",
9     message: { type: "string" } do |arguments|
10     account.owner.freeform_notify(
11       subject: "Notification Change Account",
12       message: arguments[:message]
13     )
14     "owner account Notified"
15   end

```

قد لا يكون من الواضح مباشرةً ما يحدث هنا، لذا سأقوم بتفصيله.

1. يُعرّف صنف **AccountManager** العديد من الدوال المتعلقة بإدارة الحساب. يمكنه تغيير خطتك، وإضافة وإزالة أعضاء الفريق، من بين أمور أخرى.

2. تخبر تعليماته العلوية AccountManager بأنه يجب إخطار مالك الحساب بنتائج طلب تغيير الحساب، باستخدام دالة owner._account_notify
3. يتضمن التعريف المختصر للدالة:

- اسمها
- وصفها
- معاملاتها: { message: "string" type: }
- كلفة برمجية لتنفيذها عند استدعاء الدالة

بعد تحديث نص المحادثة بنتائج كلفة الدالة، يتم استدعاء طريقة completion_chat مرة أخرى. هذه الطريقة مسؤولة عن إرسال نص المحادثة المحدث مرة أخرى إلى نموذج الذكاء الاصطناعي لمزيد من المعالجة. نشير إلى هذه العملية باسم [حلقة المحادثة](#).

عندما يتلقى نموذج الذكاء الاصطناعي طلب إكمال محادثة جديد مع نص محادثة محدث، يكون لديه إمكانية الوصول إلى نتائج الدالة المنفذة سابقاً. يمكنه تحليل هذه النتائج، ودمجها في عملية صنع القرار، وإنشاء الاستجابة أو الإجراء التالي بناءً على السياق التراكمي للمحادثة. يمكنه اختيار تنفيذ دوال إضافية بناءً على السياق المحدث، أو يمكنه إنشاء استجابة نهائية للطلب الأصلي إذا قرر أنه لا حاجة لمزيد من استدعاءات الدوال.

المتابعة الاختيارية للطلب الأصلي

عندما ترسل نتائج الأداة مرة أخرى إلى النموذج اللغوي الكبير وتواصل معالجة الطلب الأصلي، يستخدم الذكاء الاصطناعي تلك النتائج إما لاستدعاء دوال إضافية أو لإنشاء استجابة نصية نهائية.

بعض النماذج مثل [Command-R](#) من Cohere يمكنها الاستشهاد بالأدوات المحددة التي

استخدمتها في استجاباتها، مما يوفر مزيداً من الشفافية وإمكانية التتبع.



اعتماداً على النموذج المستخدم، ستكون نتائج استدعاء الدالة موجودة في رسائل نص المحادثة التي لها دور خاص بها أو تعكس في صياغة أخرى. لكن الجزء المهم هو أن تكون تلك البيانات في نص المحادثة، حتى يمكن للذكاء الاصطناعي النظر فيها عند تقرير ما يجب فعله بعد ذلك.

خطأ شائع (وقد يكون مكلفاً) هو نسيان إضافة نتائج الدالة إلى نص المحادثة قبل المتابعة. نتيجة لذلك، سيتم حث الذكاء الاصطناعي بنفس الطريقة التي كان عليها قبل استدعاء الدالة في المرة الأولى. بمعنى آخر، من وجهة نظر الذكاء الاصطناعي، لم يتم باستدعاء الدالة بعد. لذا يقوم باستدعائها مرة أخرى. ومرة أخرى. ومرة أخرى، إلى الأبد حتى تقوم بلياقفه. نأمل ألا يكون سياقك كبيراً جداً، وألا يكون نموذجك مكلفاً جداً!



أفضل الممارسات لاستخدام الأدوات

للحصول على أفضل النتائج من استخدام الأدوات، ضع في اعتبارك أفضل الممارسات التالية.

التعريفات الوصفية

قم بتوفير أسماء ووصف واضح ودقيق لكل أداة ومعاملات إدخالها. هذا يساعد النموذج اللغوي الكبير على فهم الغرض وقدرات كل أداة بشكل أفضل.

يمكنني أن أخبرك من خلال التجربة أن الحكمة الشائعة التي تقول أن "التسمية صعبة" تنطبق هنا؛ لقد رأيت نتائج مختلفة بشكل كبير من النماذج اللغوية الكبيرة فقط من خلال تغيير أسماء الدوال أو صياغة الأوصاف. في بعض الأحيان يؤدي إزالة الأوصاف إلى تحسين الأداء.

معالجة نتائج الأدوات

عند تمرير نتائج الأدوات مرة أخرى إلى النموذج اللغوي الكبير، تأكد من أنها منظمة بشكل جيد وشاملة. استخدم مفاتيح وقيم ذات معنى لتمثيل مخرجات كل أداة. جرب تنسيقات مختلفة وانظر أيها يعمل بشكل أفضل، من JSON إلى النص العادي.

يعالج مفسر النتائج هذا التحدي من خلال توظيف الذكاء الاصطناعي لتحليل النتائج وتقديم تفسيرات صديقة للإنسان، أو ملخصات، أو استنتاجات رئيسية.

معالجة الأخطاء

قم بتنفيذ آليات قوية لمعالجة الأخطاء للتعامل مع الحالات التي قد ينشئ فيها النموذج اللغوي الكبير معاملات إدخال غير صالحة أو غير مدعومة لاستدعاءات الأدوات. تعامل بسلاسة مع أي أخطاء قد تحدث أثناء تنفيذ الأداة وتعاف منها.

إحدى الميزات الرائعة للغاية للذكاء الاصطناعي هي أنه يفهم رسائل الخطأ! مما يعني أنه إذا كنت تعمل بعقلية سريعة وغير منظمة، يمكنك ببساطة التقاط أي استثناءات تم إنشاؤها في تنفيذ أداة، وتمريرها مرة أخرى إلى الذكاء الاصطناعي حتى يعرف ما حدث!

على سبيل المثال، إليك نسخة مبسطة من تنفيذ البحث في Google في Olympia:

```
1 def google_search(conversation, params)
2   conversation.update_cstatus("Google... Searching")
3   query = params[:query]
4   search = GoogleSearch.new(query).get_hash
5
6   conversation.update_cstatus("results... Summarizing")
7   SummarizeKnowledgeGraph.new.perform(conversation, search.to_json)
8 rescue StandardError => e
9   Honeybadger.notify(e)
10   { error: e.message }.inspect
11 end
```

عمليات البحث في Google في Olympia تتم على خطوتين. أولاً تقوم بالبحث، ثم تلخص النتائج. في حالة حدوث أي خطأ، مهما كان نوعه، يتم تجميع رسالة الاستثناء وإرسالها مرة أخرى إلى الذكاء الاصطناعي. هذه التقنية هي أساس جميع أنماط المعالجة الذكية للأخطاء تقريباً

على سبيل المثال، لنفترض أن استدعاء GoogleSearch فشل بسبب استثناء Service 503 Unavailable. يتصاعد هذا الخطأ إلى مستوى الإنقاذ الأعلى، ويتم إرسال وصف الخطأ مرة أخرى إلى الذكاء الاصطناعي كنتيجة لاستدعاء الدالة. بدلاً من مجرد إظهار شاشة فارغة للمستخدم أو خطأ تقني، يقول الذكاء الاصطناعي شيئاً مثل "عذراً، لكنني غير قادر على الوصول إلى إمكانيات البحث في Google في الوقت الحالي. يمكنني المحاولة مرة أخرى لاحقاً، إذا أردت."

قد يبدو هذا مجرد حيلة ذكية، ولكن فكر في نوع مختلف من الأخطاء، حيث كان الذكاء الاصطناعي يستدعي واجهة برمجة تطبيقات خارجية وكان لديه تحكم مباشر في المعاملات التي يتم تمريرها إلى واجهة

البرمجة. ربما أخطأ في كيفية إنشاء تلك المعاملات؟ بشرط أن تكون رسالة الخطأ من واجهة البرمجة الخارجية مفصلة بما فيه الكفاية، فإن تمرير رسالة الخطأ مرة أخرى إلى الذكاء الاصطناعي المستدعي يعني أنه يمكنه إعادة النظر في تلك المعاملات والمحاولة مرة أخرى. تلقائياً. بغض النظر عن نوع الخطأ. والآن فكر في ما سيتطلبه تكرار هذا النوع من المعالجة القوية للأخطاء في الشيفرة العادية. إنه أمر مستحيل عملياً.

التحسين التكراري

إذا لم يكن نموذج اللغة الكبير يوصي بالأدوات المناسبة أو ينتج استجابات دون المستوى الأمثل، قم بالتكرار على تعريفات الأدوات والأوصاف ومعاملات الإدخال. واصل تحسين وتطوير إعدادات الأدوات بناءً على السلوك الملاحظ والنتائج المرجوة.

1. ابدأ بتعريفات أدوات بسيطة: ابدأ بتعريف الأدوات بأسماء وأوصاف ومعاملات إدخال واضحة وموجزة. تجنب تعقيد إعدادات الأدوات في البداية وركز على الوظائف الأساسية. على سبيل المثال، إذا كنت تريد حفظ نتائج تحليل المشاعر، ابدأ بتعريف أساسي مثل:

```

1  {
2    "name": "save_sentiment_score",
3    "description": "score" sentiment generate and text user-provided "Analyze",
4    "parameters": {
5      "type": "object",
6      "properties": {
7        "score": {
8          "type": "float",
9          "description": "(positive)" 1 to (negative) -1 from score "sentiment"
10       }
11     },
12     "required": ["score"]
13   }
14 }
```

٢. اختبر ولاحظ: بمجرد وضع تعريفات الأدوات الأولية، قم باختبارها باستخدام مُدخلات نصية مختلفة ولاحظ كيف يتفاعل النموذج اللغوي الكبير مع الأداة. انتبه إلى جودة وملاءمة الاستجابات المُولدة.

إذا كان النموذج اللغوي الكبير يُنتج استجابات دون المستوى الأمثل، فقد حان الوقت لتحسين تعريفات الأدوات.

٣. تحسين الأوصاف: إذا كان النموذج اللغوي الكبير يسيء فهم الغرض من أداة ما، حاول تحسين وصف الأداة. قدّم المزيد من السياق والأمثلة أو التوضيحات لتوجيه النموذج اللغوي الكبير في استخدام الأداة بشكل فعال. على سبيل المثال، يمكنك تحديث وصف أداة تحليل المشاعر لتتناول بشكل أكثر تحديداً النبرة العاطفية للنص الذي يتم تحليله:

```
1 {
2   "name": "save_sentiment_score",
3   "description": ,text of piece a of tone emotional overall the "Determine
4 comments." feedback or ,posts media social ,reviews customer as such ,
5   ...
6 }
```

٤. ضبط معاملات الإدخال: إذا كان النموذج اللغوي الكبير يُنشئ معاملات إدخال غير صالحة أو غير ذات صلة للأداة، فكّر في تعديل تعريفات المعاملات. أضف المزيد من القيود المحددة، أو قواعد التحقق، أو الأمثلة لتوضيح صيغة الإدخال المتوقعة.

٥. التكرار بناءً على التغذية الراجعة: راقب باستمرار أداء أدواتك واجمع التغذية الراجعة من المستخدمين أو أصحاب المصلحة. استخدم هذه التغذية الراجعة لتحديد مجالات التحسين وإجراء تحسينات متكررة على تعريفات الأداة. على سبيل المثال، إذا أبلغ المستخدمون أن التحليل لا يتعامل جيداً مع السخرية، يمكنك إضافة ملاحظة في الوصف:

```
1 {
2   "name": "save_sentiment_score",
3   "description": sentiment a return and text given a of sentiment the "Analyze
4 be should Sarcasm Note: (positive). 1 and (negative) -1 between score
5 negative." considered ,
6   ...
7 }
```

من خلال التحسين المتكرر لتعريفات أدواتك بناءً على السلوك الملاحظ والتغذية الراجعة، يمكنك تحسين أداء وفعالية تطبيقك المدعوم بالذكاء الاصطناعي تدريجياً. تذكر أن تحافظ على تعريفات الأدوات واضحة وموجزة ومركزة على المهمة المحددة. قم باختبار وتحقق من تفاعلات الأدوات بانتظام للتأكد من توافقها مع النتائج المرجوة.

تركيب وتسلسل الأدوات

أحد أقوى جوانب استخدام الأدوات التي تم التلميح إليها فقط حتى الآن هو القدرة على تركيب وربط أدوات متعددة معاً لإنجاز المهام المعقدة . من خلال التصميم الدقيق لتعريفات أدواتك وتنسيقات المدخلات/المخرجات الخاصة بها، يمكنك إنشاء وحدات بناء قابلة لإعادة الاستخدام يمكن دمجها بطرق مختلفة.

دعنا نأخذ مثلاً حيث تقوم ببناء خط أنابيب لتحليل البيانات لتطبيقك المدعوم بالذكاء الاصطناعي. قد يكون لديك الأدوات التالية:

1. **DataRetrieval**: أداة تجلب البيانات من قاعدة بيانات أو واجهة برمجة تطبيقات بناءً على معايير محددة.

2. **DataProcessing**: أداة تقوم بإجراء العمليات الحسابية أو التحويلات أو التجميعات على البيانات المسترجعة.

3. **DataVisualization**: أداة تعرض البيانات المعالجة بتنسيق سهل الاستخدام، مثل المخططات أو الرسوم البيانية.

من خلال ربط هذه الأدوات معاً، يمكنك إنشاء سير عمل قوي يسترجع البيانات ذات الصلة ويعالجها ويعرض النتائج بطريقة ذات معنى. إليك كيف قد يبدو سير عمل استخدام الأداة:

1. يتلقى النموذج اللغوي الكبير استعلاماً من المستخدم يطلب رؤية حول بيانات المبيعات لفئة منتج معينة.

2. يختار النموذج اللغوي الكبير أداة DataRetrieval ويولد معلمات الإدخال المناسبة لجلب بيانات المبيعات ذات الصلة من قاعدة البيانات.

3. يتم "تمرير" البيانات المسترجعة إلى أداة DataProcessing، التي تحسب المقاييس مثل إجمالي الإيرادات ومتوسط سعر المبيعات ومعدل النمو.

4. ثم تتم معالجة البيانات بواسطة أداة DataVisualization، التي تنشئ مخططاً أو رسماً بيانياً جذاباً بصرياً لتمثيل الرؤى، وتكرر عنوان URL للمخطط مرة أخرى إلى النموذج اللغوي الكبير.

5. أخيراً، يقوم النموذج اللغوي الكبير بإنشاء رد منسق على استعلام المستخدم باستخدام لغة ماركداون ، مع دمج البيانات المرئية وتقديم ملخص للنتائج الرئيسية.

من خلال تركيب هذه الأدوات معاً، يمكنك إنشاء سير عمل سلس لتحليل البيانات يمكن دمجها بسهولة في تطبيقك. جمال هذا النهج هو أنه يمكن تطوير واختبار كل أداة بشكل مستقل، ثم دمجها بطرق مختلفة لحل مشاكل متنوعة.

لتكثيف التركيب والتسلسل السلس للأدوات، من المهم تحديد تنسيقات واضحة للمدخلات والمخرجات لكل أداة.

على سبيل المثال، قد تقبل أداة DataRetrieval معلمات مثل تفاصيل الاتصال بقاعدة البيانات واسم الجدول وشروط الاستعلام، وتعيد مجموعة النتائج ككائن JSON منظم. يمكن لأداة DataProcessing بعد ذلك توقع كائن JSON هذا كمدخل وإنتاج كائن JSON محول كـمخرج. من خلال توحيد تدفق البيانات بين الأدوات، يمكنك ضمان التوافق وإعادة الاستخدام.

أثناء تصميم نظامك البيئي للأدوات، فكر في كيفية دمج الأدوات المختلفة لمعالجة حالات الاستخدام الشائعة في تطبيقك. فكر في إنشاء أدوات عالية المستوى تغلف سير العمل الشائع أو منطق الأعمال، مما يجعل من السهل على النموذج اللغوي الكبير اختيارها واستخدامها بفعالية.

تذكر أن قوة استخدام الأدوات تكمن في المرونة والنمطية التي يوفرها. من خلال تقسيم المهام المعقدة إلى أدوات أصغر قابلة لإعادة الاستخدام، يمكنك إنشاء تطبيق مدعوم بالذكاء الاصطناعي قوي وقابل للتكيف يمكنه معالجة مجموعة واسعة من التحديات.

الاتجاهات المستقبلية

مع تطور مجال تطوير التطبيقات المدعومة بالذكاء الاصطناعي، يمكننا توقع المزيد من التقدم في قدرات استخدام الأدوات. تشمل بعض الاتجاهات المستقبلية المحتملة:

1. استخدام الأدوات متعدد الخطوات: قد تكون النماذج اللغوية الكبيرة قادرة على تحديد عدد المرات التي تحتاج فيها إلى استخدام الأدوات من أجل إنشاء استجابة مرضية. قد يتضمن ذلك جولات متعددة من اختيار الأدوات وتنفيذها بناءً على النتائج الوسيطة.
2. الأدوات المحددة مسبقاً: قد توفر منصات الذكاء الاصطناعي مجموعة من الأدوات المحددة مسبقاً التي يمكن للمطورين الاستفادة منها مباشرة، مثل مفسرات Python وأدوات البحث على الويب أو وظائف المرافق الشائعة.

3. التكامل السلس: مع انتشار استخدام الأدوات، يمكننا توقع تكامل أفضل بين منصات الذكاء الاصطناعي وأطر التطوير الشائعة ، مما يسهل على المطورين دمج استخدام الأدوات في تطبيقاتهم.

استخدام الأدوات هو تقنية قوية تمكن المطورين من تسخير الإمكانيات الكاملة للنماذج اللغوية الكبيرة في التطبيقات المدعومة بالذكاء الاصطناعي. من خلال ربط النماذج اللغوية الكبيرة بالأدوات والموارد الخارجية، يمكنك إنشاء أنظمة أكثر ديناميكية وذكاءً ووعياً بالسياق يمكنها التكيف مع احتياجات المستخدم وتوفير رؤى وإجراءات قيمة.

في حين أن استخدام الأدوات يوفر إمكانيات هائلة، من المهم أن تكون على دراية بالتحديات والاعتبارات المحتملة. أحد الجوانب الرئيسية هو إدارة تعقيد تفاعلات الأدوات وضمان استقرار وموثوقية النظام بأكمله. تحتاج إلى معالجة السيناريوهات التي قد تفشل فيها استدعاءات الأدوات، أو تعيد نتائج غير متوقعة، أو لها آثار على الأداء. بالإضافة إلى ذلك، يجب عليك النظر في تدابير الأمان والتحكم في الوصول لمنع الاستخدام غير المصرح به أو الضرر للأدوات. تعتبر آليات معالجة الأخطاء والتسجيل والمراقبة المناسبة أمراً حاسماً للحفاظ على سلامة وأداء تطبيقك المدعوم بالذكاء الاصطناعي.

بينما تستكشف إمكانيات استخدام الأدوات في مشاريعك الخاصة، تذكر أن تبدأ بأهداف واضحة، وتصميم تعريفات أدوات منظمة جيداً، والتكرار بناءً على التغذية الراجعة والنتائج. مع النهج والعقلية المناسبة، يمكن لاستخدام الأدوات أن يفتح مستويات جديدة من الابتكار والقيمة في تطبيقاتك المدعومة بالذكاء الاصطناعي

معالجة التدفق



تدفق البيانات عبر بروتوكول HTTP، والمعروف أيضًا باسم الأحداث المرسلة من الخادم (SSE))، هو آلية يقوم فيها الخادم بإرسال البيانات باستمرار إلى العميل حال توفرها، دون الحاجة إلى طلب صريح من العميل. وبما أن استجابة الذكاء الاصطناعي يتم توليدها بشكل تدريجي، فن المنطقي توفير تجربة مستخدم متجاوبة من خلال عرض مخرجات الذكاء الاصطناعي أثناء توليدها. وفي الواقع، جميع واجهات برمجة التطبيقات (APIs) لمزودي الذكاء الاصطناعي التي أعرفها تقدم خيار الاستجابات المتدفقة في نقاط نهاية الإكمال الخاصة بها.

السبب في ظهور هذا الفصل هنا في الكتاب، مباشرة بعد استخدام الأدوات هو بسبب قوة الجمع بين استخدام الأدوات واستجابات الذكاء الاصطناعي المباشرة للمستخدمين. يتيح ذلك تجارب ديناميكية

وتفاعلية حيث يمكن للذكاء الاصطناعي معالجة مدخلات المستخدم، واستخدام مختلف الأدوات والدوال وفقاً لتقديره، ثم تقديم استجابات في الوقت الفعلي.

لتحقيق هذا التفاعل السلس، تحتاج إلى كتابة معالجات تدفق يمكنها توجيه استدعاءات الدوال الأداة التي يستخدمها الذكاء الاصطناعي بالإضافة إلى إخراج النص العادي للمستخدم النهائي. تضيف الحاجة إلى التكرار بعد معالجة الدالة الأداة تحدياً مثيراً للاهتمام إلى المهمة.

تنفيذ ReplyStream

لتوضيح كيفية تنفيذ معالجة التدفق، سيتعمق هذا الفصل في نسخة مبسطة من فئة ReplyStream المستخدمة في Olympia. يمكن تمرير نسخ من هذه الفئة كعامل stream في مكتبات عميل الذكاء الاصطناعي مثل `ruby-openai` و `openrouter`

إليك كيفية استخدام ReplyStream في PromptSubscriber الخاص بـ Olympia والذي يستمع عبر Wisper لإنشاء رسائل المستخدم الجديدة.

```

1 class PromptSubscriber
2   include Raix::ChatCompletion
3   include Raix::PromptDeclarations
4
5   omitted... declarations other many #
6
7   prompt text: -> { user_message.content },
8               stream: -> { ReplyStream.new(self) },
9               until: -> { bot_message.complete? }
10
11   def message_created(message) Wisper by invoked #
12     return unless message.role.user? && message.content?
13
14     omitted... implementation the of rest #

```

بالإضافة إلى مرجع context الخاص بمشترك المطالبة الذي قام بإنشائه، تحتوي فئة ReplyStream أيضاً على متغيرات نسخة لتخزين مخزن مؤقت للبيانات المستلمة، ومصفوفات لتتبع أسماء الدوال والمعاملات التي تم استدعاؤها أثناء معالجة التدفق.

```

1 class ReplyStream
2   attr_accessor :buffer, :f_name, :f_arguments, :context
3
4   delegate :bot_message, :dispatch, to: :context
5
6   def initialize(context)
7     self.context = context
8     self.buffer = []
9     self.f_name = []
10    self.f_arguments = []
11  end
12
13  def call(chunk, bytesize = nil)
14    ... #
15  end
16
17  ... #
18 end

```

تقوم دالة initialize بإعداد الحالة الأولية لنسخة ReplyStream، حيث تقوم بتهيئة المخزن المؤقت والسياق والمتغيرات الأخرى.

تعتبر دالة call نقطة الدخول الرئيسية لمعالجة البيانات المتدفقة. تستقبل هذه الدالة جزءاً من البيانات (مثلاً كجدول تجزئة) ومعامل اختياري bytesize، والذي لا يتم استخدامه في مثالنا. داخل هذه الدالة، تستخدم الفئة مطابقة الأنماط للتعامل مع السيناريوهات المختلفة بناءً على هيكل الجزء المستلم.

إن استدعاء keys_symbolize_deep على الجزء يساعد في جعل مطابقة الأنماط أكثر أناقة، من خلال السماح لنا بالعمل مع الرموز بدلاً من السلاسل النصية.



```

1  def call(chunk, _bytesize)
2      case chunk.deep_symbolize_keys
3
4      in { name function match #
5          choices: [
6              {
7                  delta: {
8                      tool_calls: [
9                          { index: index, function: {name: name} }
10                     ]
11                 }
12             }
13         ] }
14
15     f_name[index] = name

```

النمط الأول الذي نطابقه هو استدعاء الأداة مع اسم الدالة المرتبط به. إذا اكتشفنا واحداً، نضعه في مصفوفة name._f نقوم بتخزين أسماء الدوال في مصفوفة مفهرسة، لأن النموذج قادر على استدعاء الدوال بشكل متوازٍ، مما يتيح إرسال أكثر من دالة للتنفيذ في وقت واحد.

استدعاء الدوال المتوازي هو قدرة نموذج الذكاء الاصطناعي على تنفيذ استدعاءات متعددة للدوال معاً، مما يسمح بحل تأثيرات ونتائج هذه الاستدعاءات بشكل متوازٍ. هذا مفيد بشكل خاص إذا كانت الدوال تستغرق وقتاً طويلاً، ويقلل من الرحلات ذهاباً وإياباً مع واجهة برمجة التطبيقات، مما يمكن أن يوفر قدراً كبيراً من استهلاك الرموز.

بعد ذلك نحتاج إلى المطابقة مع المعاملات المقابلة لاستدعاءات الدوال.

```

1   in { arguments match #
2     choices: [
3       {
4         delta: {
5           tool_calls: [
6             {
7               index: index, function: {arguments: argument }
8             }
9           ]
10        }
11      }
12    ]}

14   f_arguments[index] ||= "" already not if initialize #
15   f_arguments[index] << argument

```

وبشكل مماثل لكيفية تعاملنا مع أسماء الدوال، نقوم بتخزين المعاملات في مصفوفة مفهرسة. بعد ذلك، نبحث عن الرسائل العادية الموجهة للمستخدم، والتي ستصل من الخادم رمزاً تلو الآخر وسيتم تعيينها للمتغير `content._new` كما نحتاج أيضاً إلى مراقبة `reason._finish` سيكون قيمته `nil` حتى آخر جزء من تسلسل المخرجات.

```

1   in {
2     choices: [
3       { delta: {content: new_content}, finish_reason: finish_reason }
4     ]}

6   here... user the to chunk every transmit could you #
7   buffer << new_content.to_s

9   if finish_reason.present?
10    finalize
11  elsif new_content.to_s.match?(/\n\n/)
12    send_to_client paragraph per once transmit and buffer ...or #
13  end

```

من المهم أن نضيف تعبير مطابقة النمط للتعامل مع رسائل الخطأ المرسلّة من مزود نموذج الذكاء الاصطناعي. في ، نقوم بإثارة استثناء، ولكن في بيئة الإنتاج، نقوم بتسجيل الخطأ والإنهاء.

```

1  in { error: { message: } }
2    if Rails.env.local?
3      raise message
4    else
5      Honeybadger.notify(" Error: AI#{message}")
6      finalize
7    end

```

ستنفذ جملة else النهائية في عبارة case إذا لم تتطابق أي من الأنماط السابقة. إنه مجرد إجراء وقائي حتى نكتشف إذا بدأ نموذج الذكاء الاصطناعي في إرسال أجزاء غير معروفة إلينا.

```

1  else
2    Honeybadger.notify(" Chunk: Unrecognized#{chunk}")
3  end
4  end

```

تعتبر الدالة client_to_send مسؤولية عن إرسال المحتوى المخزن مؤقتاً إلى العميل. حيث تتحقق من أن المخزن المؤقت ليس فارغاً، وتقوم بتحديث محتوى رسالة البوت، وعرض رسالة البوت، وحفظ المحتوى في قاعدة البيانات لضمان استمرارية البيانات.

```

1  def send_to_client
2    whitespace pure process to need no #
3    return if buffer.join.squish.blank?
4
5    message bot the on content buffer the set #
6    content = buffer.join
7    bot_message.content = content
8
9    data lose never we that so database to save #
10   correctly terminate doesn't stream the if even #
11   bot_message.update_column(:content, content)
12
13   websocket via content update #
14   ConversationRenderer.update(bot_message)
15 end

```

يتم استدعاء طريقة finalize عند اكتمال معالجة التدفق. تقوم بتنفيذ استدعاءات الدالة إذا تم استقبال أي منها خلال التدفق، وتحديث رسالة البوت بالمحتوى النهائي والمعلومات الأخرى ذات الصلة، وإعادة تعيين سجل استدعاءات الدالة

```

1  def finalize
2    if f_name.any?
3      f_name.each_with_index do |name, index|
4        implemented it's wherever function the calling of care takes #
5        dispatch(name:, arguments: JSON.parse(f_arguments[index]))
6      end
7
8      history call function the reset #
9      f_name.clear
10     f_arguments.clear
11   else
12     content = buffer.join.presence
13     bot_message.update!(content:, complete: true)
14     ConversationRenderer.update(bot_message)
15   end
16 end

```

عندما يقرر النموذج استدعاء دالة، تحتاج إلى "توجيه" استدعاء تلك الدالة (الاسم والمعطيات) بطريقة تضمن تنفيذها وإضافة رسائل `call_function` و `result_function` إلى نص المحادثة من واقع تجريبي، من الأفضل التعامل مع إنشاء رسائل الدوال في مكان واحد في قاعدة الشيفرة الخاصة بك، بدلاً من الاعتماد على تطبيقات الأدوات. هذا لا يجعل الشيفرة أنظف فحسب، بل له أيضاً سبب عملي مهم جداً: إذا قام نموذج الذكاء الاصطناعي باستدعاء دالة، ولم يرَ رسائل الاستدعاء والنتيجة في نص المحادثة عند التكرار، فسيقوم باستدعاء نفس الدالة مرة أخرى. وقد يستمر في ذلك إلى ما لا نهاية. تذكر أن الذكاء الاصطناعي عديم الحالة تماماً، لذا ما لم تقم بإظهار استدعاءات تلك الدوال له مرة أخرى، فكأنها لم تحدث من الأساس.

```

1  PromptSubscriber#dispatch #
2
3  def dispatch(name:, arguments:)
4    transcript conversation the to message function_call a adds #
5    result returns and tool to dispatches plus #
6    conversation.function_call!(name, arguments).then do |result|
7      transcript the to message result function add #
8      conversation.function_result!(name, result)
9    end
10  end

```

إن مسح سجل استدعاءات الدوال بعد تنفيذها مهم تماماً مثل أهمية التأكد من وصول الاستدعاء ونتائجه إلى النص المكتوب الخاص بك، وذلك حتى لا تستمر في استدعاء نفس الدوال مراراً وتكراراً في كل مرة تدور فيها الحلقة.



حلقة المحادثة

في فئة PromptSubscriber، نستخدم طريقة prompt من وحدة PromptDeclarations لتحديد سلوك حلقة المحادثة. يتم تعيين المعامل until إلى <- { message.complete?_bot }، مما يعني أن الحلقة ستستمر حتى يتم تعليم message_bot على أنها مكتملة.

```

1  prompt text: -> { user_message.content },
2    stream: -> { ReplyStream.new(self) },
3    until: -> { bot_message.complete? }

```

ولكن متى يتم وضع علامة اكتمال على message_bot؟ إذا نسيت، ارجع إلى السطر 13 من طريقة .finalize

دعونا نراجع منطق معالجة التدفق بأكمله.

1. يتلقى PromptSubscriber رسالة مستخدم جديدة عبر طريقة created_message، والتي يتم استدعاؤها بواسطة نظام النشر/الاشتراك Wisper في كل مرة ينشئ فيها المستخدم النهائي إدخالاً جديداً.

2. تحدد طريقة الفئة prompt بشكل تصريحي سلوك منطق إكمال المحادثة لـ PromptSubscriber. سيتم تنفيذ إكمال المحادثة للنموذج الذي مع محتوى رسالة المستخدم، ونسخة جديدة من ReplyStream كمعامل التدفق، وشرط الحلقة المحدد.
3. يقوم النموذج الذي بمعالجة الإدخال ويبدأ في توليد استجابة. أثناء تدفق الاستجابة، يتم استدعاء طريقة call الخاصة بنسخة ReplyStream لكل جزء من البيانات.
4. إذا قرر النموذج الذي استدعاء دالة أداة، يتم استخراج اسم الدالة والوسائط من الجزء وتخزينها في المصفوفات name_f و arguments_f على التوالي.
5. إذا قام النموذج الذي بتوليد محتوى موجه للمستخدم، يتم تخزينه مؤقتاً وإرساله إلى العميل عبر طريقة client._to_send
6. بمجرد اكتمال معالجة التدفق، يتم استدعاء طريقة finalize. إذا تم استدعاء أي دوال أدوات أثناء التدفق، يتم إرسالها باستخدام طريقة dispatch الخاصة بـ PromptSubscriber.
7. تضيف طريقة dispatch رسالة call_function إلى نص المحادثة، وتنفذ دالة الأداة المقابلة، وتضيف رسالة result_function إلى النص مع نتيجة استدعاء الدالة.
8. بعد إرسال دوال الأدوات، يتم مسح سجل استدعاءات الدوال لمنع استدعاءات الدوال المكررة في الحلقات اللاحقة.
9. إذا لم يتم استدعاء أي دوال أدوات، تقوم طريقة finalize بتحديث message_bot بالمحتوى النهائي، وتضع عليها علامة اكتمال، وترسل الرسالة المحدثة إلى العميل.
10. يتم تقييم شرط الحلقة - { message.complete?_bot } . إذا لم يتم وضع علامة اكتمال على message_bot، تستمر الحلقة، ويتم إرسال الإدخال الأصلي مرة أخرى مع نص المحادثة المحدث.
11. تتكرر الخطوات 3-10 حتى يتم وضع علامة اكتمال على message_bot، مما يشير إلى أن النموذج الذي قد انتهى من توليد استجابته ولا يلزم تنفيذ المزيد من دوال الأدوات.

من خلال تنفيذ حلقة المحادثة هذه، تمكّن النموذج الذي من الانخراط في تفاعل متبادل مع التطبيق، وتنفيذ دوال الأدوات حسب الحاجة وتوليد استجابات موجهة للمستخدم حتى تصل المحادثة إلى نهاية طبيعية.

يتيح الجمع بين معالجة التدفق وحلقة المحادثة تجارب تفاعلية وديناميكية مدعومة بالذكاء الاصطناعي، حيث يمكن للنموذج الذي معالجة مدخلات المستخدم، واستخدام مختلف الأدوات والدوال، وتقديم استجابات في الوقت الفعلي بناءً على سياق المحادثة المتطور.

الاستمرار التلقائي

من المهم أن نكون على دراية بقيود مخرجات الذكاء الاصطناعي. معظم النماذج لديها حد أقصى لعدد الرموز التي يمكنها توليدها في استجابة واحدة، والذي يتم تحديده بواسطة معامل `tokens._max` إذا وصل نموذج الذكاء الاصطناعي إلى هذا الحد أثناء توليد استجابة، فسيتوقف فجأة ويشير إلى أن المخرجات قد تم اقتطاعها.

في الاستجابة المتدفقة من واجهة برمجة تطبيقات منصة الذكاء الاصطناعي، يمكنك اكتشاف هذه الحالة من خلال فحص متغير `reason_finish` في الجزء. إذا تم تعيين `reason_finish` إلى `"length"` (أو أي قيمة مفتاح أخرى خاصة بالنموذج)، فهذا يعني أن النموذج وصل إلى حد الرموز الأقصى أثناء التوليد وتم اقتطاع المخرجات.

إحدى طرق معالجة هذا السيناريو بسلاسة وتوفير تجربة مستخدم سلسة هي تنفيذ آلية استمرار تلقائي في منطق معالجة التدفق الخاص بك. من خلال إضافة نمط مطابقة لأسباب الإنهاء المتعلقة بالطول، يمكنك اختيار التكرار ومواصلة المخرجات تلقائياً من حيث توقفت.

إليك مثال مبسط عمداً لكيفية تعديل طريقة `call` في فئة `ReplyStream` لدعم الاستمرار التلقائي:

```

1  LENGTH_STOPS = %w[MAX_TOKENS length]
2
3  def call(chunk, _bytesize)
4    case chunk.deep_symbolize_keys
5      ... #
6
7    in {
8      choices: [
9        { delta: {content: new_content},
10          finish_reason: finish_reason } ] }
11
12    buffer << new_content.to_s
13
14    if finish_reason.blank?
15      send_to_client if new_content.to_s.match?(/\n\n/)
16    elsif LENGTH_STOPS.include?(finish_reason)
17      continue_cutoff
18    else
19      finalize

```

```

20     end
21
22     ... #
23     end
24 end
25
26 private
27
28 def continue_cutoff
29     conversation.bot_message!(buffer.join, visible: false)
30     conversation.user_message!("continue please", visible: false)
31     bot_message.update_column(:created_at, Time.current)
32 end

```

في هذه النسخة المعدلة، عندما يشير reason_finish إلى مخرجات مقتطعة، فبدلاً من إنهاء التدفق، نقوم بإضافة زوج من الرسائل إلى النص المكتوب دون إنهائه، وننقل رسالة الاستجابة الأصلية الموجهة للمستخدم إلى "أسفل" النص المكتوب عن طريق تحديث خاصية at_created الخاصة بها، ثم نترك الحلقة تحدث، بحيث يواصل الذكاء الاصطناعي التوليد من حيث توقف.

تذكر أن نقطة نهاية إكمال الذكاء الاصطناعي لا تحتفظ بالحالة. فهي "تعرف" فقط ما تخبرها به عبر النص المكتوب. في هذه الحالة، الطريقة التي نوصل بها للذكاء الاصطناعي أنه تم قطعه هي عن طريق إضافة رسائل "غير مرئية" (للمستخدم النهائي) إلى النص المكتوب. ولكن تذكر أن هذا مثال مبسط عن قصد. سيحتاج التطبيق الفعلي إلى إدارة إضافية للنص المكتوب لضمان عدم إهدار الرموز و/أو إرباك الذكاء الاصطناعي برسائل المساعد المكررة في النص.

يجب أن يحتوي التطبيق الفعلي للاستمرار التلقائي أيضاً على ما يسمى بمنطق قاطع الدائرة لمنع التكرار غير المنضبط. والسبب في ذلك هو أنه، نظراً لأنواع معينة من مطالبات المستخدم وإعدادات tokens_max المنخفضة، يمكن للذكاء الاصطناعي أن يستمر في تكرار المخرجات الموجهة للمستخدم بلا نهاية.

ضع في اعتبارك أن كل حلقة تتطلب طلباً منفصلاً، وأن كل طلب يستهلك النص المكتوب بأكمله مرة أخرى. يجب عليك بالتأكيد النظر في المفاضلات بين تجربة المستخدم واستخدام واجهة برمجة التطبيقات عند اتخاذ قرار بشأن تنفيذ الاستمرار التلقائي في تطبيقك. يمكن أن يكون الاستمرار التلقائي

بشكل خاص مكلفاً بشكل خطير، خاصة عند استخدام النماذج التجارية المتميزة.

الخلاصة

تعد معالجة التدفق جانباً حاسماً في بناء التطبيقات المدعومة بالذكاء الاصطناعي التي تجمع بين استخدام الأدوات واستجابات الذكاء الاصطناعي المباشرة. من خلال التعامل الفعال مع البيانات المتدفقة من واجهات برمجة تطبيقات منصات الذكاء الاصطناعي، يمكنك توفير تجربة مستخدم سلسة وتفاعلية، والتعامل مع الاستجابات الكبيرة، وتحسين استخدام الموارد، والتعامل مع الأخطاء بسلاسة.

يوضح صنف Conversation::ReplyStream المقدم كيف يمكن تنفيذ معالجة التدفق في تطبيق Ruby باستخدام مطابقة الأنماط والبنية المعتمدة على الأحداث. من خلال فهم واستغلال تقنيات معالجة التدفق، يمكنك إطلاق الإمكانيات الكاملة لدمج الذكاء الاصطناعي في تطبيقاتك وتقديم تجارب مستخدم قوية وجذابة.

البيانات ذاتية التصحيح



تعد البيانات ذاتية التصحيح نهجاً قوياً لضمان سلامة البيانات واتساقها وجودتها في التطبيقات من خلال الاستفادة من قدرات نماذج اللغة الكبيرة (LLMs). تركز هذه الفئة من الأنماط على فكرة استخدام الذكاء الاصطناعي للكشف التلقائي عن شذوذ البيانات وعدم اتساقها أو أخطاءها وتشخيصها وتصحيحها، مما يقلل العبء على المطورين ويحافظ على مستوى عالٍ من موثوقية البيانات.

في جوهرها، تدرك أنماط البيانات ذاتية التصحيح أن البيانات هي شريان الحياة لأي تطبيق، وأن ضمان دقتها وسلامتها أمر حاسم للأداء السليم وتجربة المستخدم للتطبيق. ومع ذلك، يمكن أن تكون إدارة جودة البيانات والحفاظ عليها مهمة معقدة وتستغرق وقتاً طويلاً، خاصة مع نمو التطبيقات في الحجم والتعقيد. وهنا تأتي قوة الذكاء الاصطناعي.

في أنماط البيانات ذاتية التصحيح، يتم توظيف العمال الاصطناعيين لمراقبة وتحليل بيانات تطبيقك بشكل مستمر. تمتلك هذه النماذج القدرة على فهم وتفسير الأنماط والعلاقات والشذوذ داخل البيانات. من خلال الاستفادة من قدراتها في معالجة اللغة الطبيعية وفهمها، يمكنها تحديد المشكلات المحتملة أو عدم الاتساق في البيانات واتخاذ الإجراءات المناسبة لتصحيحها.

تتضمن عملية البيانات ذاتية التصحيح عدة خطوات رئيسية:

1. مراقبة البيانات: يقوم العمال الاصطناعيون بمراقبة تدفقات البيانات أو قواعد البيانات أو أنظمة التخزين الخاصة بالتطبيق باستمرار، بحثاً عن أي علامات للشذوذ أو عدم الاتساق أو الأخطاء. بدلاً من ذلك، يمكنك تنشيط مكون الذكاء الاصطناعي كرد فعل على استثناء.
2. اكتشاف الشذوذ: عند اكتشاف مشكلة، يقوم العامل الاصطناعي بتحليل البيانات بالتفصيل لتحديد طبيعة ونطاق المشكلة المحددة. قد يتضمن ذلك اكتشاف القيم المفقودة، أو التنسيقات غير المتسقة، أو البيانات التي تنتهك القواعد أو القيود المحددة مسبقاً.
3. التشخيص والتصحيح: بمجرد تحديد المشكلة، يستخدم العامل الاصطناعي معرفته وفهمه لمجال البيانات لتحديد مسار العمل المناسب. قد يتضمن ذلك تصحيح البيانات تلقائياً، أو ملء القيم المفقودة، أو تحديد المشكلة للتدخل البشري إذا لزم الأمر.
4. التعلم المستمر (اختياري، حسب حالة الاستخدام): عندما يواجه العامل الاصطناعي الخاص بك مشكلات البيانات المختلفة ويحلها، يمكنه إخراج ما حدث وكيف استجاب. يمكن تغذية هذه البيانات الوصفية في عمليات التعلم التي تسمح لك (وربما للنموذج الأساسي، من خلال الضبط الدقيق) بأن تصبح أكثر فعالية وكفاءة بمرور الوقت في تحديد وحل شذوذ البيانات.

من خلال الكشف التلقائي عن مشكلات البيانات وتصحيحها، يمكنك ضمان أن تطبيقك يعمل على بيانات عالية الجودة وموثوقة. هذا يقلل من مخاطر الأخطاء وعدم الاتساق أو الأخطاء المتعلقة بالبيانات التي تؤثر على وظائف التطبيق أو تجربة المستخدم.

بمجرد أن يتولى العمال الاصطناعيون مهمة مراقبة البيانات وتصحيحها، يمكنك تركيز جهودك على الجوانب الحرجة الأخرى للتطبيق. هذا يوفر الوقت والموارد التي كان من الممكن إنفاقها على تنظيف البيانات وصيانتها يدوياً. في الواقع، مع نمو تطبيقاتك في الحجم والتعقيد، تصبح إدارة جودة البيانات يدوياً أكثر تحدياً. تتوسع أنماط "البيانات ذاتية التصحيح" بشكل فعال من خلال الاستفادة من قوة الذكاء الاصطناعي للتعامل مع كميات كبيرة من البيانات واكتشاف المشكلات في الوقت الفعلي.



بسبب طبيعتها، يمكن لنماذج الذكاء الاصطناعي التكيف مع أنماط البيانات المتغيرة أو المخططات أو المتطلبات بمرور الوقت مع إشراف قليل أو معدوم. طالما أن توجيهاتهم توفر إرشادات كافية، خاصة فيما يتعلق بالنتائج المقصودة، قد يكون تطبيقك قادراً على التطور والتعامل مع سيناريوهات البيانات الجديدة دون الحاجة إلى تدخل يدوي واسع أو تغييرات في الكود.

تتوافق أنماط البيانات ذاتية التصحيح بشكل جيد مع الفئات الأخرى من الأنماط التي ناقشناها، مثل “تعدد العمال”. يمكن اعتبار قدرة البيانات ذاتية التصحيح نوعاً متخصصاً من العمال الذي يركز تحديداً على ضمان جودة البيانات وسلامتها. يعمل هذا النوع من العمال جنباً إلى جنب مع عمال الذكاء الاصطناعي الآخرين، حيث يساهم كل منهم في جوانب مختلفة من وظائف التطبيق.

يتطلب تنفيذ أنماط البيانات ذاتية التصحيح في الممارسة العملية تصميمًا دقيقاً ودمج نماذج الذكاء الاصطناعي في هيكل التطبيق. بسبب مخاطر فقدان البيانات وتلفها، يجب عليك تحديد إرشادات واضحة لكيفية استخدام هذه التقنية. يجب عليك أيضاً مراعاة عوامل مثل الأداء وقابلية التوسع وأمن البيانات.

دراسة حالة عملية: إصلاح JSON المعطوب

إحدى أكثر الطرق العملية والمريحة للاستفادة من البيانات ذاتية التصحيح هي أيضاً بسيطة جداً للشرح: إصلاح JSON المعطوب.

يمكن تطبيق هذه التقنية على التحدي الشائع المتمثل في التعامل مع البيانات غير المثالية أو غير المتسقة التي تنتجها نماذج اللغة الكبيرة، مثل JSON المعطوب، وتوفر نهجاً للكشف التلقائي عن هذه المشكلات وتصحيحها.

في Olympia أواجه بشكل منتظم سيناريوهات حيث تقوم نماذج LLM بإنشاء بيانات JSON غير صالحة بشكل تام. يمكن أن يحدث هذا لأسباب متعددة، مثل إضافة النموذج لتعليقات قبل أو بعد كود JSON الفعلي، أو إدخال أخطاء في الصياغة مثل الفواصل المفقودة أو علامات التنصيص المزدوجة غير المعالجة. يمكن أن تؤدي هذه المشاكل إلى أخطاء في التحليل وتسبب اضطرابات في وظائف التطبيق.

لمعالجة هذه المشكلة، قمت بتنفيذ حل عملي على شكل فئة JsonFixer. تجسد هذه الفئة نمط “البيانات ذاتية التصحيح” من خلال أخذ JSON المعطوب كمدخل واستخدام LLM لإصلاحه مع الحفاظ على أكبر قدر ممكن من المعلومات والقصد.

```

1  class JsonFixer
2      include Raix::ChatCompletion
3
4      def call(bad_json, error_message)
5          raise "provided data No" if bad_json.blank? || error_message.blank?
6
7          transcript << {
8              system: "parse a generated that JSON user-provided Consider
9              the preserving while it fix to best your Do exception.
10             possible. as much as intent and content original          " }
11             transcript << { user: bad_json }
12             transcript << { assistant: "message? error the is What"}
13             transcript << { user: error_message }
14             transcript << { assistant: "JSON corrected the is Here\n`json``\n" }
15
16             self.stop = ["````"]
17
18             chat_completion(json: true)
19         end
20
21         def model
22             "mistralai/mixtral-8x7b-instruct:nitro"
23         end
24     end

```

لاحظ كيف يستخدم JsonFixer أداة Ventriloquist لتوجيه استجابات الذكاء الاصطناعي.



تعمل عملية تصحيح بيانات JSON ذاتياً على النحو التالي:

1. إنشاء JSON: يتم استخدام نموذج اللغة الكبير (LLM) لإنشاء بيانات JSON بناءً على موجهات أو متطلبات معينة. ومع ذلك، ونظراً لطبيعة نماذج اللغة الكبيرة، قد لا تكون بيانات JSON المنشأة صحيحة دائماً. سيقوم محلل JSON بالطبع برفع خطأ ParserError إذا قدمت له JSON غير صالح.

```

1 begin
2   JSON.parse(llm_generated_json)
3 rescue JSON::ParserError => e
4   JsonFixer.new.call(llm_generated_json, e.message)
5 end

```

لاحظ أن رسالة الخطأ يتم تمريرها أيضاً إلى استدعاء JSONFixer بحيث لا يحتاج إلى افتراض ما هو خطأ في البيانات بشكل كامل، خاصة أن المحلل غالباً ما يخبرك بالضبط ما هو الخطأ.

2. **التصحيح القائم على LLM:** يقوم صنف JSONFixer بإرسال JSON المعطوب مرة أخرى إلى LLM، مع توجيه أو تعليمات محددة لإصلاح JSON مع الحفاظ على المعلومات والقصد الأصلي قدر الإمكان. يحاول LLM المدرب على كميات هائلة من البيانات ومع فهم تركيب JSON، تصحيح الأخطاء وإنشاء سلسلة JSON صالحة. يتم استخدام [تسييج الاستجابة](#) لتقييد مخرجات LLM، ونختار Mixtral 8x7B كنموذج للذكاء الاصطناعي، حيث أنه جيد بشكل خاص لهذا النوع من المهام.

3. **التحقق والتكامل:** يتم تحليل سلسلة JSON المصححة التي يعيدها LLM بواسطة صنف JSONFixer نفسه، لأنه استدعى completion(json:_chat, true). إذا اجتاز JSON المصحح عملية التحقق، يتم دمجها مرة أخرى في سير عمل التطبيق، مما يسمح للتطبيق بمواصلة معالجة البيانات بسلاسة. لقد تم "شفاء" JSON المعطوب.

على الرغم من أنني كتبت وأعدت كتابة تنفيذي الخاص لـ JSONFixer عدة مرات، أشك في أن إجمالي الوقت المستثمر في جميع تلك الإصدارات يزيد عن ساعة أو ساعتين.

لاحظ أن الحفاظ على القصد هو عنصر أساسي في أي نمط للبيانات ذاتية التصحيح. تهدف عملية التصحيح القائمة على LLM إلى الحفاظ على المعلومات والقصد الأصلي لـ JSON المنشأ قدر الإمكان. هذا يضمن أن JSON المصحح يحافظ على معناه الدلالي ويمكن استخدامه بفعالية ضمن سياق التطبيق.

يوضح هذا التنفيذ العملي لنهج "البيانات ذاتية التصحيح" في Olympia بوضوح كيف يمكن الاستفادة من الذكاء الاصطناعي، وخاصة LLMs، لحل تحديات البيانات في العالم الحقيقي. إنه يظهر قوة الجمع بين تقنيات البرمجة التقليدية وقدرات الذكاء الاصطناعي لبناء تطبيقات قوية وفعالة.

قانون بوستل ونمط "البيانات ذاتية التصحيح"

يتوافق "البيانات ذاتية التصحيح"، كما يتجلى في صنف JSONFixer بشكل جيد مع المبدأ المعروف باسم قانون بوستل، والذي يشار إليه أيضاً باسم مبدأ المتانة. ينص قانون بوستل على:

"كن محافظاً فيما تفعله، وكن متساهلاً فيما تقبله من الآخرين."

هذا المبدأ، الذي صاغه جون بوستل، أحد رواد الإنترنت الأوائل، يؤكد على أهمية بناء أنظمة تتسامح مع المدخلات المتنوعة أو حتى الخاطئة قليلاً مع الحفاظ على الالتزام الصارم بالبروتوكولات المحددة عند إرسال المخرجات.

في سياق "البيانات ذاتية التصحيح"، يجسد صنف JSONFixer قانون بوستل من خلال كونه متساهلاً في قبول بيانات JSON المعطوبة أو غير المثالية التي تنتجها LLMs. لا يرفض أو يفشل على الفور عند مواجهة JSON لا يتوافق بدقة مع التنسيق المتوقع. بدلاً من ذلك، يتخذ نهجاً متساهلاً ويحاول إصلاح JSON باستخدام قوة LLMs.

من خلال كونه متساهلاً في قبول JSON غير المثالي، يظهر صنف JSONFixer المتانة والمرونة. إنه يقر بأن البيانات في العالم الحقيقي غالباً ما تأتي في أشكال مختلفة وقد لا تتوافق دائماً مع المواصفات الصارمة. من خلال التعامل مع هذه الانحرافات وتصحيحها بأناقة، يضمن الصنف أن التطبيق يمكن أن يستمر في العمل بسلاسة، حتى في وجود بيانات غير مثالية.

من ناحية أخرى، يلتزم صنف JSONFixer أيضاً بالجانب المحافظ من قانون بوستل عندما يتعلق الأمر بالمخرجات. بعد إصلاح JSON باستخدام LLMs، يتحقق الصنف من JSON المصحح للتأكد من أنه يتوافق بدقة مع التنسيق المتوقع. يحافظ على سلامة وصحة البيانات قبل تمريرها إلى أجزاء أخرى من التطبيق. يضمن هذا النهج المحافظ أن مخرجات صنف JSONFixer موثوقة ومتسقة، مما يعزز قابلية التشغيل البيني ويمنع انتشار الأخطاء.

معلومات مثيرة للاهتمام عن جون بوستل:

- كان جون بوستل (1943-1998) عالم كمبيوتر أمريكي لعب دوراً حاسماً في تطوير الإنترنت. كان يُعرف باسم "إله الإنترنت" لمساهماته المهمة في البروتوكولات والمعايير الأساسية.

- كان بوستل محرر سلسلة وثائق طلبات التعليقات (RFC) وهي سلسلة من الملاحظات التقنية والتنظيمية حول الإنترنت. قام بتأليف أو المشاركة في تأليف أكثر من 200 RFC، بما في ذلك البروتوكولات الأساسية مثل TCP وIP وSMTP.
- بالإضافة إلى مساهماته التقنية، كان بوستل معروفاً بنهجه المتواضع والتعاوني. كان يؤمن بأهمية التوصل إلى توافق في الآراء والعمل معاً لبناء شبكة قوية وقابلة للتشغيل البيئي.
- شغل بوستل منصب مدير قسم شبكات الكمبيوتر في معهد علوم المعلومات (ISI) في جامعة جنوب كاليفورنيا (USC) من عام 1977 حتى وفاته المبكرة في عام 1998.
- تقديراً لمساهماته الهائلة، حصل بوستل بعد وفاته على جائزة تورنج المرموقة في عام 1998، والتي يشار إليها غالباً باسم "جائزة نوبل في الحوسبة".

يعزز صنف JSONFixer المرونة وقابلية التشغيل البيئي، وهي القيم الأساسية التي تمسك بها بوستل طوال مسيرته المهنية. من خلال بناء أنظمة تتسامح مع العيوب مع الحفاظ على الالتزام الصارم بالبروتوكولات، يمكننا إنشاء تطبيقات أكثر مرونة وقدرة على التكيف في مواجهة تحديات العالم الحقيقي.

الاعتبارات وموانع الاستخدام

يعتمد تطبيق نهج البيانات ذاتية الإصلاح بشكل كامل على نوع البيانات التي يتعامل معها تطبيقك. هناك سبب يجعلك قد لا ترغب في مجرد استخدام التصحيح الديناميكي لـ JSON.parse للتصحيح التلقائي لجميع أخطاء تحليل JSON في تطبيقك: ليست كل الأخطاء يمكن أو ينبغي تصحيحها تلقائياً.

يصبح الإصلاح الذاتي معقداً بشكل خاص عند اقترانه بمتطلبات تنظيمية أو أمثال متعلقة بمعالجة البيانات ومعالجتها. بعض الصناعات، مثل الرعاية الصحية والتمويل، لديها لوائح صارمة للغاية فيما يتعلق بإسلامة البيانات وقابلية التدقيق لدرجة أن إجراء أي نوع من تصحيح البيانات "بالصندوق الأسود" دون إشراف أو تسجيل مناسب قد ينتهك هذه اللوائح. من الضروري ضمان توافق أي تقنيات بيانات ذاتية الإصلاح تطورها مع الأطر القانونية والتنظيمية المعمول بها.

قد يكون لتطبيق تقنيات البيانات ذاتية الإصلاح، خاصة تلك التي تتضمن نماذج الذكاء الاصطناعي، تأثير كبير أيضاً على أداء التطبيق واستخدام الموارد. يمكن أن تكون معالجة كميات كبيرة من البيانات

من خلال نماذج الذكاء الاصطناعي للكشف عن الأخطاء وتصحيحها كثيفة الاستخدام للحوسبة. من المهم تقييم المفاضلات بين فوائد البيانات ذاتية الإصلاح وتكاليف الأداء والموارد المرتبطة بها. ومع ذلك، دعونا نتعمق في العوامل المتضمنة في تحديد متى وأين نطبق هذا النهج القوي.

حساسية البيانات

عند النظر في تطبيق تقنيات البيانات ذاتية الإصلاح، من الضروري تقييم حساسية البيانات التي تتم معالجتها. يشير مستوى الحساسية إلى أهمية وحساسية البيانات في سياق تطبيقك ومجال عملك. في بعض الحالات، قد لا يكون التصحيح التلقائي لأخطاء البيانات مناسباً، خاصة إذا كانت البيانات شديدة الحساسية أو لها آثار قانونية. على سبيل المثال، انظر في السيناريوهات التالية:

1. المعاملات المالية: في التطبيقات المالية، مثل الأنظمة المصرفية أو منصات التداول، تعد دقة البيانات ذات أهمية قصوى. حتى الأخطاء البسيطة في البيانات المالية يمكن أن يكون لها عواقب كبيرة، مثل أرصدة الحسابات غير الصحيحة، أو تحويل الأموال بشكل خاطئ، أو قرارات التداول الخاطئة. في هذه الحالات، قد تؤدي التصحيحات التلقائية دون تحقق وتدقيق شامل إلى مخاطر غير مقبولة.
2. السجلات الطبية: تتعامل تطبيقات الرعاية الصحية مع بيانات المرضى شديدة الحساسية والسرية. يمكن أن يكون للأخطاء في السجلات الطبية آثار خطيرة على سلامة المرضى وقرارات العلاج. قد ينتهك التعديل التلقائي للبيانات الطبية دون إشراف وتحقق مناسب من قبل متخصصي الرعاية الصحية المؤهلين المتطلبات التنظيمية ويعرض رفاة المريض للخطر.
3. المستندات القانونية: تتطلب التطبيقات التي تتعامل مع المستندات القانونية، مثل العقود والاتفاقيات أو الملفات القضائية، دقة وسلامة صارمة. حتى الأخطاء البسيطة في البيانات القانونية يمكن أن يكون لها تداعيات قانونية كبيرة. قد لا تكون التصحيحات التلقائية في هذا المجال مناسبة، حيث غالباً ما تتطلب البيانات مراجعة يدوية وتحققاً من قبل خبراء قانونيين لضمان صحتها وقابليتها للتنفيذ.

في هذه السيناريوهات الحرجة للبيانات، غالباً ما تفوق المخاطر المرتبطة بالتصحيحات التلقائية الفوائد المحتملة. يمكن أن تكون عواقب إدخال أخطاء أو تعديل البيانات بشكل غير صحيح خطيرة، مما يؤدي إلى خسائر مالية أو مسؤوليات قانونية أو حتى ضرر للأفراد.

عند التعامل مع البيانات شديدة الحساسية، من الضروري إعطاء الأولوية لعمليات التحقق والتحقق اليدوي. الإشراف البشري والخبرة أمران حاسمان في ضمان دقة وسلامة البيانات. لا يزال من الممكن

استخدام تقنيات الإصلاح الذاتي للإشارة إلى الأخطاء أو التناقضات المحتملة، ولكن يجب أن يتضمن القرار النهائي بشأن التصحيحات الحكم والموافقة البشرية.

ومع ذلك، من المهم ملاحظة أنه ليس كل البيانات في التطبيق قد يكون لها نفس مستوى الحساسية. داخل نفس التطبيق، قد تكون هناك مجموعات فرعية من البيانات أقل حساسية أو لها تأثير أقل إذا حدثت أخطاء. في مثل هذه الحالات، يمكن تطبيق تقنيات البيانات ذاتية الإصلاح بشكل انتقائي على تلك المجموعات الفرعية المحددة من البيانات، بينما تظل البيانات الحساسة خاضعة للتحقق اليدوي.

المفتاح هو تقييم حساسية كل فئة من البيانات في تطبيقك بعناية وتحديد إرشادات وعمليات واضحة للتعامل مع التصحيحات بناءً على المخاطر والآثار المرتبطة بها. من خلال التمييز بين البيانات الحساسة (مثل دفاتر الحسابات والسجلات الطبية) والبيانات غير الحساسة (مثل عناوين البريد وتحذيرات الموارد)، يمكنك تحقيق التوازن بين الاستفادة من مزايا تقنيات البيانات ذاتية الإصلاح حيثما كان ذلك مناسباً والحفاظ على التحكم والإشراف الصارم حيثما كان ذلك ضرورياً.

في النهاية، يجب اتخاذ قرار تطبيق تقنيات البيانات ذاتية الإصلاح على البيانات الحساسة بالتشاور مع خبراء المجال والمستشارين القانونيين وأصحاب المصلحة المعنيين الآخرين. من الضروري النظر في المتطلبات واللوائح والمخاطر المحددة المرتبطة ببيانات تطبيقك ومواءمة استراتيجيات تصحيح البيانات وفقاً لذلك.

خطورة الأخطاء

عند تطبيق تقنيات البيانات ذاتية الإصلاح، من المهم تقييم خطورة وتأثير أخطاء البيانات. ليست كل الأخطاء متساوية، وقد يختلف مسار العمل المناسب اعتماداً على خطورة المشكلة.

قد تكون التناقضات البسيطة أو مشاكل التنسيق مناسبة للتصحيح التلقائي. على سبيل المثال، يمكن لمعامل البيانات ذاتي الإصلاح المكلف بإصلاح JSON المعطوب التعامل مع الفواصل المفقودة أو علامات الاقتباس المزدوجة غير المحمية دون تغيير كبير في معنى أو هيكل البيانات. هذه الأنواع من الأخطاء غالباً ما يكون من السهل تصحيحها ولها تأثير ضئيل على سلامة البيانات الشاملة.

ومع ذلك، فإن الأخطاء الأكثر خطورة التي تغير بشكل جوهري معنى البيانات أو سلامتها قد تتطلب نهجاً مختلفاً. في مثل هذه الحالات، قد لا تكون التصحيحات الآلية كافية، وقد يكون التدخل البشري ضرورياً لضمان دقة البيانات وصحتها.

وهنا يأتي دور مفهوم استخدام الذكاء الاصطناعي نفسه للمساعدة في تحديد شدة الخطأ. من خلال الاستفادة من قدرات نماذج الذكاء الاصطناعي، يمكننا تصميم عمال معالجة البيانات ذاتية التصحيح التي

لا تقوم فقط بتصحيح الأخطاء، بل تقوم أيضاً بتقييم شدة هذه الأخطاء واتخاذ قرارات مدروسة حول كيفية التعامل معها.

على سبيل المثال، دعونا نتناول عامل معالجة بيانات ذاتي التصحيح مسؤول عن تصحيح التناقضات في البيانات المتدفقة إلى قاعدة بيانات العملاء. يمكن تصميم العامل لتحليل البيانات وتحديد الأخطاء المحتملة، مثل المعلومات المفقودة أو المتضاربة. ومع ذلك، بدلاً من تصحيح جميع الأخطاء تلقائياً، يمكن تزويد العامل باستدعاءات أدوات إضافية تتيح له تمييز الأخطاء الخطيرة للمراجعة البشرية.

واليك مثال على كيفية تنفيذ ذلك:

```

1 class CustomerDataReviewer
2   include Raix::ChatCompletion
3   include Raix::FunctionDeclarations
4
5   attr_accessor :customer
6
7   function :flag_for_review, reason: { type: "string" } do |params|
8     AdminNotifier.review_request(customer, params[:reason])
9   end
10
11  def initialize(customer)
12    self.customer = customer
13  end
14
15  def call(customer_data)
16    transcript << {
17      system: "identify to is task Your reviewer. data customer a are You
18  data. customer in inconsistencies correct and
19
20  > here... instructions additional <
21
22  the use ,review human require that errors severe encounter you If
23  intervention. manual for data the flag to tool ``flag_for_review      " }
24
25    transcript << { user: customer.to_json }
26    transcript << { assistant: "data: Reviewed/corrected\n`json``\n" }
27
28    self.stop = ["````"]
29
30    chat_completion(json: true).then do |result|

```

```

31     return if result.blank?
32
33     customer.update(result)
34   end
35 end
36 end

```

في هذا المثال، تم تصميم عامل CustomerDataHealer لتحديد وتصحيح التناقضات في بيانات العملاء. مرة أخرى، نستخدم [تسييج الاستجابة وبرنامج المحاكاة الصوتية](#) للحصول على مخرجات منظمة. ومن المهم الإشارة إلى أن التوجيه النظامي للعامل يتضمن تعليمات لاستخدام دالة review_for_flag في حالة مواجهة أخطاء خطيرة.

عندما يقوم العامل بمعالجة بيانات العملاء، فإنه يحلل البيانات ويحاول تصحيح أي تناقضات. إذا قرر العامل أن الأخطاء خطيرة وتتطلب تدخلاً بشرياً، يمكنه استخدام أداة review_for_flag لتمييز البيانات وتقديم سبب لهذا التمييز.

يتم استدعاء طريقة completion_chat مع true json: لتحليل بيانات العملاء المصححة كـ JSON. لا يوجد أي إعداد للتكرار بعد استدعاء الدالة، لذا ستكون النتيجة فارغة إذا تم استدعاء review_for_flag. وإلا، يتم تحديث بيانات العميل بالبيانات التي تمت مراجعتها وتصحيحها محتملاً.

من خلال دمج تقييم خطورة الأخطاء وخيار تمييز البيانات للمراجعة البشرية، يصبح عامل البيانات ذاتي التصحيح أكثر ذكاءً وقابلية للتكيف. يمكنه معالجة الأخطاء البسيطة تلقائياً مع تصعيد الأخطاء الخطيرة إلى الخبراء البشريين للتدخل اليدوي.

يمكن تحديد المعايير المحددة لتحديد خطورة الأخطاء في توجيه العامل بناءً على المعرفة بالجمال ومتطلبات العمل. يمكن النظر في عوامل مثل التأثير على سلامة البيانات، واحتمالية فقدان البيانات أو تلفها، وعواقب البيانات غير الصحيحة عند تقييم الخطورة.

من خلال الاستفادة من الذكاء الاصطناعي لتقييم خطورة الأخطاء وتوفير خيارات للتدخل البشري، يمكن لتقنيات البيانات ذاتية التصحيح تحقيق التوازن بين الأتمتة والحفاظ على دقة البيانات. يضمن هذا النهج تصحيح الأخطاء البسيطة بكفاءة في حين تتلقى الأخطاء الخطيرة الاهتمام والخبرة اللازمة من المراجعين البشريين.

تعقيد النطاق

عند النظر في تطبيق تقنيات البيانات ذاتية التصحيح، من المهم تقييم تعقيد نطاق البيانات والقواعد التي تحكم هيكلها وعلاقاتها. يمكن أن يؤثر تعقيد النطاق بشكل كبير على فعالية وجدوى نهج التصحيح الآلي للبيانات.

تعمل تقنيات البيانات ذاتية التصحيح بشكل جيد عندما تتبع البيانات أنماطاً وقيوداً محددة بوضوح. في النطاقات التي يكون فيها هيكل البيانات بسيطاً نسبياً والعلاقات بين عناصر البيانات مباشرة، يمكن تطبيق التصحيحات الآلية بدرجة عالية من الثقة. على سبيل المثال، يمكن معالجة تصحيح مشكلات التنسيق أو فرض قيود أنواع البيانات الأساسية بفعالية من قبل عمال البيانات ذاتية التصحيح.

ومع ذلك، مع زيادة تعقيد نطاق البيانات، تزداد أيضاً التحديات المرتبطة بالتصحيح الآلي للبيانات. في النطاقات ذات المنطق التجاري المعقد، والعلاقات المعقدة بين كيانات البيانات، أو القواعد والاستثناءات الخاصة بالنطاق، قد لا تلتقط تقنيات البيانات ذاتية التصحيح دائماً الفروق الدقيقة وقد تؤدي إلى عواقب غير مقصودة.

دعونا نأخذ مثالاً على نطاق معقد: نظام التداول المالي. في هذا النطاق، تتضمن البيانات أدوات مالية متنوعة، وبيانات السوق، وقواعد التداول، والمتطلبات التنظيمية. يمكن أن تكون العلاقات بين عناصر البيانات المختلفة معقدة، ويمكن أن تكون القواعد التي تحكم صحة البيانات واتساقها خاصة جداً بالنطاق.

في مثل هذا النطاق المعقد، سيحتاج عامل البيانات ذاتي التصحيح المكلف بتصحيح التناقضات في بيانات التداول إلى فهم عميق للقواعد والقيود الخاصة بالنطاق. سيحتاج إلى النظر في عوامل مثل لوائح السوق، وحدود التداول، وحسابات المخاطر، وإجراءات التسوية. قد لا تلتقط التصحيحات الآلية في هذا السياق دائماً التعقيد الكامل للنطاق وقد تؤدي عن غير قصد إلى إدخال أخطاء أو انتهاك قواعد خاصة بالنطاق.

لمعالجة تحديات تعقيد النطاق، يمكن تعزيز تقنيات البيانات ذاتية التصحيح من خلال دمج المعرفة والقواعد الخاصة بالنطاق في نماذج وعمال الذكاء الاصطناعي. يمكن تحقيق ذلك من خلال تقنيات مثل:

1. التدريب المخصص للنطاق: يمكن توجيه نماذج الذكاء الاصطناعي المستخدمة للبيانات ذاتية التصحيح أو حتى ضبطها على مجموعات البيانات الخاصة بالنطاق التي تلتقط تعقيدات وقواعد النطاق المعين. من خلال تعريض النماذج للبيانات والسيناريوهات التمثيلية، يمكنها تعلم الأنماط والقيود والاستثناءات الخاصة بالنطاق.

2. القيود المستندة إلى القواعد: يمكن تعزيز عمال البيانات ذاتية التصحيح بقيود صريحة مستندة إلى القواعد تشفر المعرفة الخاصة بالنطاق. يمكن تحديد هذه القواعد من قبل خبراء المجال ودمجها في عملية تصحيح البيانات. يمكن لنماذج الذكاء الاصطناعي بعد ذلك استخدام هذه القواعد لتوجيه قراراتها وضمان الامتثال لمتطلبات النطاق المحددة.
3. التعاون مع خبراء المجال: في النطاقات المعقدة، من الضروري إشراك خبراء المجال في تصميم وتطوير تقنيات البيانات ذاتية التصحيح. يمكن لخبراء المجال تقديم رؤى قيمة حول تعقيدات البيانات، وقواعد العمل، والحالات الحدية المحتملة. يمكن دمج معرفتهم في نماذج وعمال الذكاء الاصطناعي لتحسين دقة وموثوقية التصحيحات الآلية للبيانات باستخدام أنماط **التدخل البشري في العملية**.
4. نهج تدريجي وتكراري: عند التعامل مع النطاقات المعقدة، غالباً ما يكون من المفيد تبني نهج تدريجي وتكراري للبيانات ذاتية التصحيح. بدلاً من محاولة أتمتة التصحيحات للنطاق بأكمله دفعة واحدة، ركز على نطاقات فرعية محددة أو فئات البيانات حيث تكون القواعد والقيود مفهومة جيداً. قم بتوسيع نطاق تقنيات التصحيح الذاتي تدريجياً مع نمو فهم النطاق وإثبات فعالية التقنيات.

من خلال النظر في تعقيد مجال البيانات ودمج المعرفة المتخصصة بالمجال في تقنيات البيانات ذاتية التصحيح، يمكنك تحقيق التوازن بين الأتمتة والدقة. من المهم إدراك أن البيانات ذاتية التصحيح ليست حلاً يناسب جميع الحالات، وأنه يجب تكييف النهج وفقاً للمتطلبات والتحديات الخاصة بكل مجال.

في المجالات المعقدة، قد يكون النهج الهجين الذي يجمع بين تقنيات البيانات ذاتية التصحيح والخبرة البشرية والإشراف هو الأكثر فعالية. يمكن للتصحيحات الآلية معالجة الحالات الروتينية والمحددة بوضوح، بينما يمكن تمييز السيناريوهات المعقدة أو الاستثناءات للمراجعة والتدخل البشري. يضمن هذا النهج التعاوني تحقيق فوائد الأتمتة مع الحفاظ على التحكم والدقة اللازمين في مجالات البيانات المعقدة.

القابلية للتفسير والشفافية

تشير القابلية للتفسير إلى القدرة على فهم وتفسير المنطق وراء القرارات التي تتخذها نماذج الذكاء الاصطناعي، بينما تتضمن الشفافية توفير رؤية واضحة لعملية تصحيح البيانات.

في العديد من السياقات، تحتاج تعديلات البيانات إلى أن تكون قابلة للتدقيق والتبرير. قد يتطلب أصحاب المصلحة، بما في ذلك مستخدمو الأعمال والمدققون والهيئات التنظيمية، تفسيرات لسبب إجراء تصحيحات معينة للبيانات وكيفية توصل نماذج الذكاء الاصطناعي إلى تلك القرارات. هذا أمر بالغ الأهمية خاصة

في المجالات التي تكون فيها دقة البيانات وسلامتها ذات آثار كبيرة، مثل التمويل والرعاية الصحية والمسائل القانونية.

لمعالجة الحاجة إلى القابلية للتفسير والشفافية، يجب أن تتضمن تقنيات البيانات ذاتية التصحيح آليات توفر رؤى حول عملية صنع القرار لنماذج الذكاء الاصطناعي. يمكن تحقيق ذلك من خلال مناهج مختلفة:

1. سلسلة التفكير: إن طلب النموذج شرح تفكيره "بصوت عالٍ" قبل تطبيق التغييرات على البيانات قد يسمح بفهم أسهل لعملية صنع القرار ويمكن أن يولد تفسيرات مفهومة للتصحيحات التي تم إجراؤها. المقايضة هي قليل من التعقيد الإضافي في فصل التفسير عن مخرجات البيانات المنظمة، والتي يمكن معالجتها من خلال...
2. توليد التفسيرات: يمكن تزويد عمال البيانات ذاتية التصحيح بالقدرة على توليد تفسيرات مفهومة للتصحيحات التي يقومون بها. يمكن تحقيق ذلك من خلال طلب النموذج إخراج عملية صنع القرار الخاصة به كتفسيرات سهلة الفهم مدمجة في البيانات نفسها. على سبيل المثال، يمكن لعامل البيانات ذاتي التصحيح إنشاء تقرير يسلط الضوء على عدم اتساق البيانات المحدد، والتصحيحات التي تم تطبيقها، والمنطق وراء تلك التصحيحات.
3. أهمية السمات: يمكن توجيه نماذج الذكاء الاصطناعي بمعلومات حول أهمية السمات أو الخصائص المختلفة في عملية تصحيح البيانات كجزء من توجيهاتها. يمكن بدورها عرض تلك التوجيهات لأصحاب المصلحة البشريين. من خلال تحديد العوامل الرئيسية التي تؤثر على قرارات النموذج، يمكن لأصحاب المصلحة اكتساب رؤى حول المنطق وراء التصحيحات وتقييم صحتها.
4. التسجيل والتدقيق: يعد تنفيذ آليات شاملة للتسجيل والتدقيق أمراً حاسماً للحفاظ على الشفافية في عملية البيانات ذاتية التصحيح. يجب تسجيل كل تصحيح للبيانات تقوم به نماذج الذكاء الاصطناعي، بما في ذلك البيانات الأصلية، والبيانات المصححة، والإجراءات المحددة المتخذة. يسمح بتسجيل التدقيق هذا بالتحليل بأثر رجعي ويوفر سجلاً واضحاً للتعديلات التي أجريت على البيانات.
5. نهج إشراك العنصر البشري في العملية: يمكن أن يعزز دمج نهج إشراك العنصر البشري في العملية القابلية للتفسير والشفافية لتقنيات البيانات ذاتية التصحيح. من خلال إشراك الخبراء البشريين في مراجعة والتحقق من صحة التصحيحات التي يولدها الذكاء الاصطناعي، يمكن للمؤسسات ضمان توافق التصحيحات مع المعرفة بالجمال ومتطلبات العمل. يضيف الإشراف البشري طبقة إضافية من المساءلة ويسمح بتحديد أي تحيزات أو أخطاء محتملة في نماذج الذكاء الاصطناعي.
6. المراقبة والتقييم المستمر: تعد المراقبة والتقييم المنتظم لأداء تقنيات البيانات ذاتية التصحيح أمراً ضرورياً للحفاظ على الشفافية والثقة. من خلال تقييم دقة وفعالية نماذج الذكاء الاصطناعي بمرور

الوقت، يمكن للمؤسسات تحديد أي انحرافات أو شذوذ واتخاذ الإجراءات التصحيحية. تساعد المراقبة المستمرة في ضمان بقاء عملية البيانات ذاتية التصحيح موثوقة ومتوافقة مع النتائج المرجوة.

تعد القابلية للتفسير والشفافية اعتبارات حاسمة عند تنفيذ تقنيات البيانات ذاتية التصحيح. من خلال تقديم تفسيرات واضحة لتصحيحات البيانات، والحفاظ على سجلات تدقيق شاملة، وإشراك الإشراف البشري، يمكن للمؤسسات بناء الثقة في عملية البيانات ذاتية التصحيح وضمان أن التعديلات التي تم إجراؤها على البيانات مبررة ومتوافقة مع أهداف العمل.

من المهم تحقيق التوازن بين فوائد الأتمتة والحاجة إلى الشفافية. في حين أن تقنيات البيانات ذاتية التصحيح يمكن أن تحسن بشكل كبير جودة البيانات والكفاءة، يجب ألا يأتي ذلك على حساب فقدان الرؤية والتحكم في عملية تصحيح البيانات. من خلال تصميم عمال البيانات ذاتية التصحيح مع وضع القابلية للتفسير والشفافية في الاعتبار، يمكن للمؤسسات تسخير قوة الذكاء الاصطناعي مع الحفاظ على المستوى اللازم من المساءلة والثقة في البيانات.

العواقب غير المقصودة

في حين أن تقنيات البيانات ذاتية التصحيح تهدف إلى تحسين جودة البيانات واتساقها، من الضروري أن نكون على دراية باحتمال حدوث عواقب غير مقصودة. قد تؤدي التصحيحات الآلية، إذا لم يتم تصميمها ومراقبتها بعناية، إلى تغيير معنى أو سياق البيانات بشكل غير مقصود، مما يؤدي إلى مشاكل لاحقة.

أحد المخاطر الرئيسية للبيانات ذاتية التصحيح هو إدخال التحيز أو الأخطاء في عملية تصحيح البيانات. يمكن أن تخضع نماذج الذكاء الاصطناعي، مثل أي نظام برمجي آخر، للتحيزات الموجودة في بيانات التدريب أو التي تم إدخالها من خلال تصميم الخوارزميات. إذا لم يتم تحديد هذه التحيزات والتخفيف من حدتها، يمكن أن تنتشر من خلال عملية البيانات ذاتية التصحيح وتؤدي إلى تعديلات بيانات منحرفة أو غير صحيحة.

على سبيل المثال، لنفترض عامل بيانات ذاتي التصحيح مكلف بتصحيح التناقضات في البيانات الديموغرافية للعملاء. إذا كان نموذج الذكاء الاصطناعي قد تعلم تحيزات من البيانات التاريخية، مثل ربط مهن معينة أو مستويات دخل محددة بأجناس أو أعراق معينة، فقد يقوم بافتراضات خاطئة وتعديل البيانات بطريقة تعزز تلك التحيزات. يمكن أن يؤدي هذا إلى ملفات تعريف غير دقيقة للعملاء، وقرارات تجارية مضللة، ونتائج قد تكون تمييزية.

ومن العواقب غير المقصودة المحتملة الأخرى فقدان معلومات أو سياق قيم أثناء عملية تصحيح البيانات. غالباً ما تركز تقنيات البيانات ذاتية التصحيح على توحيد وتطبيع البيانات لضمان الاتساق. ومع ذلك، في بعض الحالات، قد تحتوي البيانات الأصلية على دقائق أو استثناءات أو معلومات سياقية مهمة لفهم الصورة الكاملة. قد تؤدي التصحيحات الآلية التي تفرض التوحيد بشكل أعمى إلى إزالة أو إخفاء هذه المعلومات القيمة عن غير قصد.

على سبيل المثال، تخيل عامل بيانات ذاتي التصحيح مسؤول عن تصحيح التناقضات في السجلات الطبية. إذا واجه العامل تاريخاً طبياً لمرضى يعاني من حالة نادرة أو خطة علاج غير معتادة، فقد يحاول تطبيع البيانات لتتناسب مع نمط أكثر شيوعاً. ومع ذلك، في هذه العملية، قد يفقد التفاصيل والسياق المحددين اللذين يعتبران حاسمين للتمثيل الدقيق لحالة المريض الفريدة. يمكن أن يكون لفقدان هذه المعلومات آثار خطيرة على رعاية المريض واتخاذ القرارات الطبية.

للتخفيف من مخاطر العواقب غير المقصودة، من الضروري اتباع نهج استباقي عند تصميم وتنفيذ تقنيات البيانات ذاتية التصحيح:

1. **الاختبار والتحقق الشامل:** قبل نشر عمال البيانات ذاتية التصحيح في بيئة الإنتاج، من الضروري اختبار سلوكهم والتحقق منه بشكل شامل مقابل مجموعة متنوعة من السيناريوهات. يتضمن ذلك الاختبار باستخدام مجموعات بيانات تمثيلية تغطي مختلف الحالات الاستثنائية والاستثناءات والتحيزات المحتملة. يساعد الاختبار الدقيق في تحديد ومعالجة أي عواقب غير مقصودة قبل أن تؤثر على البيانات في العالم الحقيقي.

2. **المراقبة والتقييم المستمر:** يعد تنفيذ آليات المراقبة والتقييم المستمر أمراً ضرورياً لاكتشاف وتخفيف العواقب غير المقصودة مع مرور الوقت. يمكن أن تساعد المراجعة المنتظمة لنتائج عمليات البيانات ذاتية التصحيح، وتحليل التأثير على الأنظمة التابعة وصنع القرار، وجمع التغذية الراجعة من أصحاب المصلحة في تحديد أي آثار سلبية واتخاذ إجراءات تصحيحية في الوقت المناسب. إذا كان لدى مؤسستك لوحات متابعة تشغيلية، فمن المحتمل أن تكون إضافة مقاييس واضحة تتعلق بتغييرات البيانات الآلية فكرة جيدة. كما أن إضافة إنذارات مرتبطة بالانحرافات الكبيرة عن نشاط تغيير البيانات الطبيعي قد تكون فكرة أفضل!

3. **الإشراف والتدخل البشري:** الحفاظ على الإشراف البشري والقدرة على التدخل في عملية البيانات ذاتية التصحيح أمر بالغ الأهمية. في حين أن الأتمتة يمكن أن تحسن الكفاءة بشكل كبير، من المهم أن يقوم الخبراء البشريون بمراجعة والتحقق من التصحيحات التي تقوم بها نماذج الذكاء

الاصطناعي، خاصة في المجالات الحساسة أو الحرجة. يمكن للحكم البشري والخبرة في المجال أن يساعدوا في تحديد ومعالجة أي عواقب غير مقصودة قد تنشأ.

4. الذكاء الاصطناعي القابل للتفسير (XAI) والشفافية: كما نوقش في القسم الفرعي السابق، يمكن أن يساعد دمج تقنيات الذكاء الاصطناعي القابل للتفسير وضمان الشفافية في عملية البيانات ذاتية التصحيح في تخفيف العواقب غير المقصودة. من خلال تقديم تفسيرات واضحة لتصحيحات البيانات والحفاظ على مسارات تدقيق شاملة، يمكن للمؤسسات فهم وتتبع المنطق وراء التعديلات التي تقوم بها نماذج الذكاء الاصطناعي بشكل أفضل.

5. النهج التدريجي والتكراري: يمكن أن يساعد تبني نهج تدريجي وتكراري للبيانات ذاتية التصحيح في تقليل مخاطر العواقب غير المقصودة. بدلاً من تطبيق التصحيحات الآلية على مجموعة البيانات بأكملها دفعة واحدة، ابدأ بمجموعة فرعية من البيانات وقم بتوسيع النطاق تدريجياً مع إثبات فعالية وموثوقية التقنيات. يسمح هذا بالمراقبة الدقيقة والتعديل على طول الطريق، مما يقلل من تأثير أي عواقب غير مقصودة.

6. التعاون والتغذية الراجعة: يمكن أن يساعد إشراك أصحاب المصلحة من مختلف المجالات وتشجيع التعاون والتغذية الراجعة طوال عملية البيانات ذاتية التصحيح في تحديد ومعالجة العواقب غير المقصودة. يمكن أن يوفر طلب المدخلات بانتظام من خبراء المجال ومستهلكي البيانات والمستخدمين النهائيين رؤى قيمة حول التأثير الواقعي لتصحيحات البيانات وتبسيط الضوء على أي مشكلات قد تم تجاهلها.

من خلال معالجة مخاطر العواقب غير المقصودة بشكل استباقي وتنفيذ الضمانات المناسبة، يمكن للمؤسسات الاستفادة من مزايا تقنيات البيانات ذاتية التصحيح مع تقليل الآثار السلبية المحتملة. من المهم التعامل مع البيانات ذاتية التصحيح كعملية تكرارية وتعاونية، مع المراقبة والتقييم والتحسين المستمر للتقنيات لضمان توافيقها مع النتائج المرجوة والحفاظ على سلامة البيانات وموثوقيتها.

عند النظر في استخدام أنماط البيانات ذاتية التصحيح، من الضروري تقييم هذه العوامل بعناية وموازنة الفوائد مقابل المخاطر والقيود المحتملة. في بعض الحالات، قد يكون النهج الهجين الذي يجمع بين التصحيحات الآلية والإشراف والتدخل البشري هو الحل الأنسب.

من الجدير بالذكر أيضاً أن تقنيات البيانات ذاتية التصحيح لا ينبغي اعتبارها بديلاً عن آليات التحقق القوية من صحة البيانات وتقييم المدخلات ومعالجة الأخطاء. تظل هذه الممارسات الأساسية حاسمة لضمان

سلامة البيانات وأمنها. يجب النظر إلى البيانات ذاتية التصحيح كنهج تكميلي يمكن أن يعزز ويحسن هذه التدابير الحالية.

في النهاية، يعتمد قرار استخدام أنماط البيانات ذاتية التصحيح على المتطلبات والقيود والأولويات المحددة لتطبيقك. من خلال النظر بعناية في الاعتبارات المذكورة أعلاه ومواءمتها مع أهداف وبنية تطبيقك، يمكنك اتخاذ قرارات مستنيرة حول متى وكيف تستفيد من تقنيات البيانات ذاتية التصحيح بشكل فعال.

توليد المحتوى السياقي



تستفيد أنماط توليد المحتوى السياقي من قوة النماذج اللغوية الكبيرة (LLMs) لتوليد محتوى ديناميكي ومرتب بالسياق داخل التطبيقات. تدرك هذه الفئة من الأنماط أهمية تقديم محتوى مخصص وذو صلة للمستخدمين بناءً على احتياجاتهم وتفضيلاتهم المحددة، وحتى تفاعلاتهم السابقة والحالية مع التطبيق.

في سياق هذا النهج، يشير "المحتوى" إلى كل من المحتوى الأساسي (مثل المدونات والمقالات وما إلى ذلك) والمحتوى الفوقي، مثل التوصيات للمحتوى الأساسي.

يمكن لأنماط توليد المحتوى السياقي أن تلعب دوراً حاسماً في تعزيز مستويات مشاركة المستخدمين لديك، وتوفير تجارب مخصصة، وأتمتة مهام إنشاء المحتوى لك وللمستخدميك. من خلال استخدام الأنماط التي نصفها في هذا الفصل، يمكنك إنشاء تطبيقات تولد المحتوى بشكل ديناميكي، وتتكيف مع السياق والمدخلات في الوقت الفعلي.

تعمل الأنماط من خلال دمج النماذج اللغوية الكبيرة في مخرجات التطبيق، بدءاً من واجهة المستخدم (التي يشار إليها أحياناً باسم، "chrome" وصولاً إلى رسائل البريد الإلكتروني وأشكال الإشعارات الأخرى، وكذلك أي خطوط أنابيب لتوليد المحتوى.

عندما يتفاعل المستخدم مع التطبيق أو يطلب محتوى محدداً، يلتقط التطبيق السياق ذي الصلة، مثل تفضيلات المستخدم أو التفاعلات السابقة أو الإرشادات المحددة. ثم يتم تغذية هذه المعلومات السياقية إلى النموذج اللغوي الكبير، جنباً إلى جنب مع أي قوالب أو إرشادات ضرورية، واستخدامها لإنتاج مخرجات نصية كان من الممكن أن تكون إما مبرمجة بشكل ثابت، أو مخزنة في قاعدة بيانات، أو مولدة خوارزمية.

يمكن أن يأخذ المحتوى المولد من النماذج اللغوية الكبيرة أشكالاً متنوعة، مثل التوصيات الشخصية، وأوصاف المنتجات الديناميكية، وردود البريد الإلكتروني المخصصة، أو حتى المقالات أو المدونات الكاملة. أحد أكثر الاستخدامات جذرية لهذا المحتوى والتي قت براداتها قبل أكثر من عام هو توليد عناصر واجهة المستخدم بشكل ديناميكي مثل تسميات النماذج والتلميحات وأنواع أخرى من النصوص التوضيحية.

التخصيص

إحدى الفوائد الرئيسية لأنماط توليد المحتوى السياقي هي القدرة على تقديم تجارب مخصصة للغاية للمستخدمين. من خلال توليد محتوى يعتمد على السياق الخالص بكل مستخدم، تمكن هذه الأنماط التطبيقات من تخصيص المحتوى وفقاً لاهتمامات وتفضيلات وتفاعلات المستخدمين الفردية.

يتجاوز التخصيص مجرد إدراج اسم المستخدم في محتوى عام. إنه يتضمن الاستفادة من السياق الغني المتاح عن كل مستخدم لتوليد محتوى يتناغم مع احتياجاتهم ورغباتهم المحددة. يمكن أن يشمل هذا السياق مجموعة واسعة من العوامل، مثل:

1. معلومات الملف الشخصي للمستخدم: على المستوى الأكثر عمومية لتطبيق هذه التقنية، يمكن

استخدام البيانات الديموغرافية والاهتمامات والتفضيلات وسمات الملف الشخصي الأخرى لتوليد محتوى يتوافق مع خلفية المستخدم وخصائصه.

2. البيانات السلوكية: يمكن أن توفر التفاعلات السابقة للمستخدم مع التطبيق، مثل الصفحات التي تمت مشاهدتها، والروابط التي تم النقر عليها، أو المنتجات التي تم شراؤها، رؤى قيمة حول سلوكهم واهتماماتهم. يمكن استخدام هذه البيانات لتوليد اقتراحات محتوى تعكس أنماط مشاركتهم وتنبأ باحتياجاتهم المستقبلية.

3. العوامل السياقية: يمكن أن يؤثر السياق الحالي للمستخدم، مثل موقعه أو جهازه أو وقت اليوم أو حتى حالة الطقس، على عملية توليد المحتوى. على سبيل المثال، يمكن لتطبيق سفر أن يحتوي على عامل ذكاء اصطناعي قادر على توليد توصيات مخصصة بناءً على موقع المستخدم الحالي وظروف الطقس السائدة.

من خلال الاستفادة من هذه العوامل السياقية، تمكّن أنماط توليد المحتوى السياقي التطبيقات من تقديم محتوى يبدو وكأنه مصمم خصيصاً لكل مستخدم على حدة. يحقق هذا المستوى من التخصيص العديد من الفوائد المهمة:

1. زيادة المشاركة: يجذب المحتوى المخصص انتباه المستخدمين ويحافظ على مشاركتهم مع التطبيق. عندما يشعر المستخدمون أن المحتوى ذو صلة ويخاطب احتياجاتهم مباشرة، يزداد احتمال قضائهم وقتاً أطول في التفاعل مع التطبيق واستكشاف ميزات.
2. تحسين رضا المستخدم: يوضح المحتوى المخصص أن التطبيق يفهم ويهتم بالمتطلبات الفريدة للمستخدم. من خلال تقديم محتوى مفيد ومعلوماتي ومتوافق مع اهتماماتهم، يمكن للتطبيق تعزيز رضا المستخدم وبناء علاقة أقوى مع مستخدميه.
3. ارتفاع معدلات التحويل: في سياق تطبيقات التجارة الإلكترونية أو التسويق، يمكن للمحتوى المخصص أن يؤثر بشكل كبير على معدلات التحويل. من خلال تقديم منتجات أو عروض أو توصيات مصممة خصيصاً لتفضيلات وسلوك المستخدمين، يمكن للتطبيق زيادة احتمالية قيام المستخدمين باتخاذ الإجراءات المرغوبة، مثل إجراء عملية شراء أو الاشتراك في خدمة.

الإنتاجية

يمكن لأنماط توليد المحتوى السياقي أن تعزز بشكل كبير أنواعاً معينة من الإنتاجية من خلال تقليل الحاجة إلى التوليد والتحرير اليدوي للمحتوى في العمليات الإبداعية. من خلال الاستفادة من قوة النماذج اللغوية

الكبيرة، يمكنك توليد محتوى عالي الجودة على نطاق واسع، مما يوفر الوقت والجهد الذي كان سيضطر منشئ المحتوى والمطورون لديك لإنفاقه في العمل اليدوي المتكرر.

تقليدياً، يحتاج منشئ المحتوى إلى البحث والكتابة والتحرير وتنسيق المحتوى لضمان تلبية متطلبات التطبيق وتوقعات المستخدمين. يمكن أن تكون هذه العملية مستهلكة للوقت والموارد، خاصة مع زيادة حجم المحتوى.

ومع ذلك، مع أنماط توليد المحتوى السياقي، يمكن أتمتة عملية إنشاء المحتوى بشكل كبير. يمكن لنماذج اللغة الكبيرة توليد محتوى متماسك وسليم نحوياً ومرتبطة بالسياق بناءً على التوجيهات والإرشادات المقدمة. تقدم هذه الأتمتة العديد من فوائد الإنتاجية:

1. **تقليل الجهد اليدوي:** من خلال تفويض مهام توليد المحتوى إلى نماذج اللغة الكبيرة، يمكن لمنشئ المحتوى التركيز على المهام ذات المستوى الأعلى مثل استراتيجية المحتوى والأفكار وضمان الجودة. يمكنهم توفير السياق والقوالب والإرشادات اللازمة لنموذج اللغة الكبير وتركه يتعامل مع توليد المحتوى الفعلي. هذا يقلل من الجهد اليدوي المطلوب للكتابة والتحرير، مما يسمح لمنشئ المحتوى بأن يكونوا أكثر إنتاجية وكفاءة.
2. **إنشاء محتوى أسرع:** يمكن لنماذج اللغة الكبيرة توليد محتوى أسرع بكثير من الكُتاب البشريين. مع التوجيهات والإرشادات المناسبة، يمكن لنموذج اللغة الكبير إنتاج قطع متعددة من المحتوى في غضون ثوانٍ أو دقائق. تتيح هذه السرعة للتطبيقات توليد محتوى بوتيرة أسرع بكثير، مما يواكب متطلبات المستخدمين والمشهد الرقمي المتغير باستمرار.

هل يؤدي إنشاء المحتوى الأسرع إلى حالة "مأساة المشاعات" حيث يغرق الإنترنت في محتوى لا يقرأه أحد؟ للأسف، أعتقد أن الإجابة نعم.

3. **الاتساق والجودة:** يمكن لنماذج اللغة الكبيرة بسهولة مراجعة المحتوى بحث يكون متسقاً في الأسلوب والنبرة والجودة. مع وجود إرشادات وأمثلة واضحة، يمكن لأنواع معينة من التطبيقات (مثل غرف الأخبار والعلاقات العامة وما إلى ذلك) ضمان أن محتواها المنشأ بشرياً يتوافق مع صوت علامتها

التجارية ويلبي معايير الجودة المطلوبة. يقلل هذا الاتساق من الحاجة إلى التحرير والمراجعات المكثفة، مما يوفر الوقت والجهد في عملية إنشاء المحتوى.

4. التكرار والتحسين: تمكّن أنماط توليد المحتوى السياقي من التكرار السريع وتحسين المحتوى. من خلال تعديل التوجيهات أو القوالب أو الإرشادات المقدمة لنموذج اللغة الكبير، يمكن لتطبيقاتك بسرعة توليد تنوعات من المحتوى واختبار مناهج مختلفة بطريقة آلية لم تكن ممكنة في الماضي. تسمح هذه العملية التكرارية بتجريب وتحسين استراتيجيات المحتوى بشكل أسرع، مما يؤدي إلى محتوى أكثر فعالية وجاذبية مع مرور الوقت. يمكن أن تكون هذه التقنية تحديداً بمثابة تغيير جذري للتطبيقات مثل التجارة الإلكترونية التي تعتمد بشكل كامل على معدلات الارتداد والتفاعل

من المهم ملاحظة أنه على الرغم من أن أنماط توليد المحتوى السياقي يمكن أن تعزز الإنتاجية بشكل كبير، إلا أنها لا تلغي تماماً الحاجة إلى المشاركة البشرية. لا يزال منشئ المحتوى والمحررون يلعبون دوراً حاسماً في تحديد استراتيجية المحتوى الشاملة، وتوفير التوجيه لنموذج اللغة الكبير، وضمان جودة وملاءمة المحتوى المُولّد.



من خلال أتمتة الجوانب الأكثر تكراراً واستهلاكاً للوقت في إنشاء المحتوى، تحرر أنماط توليد المحتوى السياقي وقتاً وموارد بشرية قيمة يمكن إعادة توجيهها نحو المهام ذات القيمة الأعلى. تمكّنك هذه الإنتاجية المتزايدة من تقديم محتوى أكثر تخصيصاً وجاذبية للمستخدمين مع تحسين سير عمل إنشاء المحتوى.

التكرار السريع والتجريب

تمكّنك أنماط توليد المحتوى السياقي من التكرار السريع والتجريب مع تنوعات مختلفة من المحتوى، مما يسمح بتحسين وتنقيح استراتيجية المحتوى الخاصة بك بشكل أسرع. يمكنك توليد نسخ متعددة من المحتوى في غضون ثوانٍ، ببساطة عن طريق تعديل السياق أو القوالب أو الإرشادات المقدمة للنموذج.

تقدم قدرة التكرار السريع هذه العديد من الفوائد الرئيسية:

1. الاختبار والتحسين: مع القدرة على توليد تنوعات المحتوى بسرعة، يمكنك بسهولة اختبار مناهج مختلفة وقياس فعاليتها. على سبيل المثال، يمكنك توليد نسخ متعددة من وصف المنتج أو رسالة تسويقية، كل منها مصمم لشريحة مستخدمين معينة أو سياق معين. من خلال تحليل مقاييس

تفاعل المستخدم، مثل معدلات النقر أو معدلات التحويل، يمكنك تحديد تنوعات المحتوى الأكثر فعالية وتحسين استراتيجية المحتوى الخاصة بك وفقاً لذلك.

2. اختبار A/B: تمكّن أنماط توليد المحتوى السياقي من إجراء اختبار A/B سلس للمحتوى. يمكنك توليد نسختين أو أكثر من المحتوى وتقديمهما عشوائياً لمجموعات مستخدمين مختلفة. من خلال مقارنة أداء كل نسخة، يمكنك تحديد أي محتوى يتفاعل معه جمهورك المستهدف بشكل أفضل. يتيح لك هذا النهج المبني على البيانات اتخاذ قرارات مستنيرة وتحسين محتواك باستقرار لتعظيم تفاعل المستخدم وتحقيق النتائج المرجوة.

3. تجارب التخصيص: يعد التكرار السريع والتجريب ذا قيمة خاصة عندما يتعلق الأمر بالتخصيص. مع أنماط توليد المحتوى السياقي، يمكنك بسرعة توليد تنوعات محتوى مخصصة بناءً على شرائح المستخدمين المختلفة أو التفضيلات أو السلوكيات. من خلال التجريب مع استراتيجيات تخصيص مختلفة، يمكنك تحديد الأساليب الأكثر فعالية لإشراك المستخدمين الفرديين وتقديم تجارب مخصصة.

٤. التكيف مع الاتجاهات المتغيرة: تتيح لك القدرة على التكرار والتجريب السريع البقاء مرناً والتكيف مع الاتجاهات المتغيرة وتفضيلات المستخدمين. ومع ظهور مواضيع وكلمات مفتاحية أو سلوكيات جديدة للمستخدمين، يمكنك بسرعة توليد محتوى يتماشى مع هذه الاتجاهات. من خلال التجريب المستمر وتحسين المحتوى الخاص بك، يمكنك البقاء ملائماً والحفاظ على ميزة تنافسية في المشهد الرقمي المتطور باستمرار.

٥. تجريب فعال من حيث التكلفة: يتضمن التجريب التقليدي للمحتوى عادةً وقتاً وموارد كبيرة، حيث يحتاج منشئ المحتوى إلى تطوير واختبار تنوعات مختلفة يدوياً. ومع ذلك، مع أنماط توليد المحتوى السياقي، يتم تقليل تكلفة التجريب بشكل كبير. يمكن لنماذج اللغة الكبيرة توليد تنوعات المحتوى بسرعة وعلى نطاق واسع، مما يتيح لك استكشاف مجموعة واسعة من الأفكار والمناهج دون تكبد تكاليف كبيرة.

للاستفادة القصوى من التكرار السريع والتجريب، من المهم وجود إطار تجريب محدد بشكل جيد. يجب أن يتضمن هذا الإطار:

- أهداف وفرضيات واضحة لكل تجربة
- مقاييس وآليات تتبع مناسبة لقياس أداء المحتوى
- استراتيجيات التقسيم والاستهداف لضمان تقديم تنوعات المحتوى ذات الصلة للمستخدمين المناسبين
- أدوات التحليل وإعداد التقارير لاستخلاص الرؤى من البيانات التجريبية
- عملية لدمج التعلم والتحسينات في استراتيجية المحتوى الخاصة بك

من خلال تبني التكرار السريع والتجريب، يمكنك تحسين وتطوير محتواك باستقرار، مما يضمن أنه يظل جذاباً وملائماً وفعالاً في تحقيق أهداف تطبيقك. يتيح لك هذا النهج المرن في إنشاء المحتوى البقاء في المقدمة وتقديم تجارب استثنائية للمستخدمين.

قابلية التوسع والكفاءة

مع نمو التطبيقات وزيادة الطلب على المحتوى المخصص، تمكّن أنماط توليد المحتوى السياقي من توسيع نطاق إنشاء المحتوى بكفاءة. يمكن لنماذج اللغة الكبيرة توليد محتوى لعدد كبير من المستخدمين والسياقات في وقت واحد، دون الحاجة إلى زيادة متناسبة في الموارد البشرية. تتيح هذه القابلية للتوسع للتطبيقات تقديم تجارب مخصصة لقاعدة مستخدمين متنامية دون إرهاق قدراتها على إنشاء المحتوى.

لاحظ أن توليد المحتوى السياقي يمكن استخدامه بفعالية لتدويل تطبيقك "على الفور". في الواقع، هذا بالضبط ما فعلته باستخدام Gem Instant18n الخاص بي لتقديم Olympia بأكثر من نصف دزينة من اللغات، على الرغم من أننا لم نتجاوز عامنا الأول بعد.



التوطين المدعوم بالذكاء الاصطناعي

إذا سمحتم لي بالتفاخر للحظة، أعتقد أن مكتبة Instant18n الخاصة بي لتطبيقات Rails هي مثال رائد على نمط "توليد المحتوى السياقي" قيد التنفيذ، مما يوضح الإمكانيات التحويلية للذكاء الاصطناعي في تطوير التطبيقات. تستفيد هذه الحزمة من قوة نموذج GPT للغة الكبير من OpenAI لإحداث ثورة في طريقة التعامل مع التدويل والتوطين في تطبيقات Rails.

تقليدياً، يتضمن تدويل تطبيق Rails تحديد مفاتيح الترجمة يدوياً وتوفير الترجمات المقابلة لكل لغة مدعومة. يمكن أن تكون هذه العملية مستهلكة للوقت والموارد وعرضة لعدم الاتساق. ومع ذلك، مع حزمة Instant18n، يتم إعادة تعريف نموذج التوطين بالكامل.

من خلال دمج نموذج اللغة الكبير، تمكّنك حزمة Instant18n من توليد الترجمات فوراً، استناداً إلى سياق النص ومعناه. بدلاً من الاعتماد على مفاتيح الترجمة المحددة مسبقاً والترجمات الثابتة، تقوم الحزمة بترجمة النص ديناميكياً باستخدام قوة الذكاء الاصطناعي. يوفر هذا النهج العديد من المزايا الرئيسية:

١. توطين سلس: مع حزمة Instant18n، لم يعد المطورون بحاجة إلى تحديد وصيانة ملفات الترجمة يدوياً لكل لغة مدعومة. تقوم الحزمة تلقائياً بتوليد الترجمات بناءً على النص المقدم واللغة المستهدفة المطلوبة، مما يجعل عملية التوطين سهلة وسلسة.
٢. دقة سياقية: يمكن تزويد الذكاء الاصطناعي بسياق كاف لفهم دقائق النص المراد ترجمته. يمكنه مراعاة السياق المحيط والتعبيرات الاصطلاحية والمراجع الثقافية لتوليد ترجمات دقيقة وطبيعية ومناسبة سياقياً.
٣. دعم لغوي واسع: تستفيد حزمة Instant18n من المعرفة والقدرات اللغوية الواسعة لـ GPT، مما يتيح الترجمة إلى مجموعة واسعة من اللغات. من اللغات الشائعة مثل الإسبانية والفرنسية إلى اللغات الأكثر غموضاً أو الخيالية مثل الكلينجون والإلفية، يمكن للحزمة التعامل مع مجموعة متنوعة من متطلبات الترجمة.
٤. المرونة والإبداع: تتجاوز الحزمة الترجمات اللغوية التقليدية وتسمح بخيارات توطين إبداعية وغير تقليدية. يمكن للمطورين ترجمة النص إلى أنماط ولهجات مختلفة، أو حتى لغات خيالية، مما يفتح إمكانيات جديدة لتجارب مستخدم فريدة ومحتوى جذاب.
٥. تحسين الأداء: تتضمن حزمة Instant18n آليات للتخزين المؤقت لتحسين الأداء وتقليل العبء الناتج عن الترجمات المتكررة. يتم تخزين النص المترجم مؤقتاً، مما يسمح بتقديم الطلبات اللاحقة لنفس الترجمة بسرعة دون الحاجة إلى استدعاءات API زائدة.
- تجسد حزمة Instant18n قوة نمط "توليد المحتوى السياقي" من خلال الاستفادة من الذكاء الاصطناعي لتوليد محتوى موطن بشكل ديناميكي. إنها توضح كيف يمكن دمج الذكاء الاصطناعي في الوظائف الأساسية لتطبيق Rails، مما يغير الطريقة التي يتعامل بها المطورون مع التدويل والتوطين.
- من خلال القضاء على الحاجة إلى إدارة الترجمة اليدوية وتمكين الترجمات الفورية بناءً على السياق، توفر حزمة Instant18n وقتاً وجهداً كبيراً للمطورين. فهي تتيح لهم التركيز على بناء الميزات الأساسية لتطبيقهم مع ضمان معالجة جانب التوطين بسلاسة ودقة.

أهمية اختبار المستخدم والتغذية الراجعة

وأخيراً، ضع دائماً في اعتبارك أهمية اختبار المستخدم والتغذية الراجعة. من الضروري التحقق من أن إنشاء المحتوى السياقي يلبي توقعات المستخدمين ويتوافق مع أهداف التطبيق. واصل التكرار وتحسين المحتوى

المنشأ بناءً على رؤى المستخدمين والتحليلات. إذا كنت تقوم بإنشاء محتوى ديناميكي على نطاق واسع يستحيل التحقق منه يدوياً من قبلك وفريقك، ففكر في إضافة آليات للتغذية الراجعة تسمح للمستخدمين بالإبلاغ عن المحتوى الغريب أو الخاطئ، مع شرح السبب. يمكن حتى تغذية هذه الملاحظات القيمة إلى عامل ذكي اصطناعي مكلف بإجراء تعديلات على المكون الذي أنشأ المحتوى!

واجهة المستخدم التوليدية



نظراً لأن الانتباه أصبح نادراً هذه الأيام، فإن المشاركة الفعالة للمستخدم تتطلب الآن تجارب برمجية ليست سلسلة وبديهية فحسب، بل مخصصة أيضاً بدرجة عالية للاحتياجات والتفضيلات والسياقات الفردية. ونتيجة لذلك، يواجه المصممون والمطورون بشكل متزايد تحدي إنشاء واجهات مستخدم يمكنها التكيف وتلبية المتطلبات الفريدة لكل مستخدم على نطاق واسع.

تُعد واجهة المستخدم التوليدية (GenUI) نهجاً ثورياً حقاً في تصميم واجهة المستخدم يستفيد من قوة النماذج اللغوية الكبيرة (LLMs) لإنشاء تجارب مستخدم مخصصة للغاية وديناميكية بشكل فوري. أردت التأكد من تقديم مقدمة أساسية على الأقل عن GenUI في هذا الكتاب، لأنني أعتقد أنها واحدة من أكثر الفرص الواعدة التي توجد حالياً في مجال تصميم التطبيقات وأطر العمل. أنا مقتنع بأن العشرات أو أكثر من

المشاريع التجارية ومفتوحة المصدر الناجحة ستظهر في هذا المجال المتخصص.

في جوهرها، تجمع GenUI بين مبادئ توليد المحتوى السياقي وتقنيات الذكاء الاصطناعي المتقدمة لتوليد عناصر واجهة المستخدم، مثل النصوص والصور والتخطيطات، بشكل ديناميكي بناءً على فهم عميق لسياق المستخدم وتفضيلاته وأهدافه. تُمكن GenUI المصممين والمطورين من إنشاء واجهات تتكيف وتتطور استجابةً لتفاعلات المستخدم، مما يوفر مستوى من التخصيص لم يكن ممكناً تحقيقه من قبل.

تمثل GenUI تغييراً أساسياً في طريقة تناولنا لتصميم واجهة المستخدم. بدلاً من التصميم للجمهور، يتيح لنا GenUI التصميم للفرد. يمتلك المحتوى والواجهات المخصصة القدرة على إنشاء تجارب مستخدم تتناغم مع كل مستخدم على مستوى أعمق، مما يزيد من المشاركة والرضا والولاء.

ك تقنية حديثة جداً، فإن الانتقال إلى GenUI مليء بالتحديات المفاهيمية والعملية. دمج الذكاء الاصطناعي في عملية التصميم، وضمان أن الواجهات المولدة ليست مخصصة فحسب بل أيضاً قابلة للاستخدام ويمكن الوصول إليها ومتوافقة مع العلامة التجارية وتجربة المستخدم الشاملة، كل هذه تحديات تجعل GenUI مسعى للقلّة وليس للكثيرين. بالإضافة إلى ذلك، يثير إشراك الذكاء الاصطناعي أسئلة حول خصوصية البيانات والشفافية وربما حتى الآثار الأخلاقية

على الرغم من التحديات، فإن التجارب المخصصة على نطاق واسع لديها القدرة على تحويل الطريقة التي تتفاعل بها مع المنتجات والخدمات الرقمية بشكل كامل. إنها تفتح إمكانيات لإنشاء واجهات شاملة ويمكن الوصول إليها تلبي الاحتياجات المتنوعة للمستخدمين، بغض النظر عن قدراتهم أو خلفياتهم أو تفضيلاتهم. في هذا الفصل، سنستكشف مفهوم GenUI مع دراسة بعض الخصائص المميزة والفوائد الرئيسية والتحديات المحتملة. نبدأ بالنظر في أبسط وأسهل شكل من أشكال GenUI: توليد نصوص النسخ لواجهات المستخدم المصممة والمنفذة بالطريقة التقليدية.

توليد النصوص لواجهات المستخدم

عناصر النص الموجودة في واجهة تطبيقك، مثل تسميات النماذج والتلميحات والنصوص التوضيحية، عادةً ما تكون مدمجة في القوالب أو مكونات واجهة المستخدم، مما يوفر تجربة متسقة ولكنها عامة لجميع المستخدمين. باستخدام أنماط توليد المحتوى السياقي، يمكنك تحويل هذه العناصر الثابتة إلى مكونات ديناميكية وواعية بالسياق ومخصصة.

النماذج المخصصة

النماذج جزء لا يتجزأ من تطبيقات الويب والهاتف المحمول، حيث تعمل كوسيلة أساسية لجمع مدخلات المستخدم. ومع ذلك، غالباً ما تقدم النماذج التقليدية تجربة عامة وغير شخصية، مع تسميات وحقول قياسية قد لا تتوافق دائماً مع السياق أو احتياجات المستخدم المحددة. من المرجح أن يقوم المستخدمون بإكمال النماذج التي تبدو مصممة خصيصاً لاحتياجاتهم وتفضيلاتهم، مما يؤدي إلى معدلات تحويل ورضا مستخدم أعلى.

ومع ذلك، من المهم تحقيق التوازن بين التخصيص والاتساق. في حين أن تكييف النماذج للمستخدمين الفرديين يمكن أن يكون مفيداً، فمن الضروري الحفاظ على مستوى من الألفة والقدرة على التنبؤ. يجب أن يظل المستخدمون قادرين على التعرف على النماذج والتنقل فيها بسهولة، حتى مع وجود العناصر المخصصة. إليك بعض أفكار النماذج المخصصة للإلهام:

اقتراحات الحقول السياقية

يمكن لـ GenUI تحليل التفاعلات السابقة للمستخدم وتفضيلاته وبياناته لتقديم اقتراحات ذكية للحقول كتنبؤات. على سبيل المثال، إذا قام المستخدم سابقاً بإدخال عنوان الشحن الخاص به، يمكن للنموذج ملء الحقول ذات الصلة تلقائياً بمعلوماته المحفوظة. هذا لا يوفر الوقت فحسب، بل يظهر أيضاً أن التطبيق يفهم ويتذكر تفضيلات المستخدم.

انتظر لحظة، أليست هذه التقنية شيئاً يمكن تنفيذه دون استخدام الذكاء الاصطناعي؟ بالطبع، لكن جمال تشغيل هذا النوع من الوظائف باستخدام الذكاء الاصطناعي يكمن في نقطتين: (١) مدى سهولة التنفيذ (٢) مدى مرونة النظام مع تغير وتطور واجهة المستخدم بمرور الوقت.

دعنا نقوم بإنشاء خدمة لنظام معالجة الطلبات النظري الخاص بنا، والتي تحاول ملء عنوان الشحن الصحيح للمستخدم بشكل استباقي.

```

1  class OrderShippingAddressSubscriber
2    include Raix::ChatCompletion
3
4    attr_accessor :order
5
6    delegate :customer, to: :order
7
8    DIRECTIVE = "the Given assistant. processing order smart a are You
9    address shipping likely most the guess ,history order customer's
10   order. current the for "
11
12   def order_created(order)
13     return unless order.pending? && order.shipping_address.blank?
14
15     self.order = order
16
17     transcript.clear
18     transcript << { system: DIRECTIVE }
19     transcript << { user: " History: Order#{order.history.to_json}" }
20     transcript << { user: " Order: Current#{order.to_json}" }
21
22     response = chat_completion
23     apply_predicted_shipping_address(order, response)
24   end
25
26   private
27
28   def apply_predicted_shipping_address(order, response)
29     response... the from address shipping the extract #
30     fields address the of update live of sort some there's assume ...and #
31     order.update(shipping_address:)
32   end
33
34   def order_history
35     customer.orders.successful.limit(100).map do |order|
36       {
37         date: order.date,
38         description: order.description,
39         shipping_address: order.shipping_address
40       }
41     end
42   end

```

43 **end**

هذا المثال مبسط للغاية، ولكنه يجب أن يعمل في معظم الحالات. الفكرة هي السماح للذكاء الاصطناعي بالتخمين بنفس الطريقة التي يفعلها الإنسان. ولتوضيح ما أتحدث عنه، دعنا ننظر إلى بعض البيانات التجريبية:

```
1 Order History:
2 [
3   {"date": "2024-01-03", "description": "garden soil mix",
4     "shipping_address": "123 Country Lane, Rural Town"},
5   {"date": "2024-01-15", "description": "hardcover fiction novels",
6     "shipping_address": "456 City Apt, Metroville"},
7   {"date": "2024-01-22", "description": "baby diapers", "shipping_address":
8     "789 Suburb St, Quietville"},
9   {"date": "2024-02-01", "description": "organic vegetables",
10    "shipping_address": "123 Country Lane, Rural Town"},
11   {"date": "2024-02-17", "description": "mystery thriller book set",
12    "shipping_address": "456 City Apt, Metroville"},
13   {"date": "2024-02-25", "description": "baby wipes",
14    "shipping_address": "789 Suburb St, Quietville"},
15   {"date": "2024-03-05", "description": "flower seeds",
16    "shipping_address": "123 Country Lane, Rural Town"},
17   {"date": "2024-03-20", "description": "biographies",
18    "shipping_address": "456 City Apt, Metroville"},
19   {"date": "2024-03-30", "description": "baby formula",
20    "shipping_address": "789 Suburb St, Quietville"},
21   {"date": "2024-04-12", "description": "lawn fertilizer",
22    "shipping_address": "123 Country Lane, Rural Town"},
23   {"date": "2024-04-22", "description": "science fiction novels",
24    "shipping_address": "456 City Apt, Metroville"},
25   {"date": "2024-05-02", "description": "infant toys",
26    "shipping_address": "789 Suburb St, Quietville"},
27   {"date": "2024-05-14", "description": "outdoor grill",
28    "shipping_address": "123 Country Lane, Rural Town"},
29   {"date": "2024-05-29", "description": "literary classics",
30    "shipping_address": "456 City Apt, Metroville"},
31   {"date": "2024-06-11", "description": "baby clothes",
32    "shipping_address": "789 Suburb St, Quietville"},
33   {"date": "2024-07-01", "description": "watering can",
34    "shipping_address": "123 Country Lane, Rural Town"},
```

```
35 {"date": "2024-07-18", "description": "non-fiction essays",
36 "shipping_address": "456 City Apt, Metroville"},
37 {"date": "2024-07-28", "description": "baby bath items",
38 "shipping_address": "789 Suburb St, Quietville"},
39 {"date": "2024-08-09", "description": "herb garden kit",
40 "shipping_address": "123 Country Lane, Rural Town"},
41 {"date": "2024-08-24", "description": "children's books",
42 "shipping_address": "456 City Apt, Metroville"}
43 ]
```

هل لاحظت النمط في البيانات؟ أؤكد لك أن هذا أمر سهل للغاية بالنسبة للنموذج اللغوي الكبير. للتوضيح، دعنا نسأل 4GPT- عن عنوان الشحن الأكثر احتمالاً لـ "ميزان الحرارة".

```
1 From the order history you've provided, it looks like the purchases are
2 generally clustered into three main types based on the shipping addresses:
3
4 123 Country Lane, Rural Town - This address often orders garden and
5 outdoor-related items like soil mix, vegetables, flower seeds, lawn
6 fertilizer, an outdoor grill, a watering can, and an herb garden kit.
7
8 456 City Apt, Metroville - This address tends to order various types of
9 books, indicating an interest in reading across genres like fiction,
10 thrillers, biographies, science fiction, literary classics, non-fiction
11 essays, and children's books.
12
13 789 Suburb St, Quietville - The purchases for this address are consistently
14 baby-related products, such as diapers, wipes, baby formula, infant toys,
15 baby clothes, and baby bath items.
16
17 Given these patterns, a digital thermometer could potentially be a fit for
18 any of the addresses based on different needs: outdoor temperature tracking,
19 a tool for a passionate reader interested in science gadgets, or a health
20 tool for a baby. However, considering its common use in monitoring health,
21 especially for babies, it's most likely to be purchased by someone who's
22 already buying baby care products.
23
24 Therefore, the most likely shipping address for a digital thermometer, based
25 on the purchasing patterns, would be 789 Suburb St, Quietville.
```

إذا كنت تعتقد أن استخدام نموذج مكلف مثل 4GPT- لهذه المهمة هو إفراط، فأنت على حق! لقد

جربت نفس الاستفسار على Instruct 7B Mistral وأنتج الاستجابة التالية بمعدل 75 رمزاً في الثانية، وبتكلفة زهيدة قدرها 000218.0 دولار أمريكي.

Based on the given order history, it appears that the customer has received orders for baby-related items (diapers, wipes, formula, clothes, bath items) at the address "789 Suburb St, Quietville" and garden-related items (soil, seeds, watering can, herb garden kit) at the address "123 Country Lane, Rural Town". The customer has also received books (fiction, non-fiction, literary classics, mystery-thriller, science fiction, biographies, children's books) at the address "456 City Apt, Metroville".

Given that the new order is for a "thermometer", which doesn't seem to fit into any of the categories that the customer has previously ordered, it's hard to accurately predict the shipping address based solely on the order history. However, given the 50%-50% split between baby-related and garden-related items, it could somewhat lean towards the Baby-related items address ("789 Suburb St, Quietville"). But remember, this is an assumption and cannot be definitively confirmed without more context or information.

هل تستحق التكلفة والجهد المبذول في هذه التقنية لجعل تجربة الدفع أكثر سحراً؟ بالنسبة للعديد من تجار التجزئة عبر الإنترنت، بالتأكيد نعم. ويبدو أن تكلفة الحوسبة باستخدام الذكاء الاصطناعي ستستمر في الانخفاض، خاصة مع مقدمي خدمات استضافة النماذج مفتوحة المصدر في سباق نحو الأسعار الأقل.

استخدم قالب التوجيه وStructuredIO مع تسييج الاستجابة لتحسين هذا النوع من إكمال المحادثة.



ترتيب الحقول التكميلي

يمكن أن يؤثر ترتيب حقول النموذج بشكل كبير على تجربة المستخدم ومعدلات الإكمال. مع GenUI، يمكنك تعديل ترتيب الحقول ديناميكياً بناءً على سياق المستخدم وأهمية كل حقل. على سبيل المثال، إذا كان المستخدم يملأ نموذج تسجيل لتطبيق لياقة بدنية، يمكن للنموذج إعطاء الأولوية للحقول المتعلقة بأهدافه وتفضيلاته في اللياقة البدنية، مما يجعل العملية أكثر صلة وجاذبية.

النصوص المصغرة المخصصة

يمكن أيضاً تخصيص النص التعليمي ورسائل الخطأ والنصوص المصغرة الأخرى المرتبطة بالنماذج باستخدام GenUI. بدلاً من عرض رسائل خطأ عامة مثل "عنوان البريد الإلكتروني غير صالح"، يمكنك إنشاء رسائل أكثر فائدة وسياقية مثل "يرجى إدخال عنوان بريد إلكتروني صالح لتلقي تأكيد طلبك". هذه اللمسات الشخصية يمكن أن تجعل تجربة النموذج أكثر سهولة وأقل إحباطاً.

التحقق المخصص

على نفس منوال النصوص المصغرة المخصصة، يمكنك استخدام الذكاء الاصطناعي للتحقق من صحة النموذج بطرق تبدو سحرية. تخيل السماح للذكاء الاصطناعي بالتحقق من نموذج ملف المستخدم، بحثاً عن الأخطاء المحتملة على المستوى الدلالي.

Create your account

Full name

Obie Fernandez

Email

obiefernandez@gmail.com



Did you mean obiefernandez@gmail.com? [Yes, update.](#)

Country ⓘ

 United States



Password

.....



✓ Nice work. This is an excellent password.

الشكل 9.12. هل يمكنك ملاحظة التحقق الدلالي الجاري؟

الكشف التدريجي

يمكن لـ GenUI تحديد الحقول الأساسية بذكاء بناءً على سياق المستخدم والكشف تدريجياً عن الحقول الإضافية حسب الحاجة. تساعد تقنية الكشف التدريجي هذه في تقليل العبء المعرفي وتجعل عملية ملء النموذج أكثر قابلية للإدارة. على سبيل المثال، إذا كان المستخدم يسجل في اشتراك أساسي، يمكن للنموذج

أن يعرض في البداية الحقول الأساسية فقط، ومع تقدم المستخدم أو اختياره لخيارات معينة، يمكن إدخال حقول إضافية ذات صلة بشكل ديناميكي.

النص التوضيحي المدرك للسياق

غالباً ما تُستخدم تلميحات الأدوات لتقديم معلومات إضافية أو إرشادات للمستخدمين عند تمرير المؤشر فوق عناصر معينة أو التفاعل معها. من خلال نهج "توليد المحتوى السياقي"، يمكنك إنشاء تلميحات أدوات تتكيف مع سياق المستخدم وتقدم معلومات ذات صلة. على سبيل المثال، إذا كان المستخدم يستكشف ميزة معقدة، يمكن للتلميح الأداة تقديم نصائح أو أمثلة مخصصة بناءً على تفاعلاته السابقة أو مستوى مهاراته.

يمكن توليد النص التوضيحي، مثل التعليمات أو الأوصاف أو رسائل المساعدة، ديناميكياً بناءً على سياق المستخدم. بدلاً من تقديم تفسيرات عامة، يمكنك استخدام نماذج اللغة الكبيرة لتوليد نص مخصص لاحتياجات أو أسئلة المستخدم المحددة. على سبيل المثال، إذا كان المستخدم يواجه صعوبة في خطوة معينة من العملية، يمكن للنص التوضيحي تقديم إرشادات مخصصة أو نصائح لحل المشكلات.

تشير النصوص المصغرة إلى قطع النص الصغيرة التي ترشد المستخدمين خلال تطبيقك، مثل تسميات الأزرار أو رسائل الخطأ أو مطالبات التأكيد. من خلال تطبيق نهج **توليد المحتوى السياقي** على النصوص المصغرة، يمكنك إنشاء واجهة مستخدم تكيفية تستجيب لإجراءات المستخدم وتوفر نصاً مفيداً وذات صلة. على سبيل المثال، إذا كان المستخدم على وشك تنفيذ إجراء مهم، يمكن إنشاء مطالبة التأكيد ديناميكياً لتقديم رسالة واضحة ومخصصة.

يمكن للنص التوضيحي المخصص وتلميحات الأدوات أن تعزز بشكل كبير عملية التأهيل للمستخدمين الجدد. من خلال تقديم إرشادات وأمثلة خاصة بالسياق، يمكنك مساعدة المستخدمين على فهم التطبيق والتنقل فيه بسرعة، مما يقلل من منحنى التعلم ويزيد من معدل التبني.

يمكن أيضاً لعناصر الواجهة الديناميكية والمدركة للسياق أن تجعل التطبيق يبدو أكثر بديهية وجاذبية. من المرجح أن يتفاعل المستخدمون مع الميزات ويستكشفوها عندما يكون النص المصاحب مخصصاً لاحتياجاتهم واهتماماتهم المحددة.

حتى الآن تناولنا أفكاراً لتحسين نماذج واجهة المستخدم الحالية باستخدام الذكاء الاصطناعي، ولكن ماذا عن إعادة التفكير في كيفية تصميم وتنفيذ واجهات المستخدم بطريقة أكثر جذرية؟

تعريف واجهة المستخدم التوليدية

على عكس تصميم واجهة المستخدم التقليدي، حيث يقوم المصممون بإنشاء واجهات ثابتة وستاتيكية، تشير واجهة المستخدم التوليدية إلى مستقبل تتميز فيه برمجياتنا بتجارب مرنة ومخصصة يمكنها التطور والتكيف في الوقت الفعلي. في كل مرة نستخدم فيها واجهة محدثة مدعومة بالذكاء الاصطناعي، نسمح للذكاء الاصطناعي بالتكيف مع احتياجات المستخدم الخاصة. تأخذ واجهة المستخدم التوليدية الأمور خطوة إلى الأمام من خلال تطبيق هذا المستوى من القابلية للتكيف على الواجهة المرئية للبرمجيات.

السبب في إمكانية تجربة أفكار واجهة المستخدم التوليدية اليوم هو أن نماذج اللغة الكبيرة تفهم بالفعل البرمجة وتتضمن معرفتها الأساسية تقنيات وأطر عمل واجهة المستخدم. السؤال هو ما إذا كان يمكن استخدام نماذج اللغة الكبيرة لتوليد عناصر واجهة المستخدم، مثل النصوص والصور والتخطيطات، وحتى الواجهات بأكملها، المصممة خصيصاً لكل مستخدم على حدة. يمكن توجيه النموذج لمراعاة عوامل مختلفة، مثل تفاعلات المستخدم السابقة، والتفضيلات المعلنة، والمعلومات الديموغرافية، والسياق الحالي للاستخدام، لإنشاء واجهات مخصصة وذات صلة للغاية.

تختلف واجهة المستخدم التوليدية عن تصميم واجهة المستخدم التقليدية في عدة جوانب رئيسية:

1. ديناميكية وقابلة للتكيف: يتضمن تصميم واجهة المستخدم التقليدي إنشاء واجهات ثابتة وستاتيكية تظل كما هي لجميع المستخدمين. في المقابل، تتيح واجهة المستخدم التوليدية واجهات يمكنها التكيف والتغير ديناميكياً بناءً على احتياجات المستخدم والسياق. هذا يعني أن التطبيق نفسه يمكن أن يقدم واجهات مختلفة لمستخدمين مختلفين أو حتى للمستخدم نفسه في مواقف مختلفة.
2. التخصيص على نطاق واسع: مع التصميم التقليدي، غالباً ما يكون إنشاء تجارب مخصصة لكل مستخدم غير عملي بسبب الوقت والموارد المطلوبة. من ناحية أخرى، تسمح واجهة المستخدم التوليدية بالتخصيص على نطاق واسع. من خلال الاستفادة من الذكاء الاصطناعي، يمكن للمصممين إنشاء واجهات تتكيف تلقائياً مع احتياجات وتفضيلات كل مستخدم، دون الحاجة إلى تصميم وتطوير واجهات منفصلة لكل شريحة من المستخدمين.
3. التركيز على النتائج: غالباً ما يركز تصميم واجهة المستخدم التقليدي على إنشاء واجهات جذابة بصرياً وعملية. في حين أن هذه الجوانب لا تزال مهمة في واجهة المستخدم التوليدية، إلا أن التركيز الأساسي يتحول نحو تحقيق النتائج المرجوة للمستخدم. تهدف واجهة المستخدم التوليدية إلى

إنشاء واجهات محسنة لأهداف ومهام كل مستخدم على وجه التحديد، مع إعطاء الأولوية لسهولة الاستخدام والفعالية على الاعتبارات الجمالية البحتة.

4. التعلم والتحسين المستمر: يمكن لأنظمة واجهة المستخدم التوليدية التعلم والتحسين باستمرار مع مرور الوقت بناءً على تفاعلات المستخدم وتعليقاته. عندما يتفاعل المستخدمون مع الواجهات المولدة، يمكن لنماذج الذكاء الاصطناعي جمع البيانات حول سلوك المستخدم وتفضيلاته ونتائجه، واستخدام هذه المعلومات لتحسين وتطوير عمليات توليد الواجهة المستقبلية. تتيح عملية التعلم التكرارية هذه لأنظمة واجهة المستخدم التوليدية أن تصبح أكثر فعالية في تلبية احتياجات المستخدم مع مرور الوقت.

من المهم ملاحظة أن واجهة المستخدم التوليدية ليست نفس أدوات التصميم المدعومة بالذكاء الاصطناعي، مثل تلك التي تقدم اقتراحات أو تؤتمت مهام تصميم معينة. في حين أن هذه الأدوات يمكن أن تكون مفيدة في تبسيط عملية التصميم، إلا أنها لا تزال تعتمد على المصممين لاتخاذ القرارات النهائية وإنشاء واجهات ثابتة. من ناحية أخرى، تتضمن واجهة المستخدم التوليدية قيام نظام الذكاء الاصطناعي بدور أكثر نشاطاً في التوليد الفعلي للواجهات وتكييفها بناءً على بيانات المستخدم والسياق.

تمثل واجهة المستخدم التوليدية تحولاً كبيراً في كيفية تعاملنا مع تصميم واجهة المستخدم، مبتعدين عن الحلول الموحدة للجميع ومتجهين نحو تجارب مخصصة وقابلة للتكيف للغاية. من خلال الاستفادة من قوة الذكاء الاصطناعي، لدى واجهة المستخدم التوليدية القدرة على إحداث ثورة في طريقة تفاعلنا مع المنتجات والخدمات الرقمية، مما يخلق واجهات أكثر بديهية وجاذبية وفعالية لكل مستخدم على حدة.

مثال

لتوضيح مفهوم واجهة المستخدم التوليدية، دعونا نأخذ في الاعتبار تطبيقاً افتراضياً للباقة البدنية يسمى "FitAI". يهدف هذا التطبيق إلى تقديم خطط تمارين وتغذية مخصصة للمستخدمين بناءً على أهدافهم ومستويات لياقتهم وتفضيلاتهم الفردية.

في نهج تصميم واجهة المستخدم التقليدي، قد يحتوي FitAI على مجموعة ثابتة من الشاشات والعناصر التي تكون متماثلة لجميع المستخدمين. ومع ذلك، مع واجهة المستخدم التوليدية، يمكن لواجهة التطبيق أن تتكيف ديناميكياً مع احتياجات وسياق كل مستخدم على حدة.

هذا النهج يعد نوعاً من التصور البعيد للتنفيذ في عام 2024 وقد لا يكون له عائد على الاستثمار كافٍ، ولكنه ممكن.

إليك كيف يمكن أن يعمل:

1. التأهيل المبدئي:

- بدلاً من الاستبيان القياسي، يستخدم FitAI ذكاءً اصطناعياً محاداً لجمع معلومات حول أهداف المستخدم ومستوى لياقته الحالي وتفضيلاته.
- بناءً على هذا التفاعل الأولي، يقوم الذكاء الاصطناعي بتوليد تخطيط لوحة معلومات مخصصة، مع إبراز الميزات والمعلومات الأكثر صلة بأهداف المستخدم.
- قد تمتلك تقنية الذكاء الاصطناعي الحالية مجموعة من مكونات الشاشة تحت تصرفها لاستخدامها في تكوين لوحة المعلومات المخصصة.
- قد تتولى تقنية الذكاء الاصطناعي المستقبلية دور مصمم واجهة المستخدم المتمرس وتقوم بالفعل بإنشاء لوحة المعلومات من الصفر.

2. مخطط التمارين:

- تتكيف واجهة مخطط التمارين بواسطة الذكاء الاصطناعي لتناسب خصيصاً مع مستوى خبرة المستخدم والمعدات المتاحة له.
- بالنسبة للمبتدئ الذي لا يملك معدات، قد تعرض تمارين بسيطة باستخدام وزن الجسم مع تعليمات مفصلة ومقاطع فيديو.
- بالنسبة للمستخدم المتقدم الذي لديه إمكانية الوصول إلى صالة رياضية، يمكن أن تعرض روتينات أكثر تعقيداً مع محتوى توضيحي أقل.
- محتوى مخطط التمارين لا يتم تصفيته ببساطة من مجموعة أكبر. يمكن توليده مباشرة استناداً إلى قاعدة معرفية يتم الاستعلام عنها مع سياق يتضمن كل ما هو معروف عن المستخدم.

3. تتبع التقدم:

- تتطور واجهة تتبع التقدم بناءً على أهداف المستخدم وأنماط مشاركته.
- إذا كان المستخدم يركز بشكل أساسي على فقدان الوزن، قد تعرض الواجهة بشكل بارز رسماً بيانياً لاتجاه الوزن وإحصائيات حرق السعرات الحرارية.

- بالنسبة للمستخدم الذي يبني العضلات، يمكن أن تسلط الضوء على مكاسب القوة وتغيرات تكوين الجسم.
- يمكن للذكاء الاصطناعي تكيف هذا الجزء من التطبيق مع التقدم الفعلي للمستخدم. إذا توقف التقدم لفترة من الوقت، يمكن للتطبيق الانتقال إلى وضع يحاول فيه استدراج المستخدم للكشف عن أسباب التراجع، من أجل معالجتها.

٤. النصائح الغذائية:

- يتكيف قسم التغذية مع التفضيلات والقيود الغذائية للمستخدم.
- بالنسبة للمستخدم النباتي، قد يعرض اقتراحات وجبات نباتية ومصادر البروتين.
- بالنسبة للمستخدم الذي يعاني من حساسية الغلوتين، سيقوم تلقائياً بتصفية الأطعمة المحتوية على الغلوتين من التوصيات.
- مرة أخرى، لا يتم استخلاص المحتوى من مجموعة ضخمة من بيانات الوجبات التي تنطبق على جميع المستخدمين، بل يتم توليفه من قاعدة معرفية تحتوي على معلومات قابلة للتكيف بناءً على وضع المستخدم المحدد وقيوده.
- على سبيل المثال، يتم توليد الوصفات مع مواصفات المكونات التي تتناسب مع احتياجات السرعات الحرارية المتغيرة باستمرار للمستخدم مع تطور مستوى لياقته وإحصاءات جسمه.

٥. العناصر التحفيزية:

- يتم تخصيص محتوى التطبيق التحفيزي والإشعارات بناءً على نوع شخصية المستخدم واستجابته لاستراتيجيات التحفيز المختلفة.
- قد يتلقى بعض المستخدمين رسائل تشجيعية، بينما يحصل آخرون على تعليقات تستند إلى البيانات بشكل أكبر.

في هذا المثال، تمكّن واجهة المستخدم التوليدية تطبيق FitAI من إنشاء تجربة مخصصة للغاية لكل مستخدم، مما يزيد من احتمالية المشاركة والرضا وتحقيق أهداف اللياقة البدنية. تتكيف عناصر الواجهة والمحتوى وحتى "شخصية" التطبيق لخدمة احتياجات وتفضيلات كل مستخدم على أفضل وجه.

التحول نحو التصميم الموجه نحو النتائج

تمثل واجهة المستخدم التوليدية تحولاً أساسياً في نهج تصميم واجهة المستخدم، من التركيز على إنشاء عناصر واجهة محددة إلى نهج أكثر شمولية موجه نحو النتائج. لهذا التحول عدة آثار مهمة:

١. التركيز على أهداف المستخدم:

- سيحتاج المصممون إلى التفكير بعمق أكبر في أهداف المستخدم والنتائج المرجوة بدلاً من مكونات الواجهة المحددة.
- سيكون التركيز على إنشاء أنظمة يمكنها توليد واجهات تساعد المستخدمين على تحقيق أهدافهم بكفاءة وفعالية.
- ستظهر أطر عمل جديدة لواجهة المستخدم تمنح المصممين القائمين على الذكاء الاصطناعي الأدوات التي يحتاجون إليها لتوليد تجارب المستخدم مباشرةً ومن الصفر - بدلاً من الاعتماد على مواصفات شاشة محددة مسبقاً.

٢. تغيير دور المصممين:

- سينتقل المصممون من إنشاء تخطيطات ثابتة إلى تحديد القواعد والقيود والإرشادات التي يجب أن تتبعها أنظمة الذكاء الاصطناعي عند توليد الواجهات.
- سيحتاجون إلى تطوير مهارات في مجالات مثل تحليل البيانات وهندسة موجّهات الذكاء الاصطناعي والتفكير المنظومي للتوجيه الفعال لأنظمة واجهة المستخدم التوليدية.

٣. أهمية أبحاث المستخدم:

- تصبح أبحاث المستخدم أكثر أهمية في سياق واجهة المستخدم التوليدية، حيث يحتاج المصممون إلى فهم ليس فقط تفضيلات المستخدم، ولكن أيضاً كيف تتغير هذه التفضيلات والاحتياجات في سياقات مختلفة.
- ستكون حلقات اختبار المستخدم المستمرة والتغذية الراجعة ضرورية لتحسين وتطوير قدرة الذكاء الاصطناعي على توليد واجهات فعالة.

٠٤. التصميم للتغير:

- بدلاً من إنشاء واجهة "مثالية" واحدة، سيحتاج المصممون إلى النظر في تنوعات متعددة محتملة وضمان قدرة النظام على توليد واجهات مناسبة لاحتياجات المستخدمين المتنوعة.
- يشمل ذلك التصميم للحالات الاستثنائية وضمان الحفاظ على قابلية الاستخدام وإمكانية الوصول في التكوينات المختلفة.
- يأخذ تمييز المنتج أبعاداً جديدة تتضمن وجهات نظر متباينة حول علم نفس المستخدم والاستفادة من مجموعات البيانات وقواعد المعرفة الفريدة غير المتاحة للمنافسين.

التحديات والاعتبارات

في حين أن واجهة المستخدم التوليدية تقدم إمكانيات مثيرة، إلا أنها تطرح أيضاً العديد من التحديات والاعتبارات:

٠١. القيود التقنية:

- تقنية الذكاء الاصطناعي الحالية، رغم تقدمها، لا تزال لديها قيود في فهم نوايا المستخدم المعقدة وتوليد واجهات تدرك السياق حقاً.
- مشاكل الأداء المتعلقة بالتوليد في الوقت الفعلي لعناصر الواجهة، خاصة على الأجهزة الأقل قوة.

٠٢. متطلبات البيانات:

- اعتماداً على حالة الاستخدام، قد تتطلب أنظمة واجهات المستخدم التوليدية الفعالة كميات كبيرة من بيانات المستخدم لإنشاء واجهات مخصصة.
- تثير التحديات في الحصول على بيانات المستخدم بشكل أخلاقي مخاوف بشأن خصوصية البيانات وأمنها، بالإضافة إلى التحيزات المحتملة في البيانات المستخدمة لتدريب نماذج واجهات المستخدم التوليدية.

٠٣. قابلية الاستخدام والاتساق:

- على الأقل حتى تصبح هذه الممارسة منتشرة، قد يؤدي التطبيق ذو الواجهات المتغيرة باستمرار إلى مشاكل في قابلية الاستخدام، حيث قد يجد المستخدمون صعوبة في العثور على العناصر المألوفة أو التنقل بكفاءة.
- سيكون تحقيق التوازن بين التخصيص والحفاظ على واجهة متسقة وقابلة للتعليم أمراً حاسماً.

٤. الاعتماد المفرط على الذكاء الاصطناعي:

- هناك خطر في تفويض قرارات التصميم بشكل مفرط لأنظمة الذكاء الاصطناعي، مما قد يؤدي إلى خيارات واجهة غير ملهمة أو إشكالية أو معطلة.
- سيظل الإشراف البشري والقدرة على تجاوز التصميمات المولدة بالذكاء الاصطناعي مهماً في المستقبل المنظور.

٥. مخاوف إمكانية الوصول:

- يمثل ضمان بقاء الواجهات المولدة ديناميكياً متاحة للمستخدمين ذوي الإعاقة تحديات جديدة تماماً، وهو أمر مقلق نظراً لضعف مستوى الامتثال لمعايير إمكانية الوصول في الأنظمة النموذجية.
- من ناحية أخرى، يمكن تنفيذ مصممي الذكاء الاصطناعي مع اهتمام مدجج بإمكانية الوصول، وقدرات لبناء واجهات يمكن الوصول إليها فوراً تماماً كما يبنون واجهات المستخدم للمستخدمين غير المعاقين.
- في كلتا الحالتين، يجب تصميم أنظمة واجهات المستخدم التوليدية مع إرشادات وعمليات اختبار قوية لإمكانية الوصول.

٦. ثقة المستخدم والشفافية:

- قد يشعر المستخدمون بعدم الارتياح من الواجهات التي تبدو أنها "تعرف الكثير" عنهم أو تتغير بطرق لا يفهمونها.
- سيكون توفير الشفافية حول كيفية وسبب تخصيص الواجهات مهماً لبناء ثقة المستخدم.

النظرة المستقبلية والفرص

يحمل مستقبل واجهات المستخدم التوليدية وعداً هائلاً لإحداث ثورة في طريقة تفاعلنا مع المنتجات والخدمات الرقمية. مع استمرار تطور هذه التقنية، يمكننا توقع تحول كبير في كيفية تصميم وتنفيذ وتجربة واجهات المستخدم. نعتقد أن واجهات المستخدم التوليدية هي الظاهرة التي ستدفع برمجياتنا أخيراً إلى مجال ما يعتبر حالياً خيالاً علمياً.

أحد أكثر الآفاق إثارة لواجهات المستخدم التوليدية هو إمكانياتها في تعزيز إمكانية الوصول على نطاق واسع يتجاوز مجرد ضمان عدم استبعاد الأشخاص ذوي الإعاقات الشديدة من استخدام برمجياتك. من خلال تكييف الواجهات تلقائياً لاحتياجات المستخدم الفردية، يمكن لواجهات المستخدم التوليدية أن تجعل التجارب الرقمية أكثر شمولاً من أي وقت مضى. تخيل واجهات تتكيف بسلاسة لتوفير نص أكبر للمستخدمين الصغار أو ضعايف البصر أو تخطيطات مبسطة لذوي الإعاقات الإدراكية، كل ذلك دون الحاجة إلى تكوين يدوي أو إصدارات "يمكن الوصول إليها" منفصلة من التطبيقات.

من المحتمل أن تؤدي قدرات التخصيص في واجهات المستخدم التوليدية إلى زيادة مشاركة المستخدم ورضاه وولائه عبر مجموعة واسعة من المنتجات الرقمية. مع أصبحت الواجهات أكثر تناعماً مع التفضيلات والسلوكيات الفردية، سيجد المستخدمون التجارب الرقمية أكثر بديهية ومتعة، مما قد يؤدي إلى تفاعلات أعمق وأكثر معنى مع التكنولوجيا.

كما تمتلك واجهات المستخدم التوليدية القدرة على تحويل عملية تأهيل المستخدمين الجدد. من خلال إنشاء تجارب مستخدم أولية بديهية ومخصصة تتكيف بسرعة مع مستوى خبرة كل مستخدم، يمكن لواجهات المستخدم التوليدية تقليل منحنى التعلم المرتبط بالتطبيقات الجديدة بشكل كبير. يمكن أن يؤدي هذا إلى معدلات تبني أسرع وزيادة ثقة المستخدم في استكشاف الميزات والوظائف الجديدة.

إمكانية أخرى مثيرة هي قدرة واجهات المستخدم التوليدية على الحفاظ على تجربة مستخدم متسقة عبر الأجهزة والمنصات المختلفة مع تحسين كل سياق استخدام محدد. يمكن أن يحل هذا التحدي طويل الأمد المتمثل في توفير تجارب متماسكة عبر مشهد الأجهزة المتزايد التجزؤ، من الهواتف الذكية والأجهزة اللوحية إلى أجهزة الكمبيوتر المكتبية والتقنيات الناشئة مثل نظارات الواقع المعزز.

تفتح الطبيعة المعتمدة على البيانات لواجهات المستخدم التوليدية فرصاً للتكرار والتحسين السريع في تصميم واجهة المستخدم. من خلال جمع بيانات في الوقت الفعلي حول كيفية تفاعل المستخدمين مع الواجهات المولدة، يمكن للمصممين والمطورين اكتساب رؤى غير مسبقة حول سلوك المستخدم وتفضيلاته. يمكن

أن تؤدي حلقة التغذية الراجعة هذه إلى تحسينات مستمرة في تصميم واجهة المستخدم، مدفوعة بأنماط الاستخدام الفعلية بدلاً من الافتراضات أو اختبارات المستخدم المحدودة.

للتحضير لهذا التحول، سيحتاج المصممون إلى تطوير مهاراتهم وعقلياتهم. سيتحول التركيز من إنشاء تخطيطات ثابتة إلى تطوير أنظمة وإرشادات تصميم شاملة يمكن أن توجه توليد الواجهة المدعوم بالذكاء الاصطناعي. سيحتاج المصممون إلى تنمية فهم عميق لتحليل البيانات وتقنيات الذكاء الاصطناعي والتفكير المنظومي للتوجيه الفعال لأنظمة واجهات المستخدم التوليدية.

علاوة على ذلك، مع تلاشي الحدود بين التصميم والتكنولوجيا، سيحتاج المصممون إلى التعاون بشكل أوثق مع المطورين وعلماء البيانات. سيكون هذا النهج متعدد التخصصات حاسماً في إنشاء أنظمة واجهات مستخدم توليدية ليست جذابة بصرياً وسهلة الاستخدام فحسب، بل أيضاً قوية تقنياً وسليمة أخلاقياً.

ستبرز التداعيات الأخلاقية لتقنية GenUI إلى الواجهة مع نضوج هذه التكنولوجيا. وسيلعب المصممون دوراً حاسماً في تطوير أطر العمل للاستخدام المسؤول للذكاء الاصطناعي في تصميم الواجهات، مع ضمان أن يعزز التخصيص تجارب المستخدمين دون المساس بالخصوصية أو التلاعب بسلوك المستخدم بطرق غير أخلاقية.

وبينما نتطلع إلى المستقبل، تقدم تقنية GenUI فرصاً مثيرة وتحديات كبيرة على حد سواء. فلديها القدرة على خلق تجارب رقمية أكثر بداهة وكفاءة وإرضاء للمستخدمين في جميع أنحاء العالم. وفي حين أنها ستتطلب من المصممين التكيف واكتساب مهارات جديدة، إلا أنها تقدم أيضاً فرصة غير مسبقة لتشكيل مستقبل التفاعل بين الإنسان والحاسوب بطرق عميقة وذات مغزى. ولا شك أن الرحلة نحو أنظمة GenUI المتطورة بالكامل ستكون معقدة، ولكن المكافآت المحتملة من حيث تحسين تجارب المستخدم وإمكانية الوصول الرقمي تجعل هذا المستقبل يستحق السعي من أجله.

تنسيق سير العمل الذكي



في مجال تطوير التطبيقات ، يلعب سير العمل دوراً حاسماً في تحديد كيفية هيكلة وتنفيذ المهام والعمليات وتفاعلات المستخدمين. ومع تزايد تعقيد التطبيقات وارتفاع توقعات المستخدمين، تزداد الحاجة إلى تنسيق ذكي وقابل للتكيف لسير العمل.

يركز نهج "تنسيق سير العمل الذكي" على الاستفادة من مكونات الذكاء الاصطناعي لتنسيق وتحسين مسارات العمل المعقدة داخل التطبيقات بشكل ديناميكي. الهدف هو إنشاء تطبيقات أكثر كفاءة واستجابة وقدرة على التكيف مع البيانات والسياق في الوقت الفعلي.

في هذا الفصل، سنستكشف المبادئ والأنماط الرئيسية التي تدعم نهج تنسيق سير العمل الذكي. سننظر في كيفية استخدام الذكاء الاصطناعي لتوجيه المهام بذكاء، وأتمتة صنع القرار، وتكييف مسارات العمل

بشكل ديناميكي بناءً على عوامل مختلفة مثل سلوك المستخدم، وأداء النظام، والقواعد التجارية . من خلال الأمثلة العملية والسيناريوهات الواقعية، سنوضح الإمكانيات التحويلية للذكاء الاصطناعي في تبسيط وتحسين سير عمل التطبيقات.

سواء كنت تبني تطبيقات مؤسسية ذات عمليات تجارية معقدة أو تطبيقات موجهة للمستهلكين مع رحلات مستخدم ديناميكية، فإن الأنماط والتقنيات التي نناقشها في هذا الفصل ستزودك بالمعرفة والأدوات اللازمة لإنشاء مسارات عمل ذكية وفعالة تعزز تجربة المستخدم الشاملة وتدفع القيمة التجارية.

الحاجة التجارية

غالباً ما تعتمد الأساليب التقليدية لإدارة سير العمل على قواعد محددة مسبقاً وأشجار قرار ثابتة ، والتي يمكن أن تكون جامدة وغير مرنة وغير قادرة على التعامل مع الطبيعة الديناميكية للتطبيقات الحديثة.

فكر في سيناريو حيث يحتاج تطبيق للتجارة الإلكترونية إلى معالجة عملية معقدة لتلبية الطلبات. قد يتضمن سير العمل خطوات متعددة مثل التحقق من صحة الطلب، وفحص المخزون، ومعالجة الدفع، والشحن، وإخطارات العملاء. قد تحتوي كل خطوة على مجموعتها الخاصة من القواعد والتبعيات والتكاملات الخارجية وآليات معالجة الاستثناءات. يمكن أن تصبح إدارة مثل هذا سير العمل يدوياً أو من خلال منطق مبرمج بشكل ثابت مرهقة بسرعة وعرضة للأخطاء ويصعب صيانتها.

علاوة على ذلك، مع نمو التطبيق وزيادة عدد المستخدمين المتزامنين، قد يحتاج سير العمل إلى التكيف وتحسين نفسه بناءً على البيانات في الوقت الفعلي وأداء النظام. على سبيل المثال، خلال فترات الذروة في حركة المرور، قد يحتاج التطبيق إلى تعديل سير العمل ديناميكياً لإعطاء الأولوية لمهام معينة، وتخصيص الموارد بكفاءة، وضمان تجربة مستخدم سلسة.

هنا يأتي دور نهج "تنسيق سير العمل الذكي". من خلال الاستفادة من مكونات الذكاء الاصطناعي، يمكن للمطورين إنشاء مسارات عمل ذكية وقابلة للتكيف وذاتية التحسين. يمكن للذكاء الاصطناعي تحليل كميات هائلة من البيانات، والتعلم من التجارب السابقة، واتخاذ قرارات مستنيرة في الوقت الفعلي لتنسيق سير العمل بشكل فعال.

الفوائد الرئيسية

1. زيادة الكفاءة: يمكن للذكاء الاصطناعي تحسين تخصيص المهام، واستخدام الموارد، وتنفيذ سير العمل، مما يؤدي إلى أوقات معالجة أسرع وتحسين الكفاءة الشاملة.
2. القابلية للتكيف: يمكن لمسارات العمل المدفوعة بالذكاء الاصطناعي التكيف ديناميكياً مع الظروف المتغيرة، مثل التقلبات في طلب المستخدم، وأداء النظام، أو المتطلبات التجارية، مما يضمن أن يظل التطبيق مستجيباً ومرناً.
3. صنع القرار الآلي: يمكن للذكاء الاصطناعي أتمتة عمليات صنع القرار المعقدة داخل سير العمل، مما يقلل من التدخل اليدوي ويقلل من مخاطر الأخطاء البشرية.
4. التخصيص: يمكن للذكاء الاصطناعي تحليل سلوك المستخدم وتفضيلاته والسياق لتخصيص سير العمل وتقديم تجارب مخصصة للمستخدمين الفرديين.
5. قابلية التوسع: يمكن لمسارات العمل المدعومة بالذكاء الاصطناعي التوسع بسلاسة للتعامل مع الأحمال المتزايدة من البيانات وتفاعلات المستخدمين، دون المساس بالأداء أو الموثوقية.

في الأقسام التالية، سنستكشف الأنماط والتقنيات الرئيسية التي تمكن من تنفيذ مسارات العمل الذكية ونعرض أمثلة من العالم الحقيقي لكيفية تحويل الذكاء الاصطناعي لإدارة سير العمل في التطبيقات الحديثة.

الأنماط الرئيسية

لتنفيذ تنسيق سير العمل الذكي في التطبيقات، يمكن للمطورين الاستفادة من عدة أنماط رئيسية تسخر قوة الذكاء الاصطناعي. توفر هذه الأنماط نهجاً منظماً لتصميم وإدارة مسارات العمل، مما يمكن التطبيقات من التكيف والتحسين وأتمتة العمليات بناءً على البيانات والسياق في الوقت الفعلي. دعونا نستكشف بعض الأنماط الأساسية في تنسيق سير العمل الذكي.

التوجيه الديناميكي للمهام

يتضمن هذا النمط استخدام الذكاء الاصطناعي لتوجيه المهام بذكاء داخل سير العمل بناءً على عوامل مختلفة مثل أولوية المهمة، وتوفر الموارد، وأداء النظام. يمكن لخوارزميات الذكاء الاصطناعي تحليل خصائص

كل مهمة، والنظر في الحالة الحالية للنظام، واتخاذ قرارات مستنيرة لتعيين المهام للموارد أو مسارات المعالجة الأكثر ملاءمة. يضمن التوجيه الديناميكي للمهام توزيع المهام وتنفيذها بكفاءة، مما يحسن أداء سير العمل الشامل.

```

1  class TaskRouter
2      include Raix::ChatCompletion
3      include Raix::FunctionDispatch
4
5      attr_accessor :task
6
7      its at entirely AI the by called be can that functions of list #
8      received task the on depending discretion #
9
10     function :analyze_task_priority do
11         TaskPriorityAnalyzer.perform(task)
12     end
13
14     function :check_resource_availability, ... #
15     function :assess_system_performance, ... #
16     function :assign_task_to_resource, ... #
17
18     DIRECTIVE = "intelligently for responsible ,router task a are You
19     resource ,priority on based resources available to tasks assigning
20     performance... system and ,availability  "
21
22     def initialize(task)
23         self.task = task
24         transcript << { system: DIRECTIVE }
25         transcript << { user: task.to_json }
26     end
27
28     def perform
29         while task.unassigned?
30             chat_completion
31
32             break and counter loop max add todo: #
33         end
34
35         analysis later for transcript the capture #
36         task.update(routing_transcript: transcript)
37     end

```

لاحظ الحلقة التي تم إنشاؤها بواسطة تعبير while في السطر 29، والتي تستمر في مطالبة الذكاء الاصطناعي حتى يتم تعيين المهمة. في السطر 35، نقوم بحفظ النص الكامل للمهمة للتحليل وتصحيح الأخطاء لاحقاً إذا لزم الأمر.

اتخاذ القرارات السياقية

يمكنك استخدام كود مشابه جداً لاتخاذ قرارات تعتمد على السياق ضمن سير العمل. من خلال تحليل نقاط البيانات ذات الصلة مثل تفضيلات المستخدم والأنماط التاريخية والمدخلات في الوقت الفعلي، يمكن لمكونات الذكاء الاصطناعي تحديد مسار العمل الأنسب عند كل نقطة قرار في سير العمل. قم بتكييف سلوك سير العمل الخاص بك بناءً على السياق المحدد لكل مستخدم أو سيناريو، مما يوفر تجارب مخصصة ومحسنة.

تكوين سير العمل التكيفي

يركز هذا النمط على تكوين وتعديل مسارات العمل بشكل ديناميكي بناءً على المتطلبات أو الظروف المتغيرة. يمكن للذكاء الاصطناعي تحليل الحالة الحالية لسير العمل، وتحديد الاختناقات أو أوجه عدم الكفاءة، وتعديل هيكل سير العمل تلقائياً لتحسين الأداء. يسمح تكوين سير العمل التكيفي للتطبيقات بالتطور المستمر وتحسين عملياتها دون الحاجة إلى تدخل يدوي.

معالجة الاستثناءات والتعافي

تعد معالجة الاستثناءات والتعافي جوانب حاسمة في تنسيق سير العمل الذكي. عند العمل مع مكونات الذكاء الاصطناعي ومسارات العمل المعقدة، من الضروري توقع الاستثناءات ومعالجتها بأناقة لضمان استقرار النظام وموثوقيته.

فيما يلي بعض الاعتبارات والتقنيات الرئيسية لمعالجة الاستثناءات والتعافي في مسارات العمل الذكية:

1. نشر الاستثناءات: تنفيذ نهج متسق لنشر الاستثناءات عبر مكونات سير العمل. عندما يحدث استثناء داخل أحد المكونات، يجب التقاطه وتسجيله ونشره إلى المنسق أو المكون المنفصل المسؤول عن معالجة الاستثناءات. الفكرة هي مركزة معالجة الاستثناءات ومنع ابتلاع الاستثناءات بصمت، بالإضافة إلى فتح إمكانيات معالجة الأخطاء الذكية.
2. آليات إعادة المحاولة: تساعد آليات إعادة المحاولة في تحسين مرونة سير العمل ومعالجة الأعطال المؤقتة بأنافة. من المؤكد أنه يجب تنفيذ آليات إعادة المحاولة للاستثناءات العابرة أو القابلة للتعافي، مثل اتصال الشبكة أو عدم توفر الموارد التي يمكن إعادة محاولتها تلقائياً بعد تأخير محدد. إن وجود منسق أو معالج استثناءات مدعوم بالذكاء الاصطناعي يعني أن استراتيجيات إعادة المحاولة الخاصة بك لا يجب أن تكون آلية بطبيعتها، معتمدة على خوارزميات ثابتة مثل التراجع الأسّي. يمكنك ترك معالجة إعادة المحاولة وفقاً "لتقدير" مكون الذكاء الاصطناعي المسؤول عن تحديد كيفية معالجة الاستثناء.
3. استراتيجيات احتياطية: إذا فشل مكون الذكاء الاصطناعي في تقديم استجابة صالحة أو واجه خطأ - وهو أمر شائع نظراً لطبيعته المتطورة - فيجب أن يكون لديك آلية احتياطية لضمان استمرار سير العمل. قد يتضمن ذلك استخدام قيم افتراضية أو خوارزميات بديلة أو إشراك الإنسان في الحلقة لاتخاذ القرارات والحفاظ على تقدم سير العمل.
4. الإجراءات التعويضية: يجب أن تتضمن توجيهات المنسق تعليمات حول الإجراءات التعويضية لمعالجة الاستثناءات التي لا يمكن حلها تلقائياً. الإجراءات التعويضية هي خطوات تُتخذ للتراجع عن آثار العملية الفاشلة أو تخفيفها. على سبيل المثال، إذا فشلت خطوة معالجة الدفع، يمكن أن يكون الإجراء التعويضي هو التراجع عن المعاملة وإخطار المستخدم. تساعد الإجراءات التعويضية في الحفاظ على اتساق البيانات وسلامتها في مواجهة الاستثناءات.
5. مراقبة الاستثناءات والتنبيه: قم بإعداد آليات المراقبة والتنبيه لاكتشاف الاستثناءات الحرجة وإخطار أصحاب المصلحة المعنيين بها. يمكن جعل المنسق على دراية بالعتبات والقواعد لإطلاق التنبيهات عندما تتجاوز الاستثناءات حدوداً معينة أو عند حدوث أنواع محددة من الاستثناءات. يسمح ذلك بالتحديد الاستباقي للمشكلات وحلها قبل أن تؤثر على النظام بأكمله.

فيما يلي مثال على معالجة الاستثناءات والتعافي في مكون سير العمل بلغة Ruby:

```

1  class InventoryManager
2      def check_availability(order)
3          begin
4              logic check inventory Perform #
5              inventory = Inventory.find_by(product_id: order.product_id)
6              if inventory.available_quantity >= order.quantity
7                  return true
8              else
9                  raise InsufficientInventoryError,
10                     " product for inventory Insufficient#{order.product_id}"
11              end
12          rescue InsufficientInventoryError => e
13              exception the Log #
14              logger.error(" failed: check Inventory#{e.message}")
15
16              delay a after operation the Retry #
17              retry_count ||= 0
18              if retry_count < MAX_RETRIES
19                  retry_count += 1
20                  sleep(RETRY_DELAY)
21                  retry
22              else
23                  intervention manual to Fallback #
24                  NotificationService.admin(" Order failed: check Inventory#{order.id}")
25                  return false
26              end
27          end
28      end
29  end

```

في هذا المثال، يقوم مكوّن InventoryManager بالتحقق من توفر المنتج للطلب المحدد. إذا كانت الكمية المتوفرة غير كافية، فإنه يثير خطأ InsufficientInventoryError. يتم التقاط الاستثناء وتسجيله، وتنفيذ آلية إعادة المحاولة. إذا تم تجاوز حد إعادة المحاولة، ينتقل المكون إلى التدخل اليدوي عن طريق إخطار المسؤول.

من خلال تنفيذ آليات قوية لمعالجة الاستثناءات والتعافي، يمكنك ضمان أن تكون سير العمل الذكية لديك مرنة وقابلة للصيانة وقادرة على التعامل مع المواقف غير المتوقعة بسلاسة.

تشكل هذه الأنماط أساس تنسيق سير العمل الذكي ويمكن دمجها وتكييفها لتناسب المتطلبات المحددة للتطبيقات المختلفة. من خلال الاستفادة من هذه الأنماط، يمكن للمطورين إنشاء مسارات عمل مرنة ومتينة ومحسّنة للأداء وتجربة المستخدم.

في القسم التالي، سنستكشف كيفية تنفيذ هذه الأنماط عملياً، باستخدام أمثلة من العالم الحقيقي ومقتطفات برمجية لتوضيح دمج مكونات الذكاء الاصطناعي في إدارة سير العمل.

تنفيذ تنسيق سير العمل الذكي عملياً

الآن بعد أن استكشفنا الأنماط الرئيسية في تنسيق سير العمل الذكي، دعونا نتعمق في كيفية تنفيذ هذه الأنماط في التطبيقات العملية. سنقدم أمثلة عملية ومقتطفات برمجية لتوضيح دمج مكونات الذكاء الاصطناعي في إدارة سير العمل.

معالج الطلبات الذكي

دعونا نتعمق في مثال عملي لتنفيذ تنسيق سير العمل الذكي باستخدام مكون `OrderProcessor` مدعوم بالذكاء الاصطناعي في تطبيق التجارة الإلكترونية `Rails on Ruby`. يحقق `OrderProcessor` مفهوم **مدير العمليات للتكامل المؤسسي** الذي صادفناه لأول مرة في الفصل 3 عند مناقشة **تعدد العمال**. سيكون المكون مسؤولاً عن إدارة سير عمل تنفيذ الطلبات، واتخاذ قرارات التوجيه بناءً على النتائج المرحلية، وتنسيق تنفيذ خطوات المعالجة المختلفة.

تتضمن عملية تنفيذ الطلب خطوات متعددة مثل التحقق من صحة الطلب، والتحقق من المخزون، ومعالجة الدفع، والشحن. يتم تنفيذ كل خطوة كعملية عامل منفصلة تؤدي مهمة محددة وتعيد النتيجة إلى `OrderProcessor`. الخطوات ليست إلزامية، ولا يتعين حتى تنفيذها بترتيب محدد.

إليك مثال على تنفيذ `OrderProcessor`. يتميز باثنين من الـ `mixins` من `Raix`. الأول (`ChatCompletion`) يمنحه القدرة على إكمال المحادثة، وهذا ما يجعل منه مكوناً للذكاء الاصطناعي. والثاني (`FunctionDispatch`) يمكّن استدعاء الدوال بواسطة الذكاء الاصطناعي، مما يسمح له بالرد على الطلب باستدعاء دالة بدلاً من رسالة نصية.

تفوض دوال العامل (`order_validate`، و `inventory_check`، وما إلى ذلك) إلى فئات العمال الخاصة بها، والتي يمكن أن تكون مكونات ذكاء اصطناعي أو غير ذكاء اصطناعي، مع المتطلب الوحيد وهو أن

تعيد نتائج عملها في تنسيق يمكن تمثيله كسلسلة نصية.

كما هو الحال مع جميع الأمثلة الأخرى في هذا الجزء من الكتاب، هذا الكود هو في الأساس شبه كود ويهدف فقط إلى نقل معنى النمط وإلهام إبداعاتك الخاصة. يتم تضمين الأوصاف الكاملة للأنماط وأمثلة الكود الكاملة في الجزء 2.



```

1  class OrderProcessor
2      include Raix::ChatCompletion
3      include Raix::FunctionDispatch
4
5      SYSTEM_DIRECTIVE = "with... tasked ,processor order an are You"
6
7      def initialize(order)
8          self.order = order
9          transcript << { system: SYSTEM_DIRECTIVE }
10         transcript << { user: order.to_json }
11     end
12
13     def perform
14         called is ``stop_looping! until looping continue will #
15         chat_completion(loop: true)
16     end
17
18     AI the by called be to available functions of list #
19     brevity for truncated #
20
21     def functions
22         [
23             {
24                 name: "validate_order",
25                 description: "order of validity check to Invoke",
26                 parameters: {
27                     ...
28                 },
29                 ...
30             ]
31     end
32
33     AI the by called be can that functions of implementation #

```

```

34     order the of needs the on depending ,discretion its at entirely #
35
36     def validate_order
37         OrderValidationWorker.perform(@order)
38     end
39
40     def check_inventory
41         InventoryCheckWorker.perform(@order)
42     end
43
44     def process_payment
45         PaymentProcessingWorker.perform(@order)
46     end
47
48     def schedule_shipping
49         ShippingSchedulerWorker.perform(@order)
50     end
51
52     def send_confirmation
53         OrderConfirmationWorker.perform(@order)
54     end
55
56     def finished_processing
57         @order.update!(transcript:, processed_at: Time.current)
58         stop_looping!
59     end
60 end

```

في هذا المثال، يتم تهيئة معالج الطلبات بكائن الطلب ويحتفظ بنص سجل لتنفيذ سير العمل، وذلك بتنسيق نص المحادثة النموذجي الذي تستخدمه النماذج اللغوية الكبيرة. يتم منح التحكم الكامل للذكاء الاصطناعي لتنظيم تنفيذ خطوات المعالجة المختلفة، مثل التحقق من صحة الطلب، والتحقق من المخزون، ومعالجة الدفع، والشحن.

في كل مرة يتم فيها استدعاء طريقة completion_chat، يتم إرسال نص السجل إلى الذكاء الاصطناعي ليقدّم إكمالاً على شكل استدعاء دالة. يعود الأمر بالكامل إلى الذكاء الاصطناعي لتحليل نتيجة الخطوة السابقة وتحديد الإجراء المناسب الذي يجب اتخاذه. على سبيل المثال، إذا كشف فحص المخزون عن مستويات مخزون منخفضة، يمكن لـ OrderProcessor جدولة مهمة إعادة التزويد. إذا فشلت معالجة الدفع، يمكنه بدء إعادة المحاولة أو إخطار دعم العملاء.

A > يُذكر أن المثال أعلاه لا يحتوي على دوال محددة لإعادة التزويد أو إخطار دعم العملاء، ولكن من الممكن تماماً إضافتها.

ينمو نص السجل في كل مرة يتم فيها استدعاء دالة ويعمل كسجل لتنفيذ سير العمل، بما في ذلك نتائج كل خطوة والتعليمات التي ينشئها الذكاء الاصطناعي للخطوات التالية. يمكن استخدام نص السجل هذا لتصحيح الأخطاء والتدقيق وتوفير رؤية واضحة لعملية تنفيذ الطلب.

من خلال الاستفادة من الذكاء الاصطناعي في OrderProcessor، يمكن لتطبيق التجارة الإلكترونية تكييف سير العمل ديناميكياً بناءً على البيانات في الوقت الفعلي والتعامل مع الاستثناءات بذكاء. يمكن لمكون الذكاء الاصطناعي اتخاذ قرارات مدروسة، وتحسين سير العمل، وضمان معالجة سلسلة للطلبات حتى في السيناريوهات المعقدة.

حقيقة أن المتطلب الوحيد لعمليات المعالجة هو إرجاع مخرجات مفهومة للذكاء الاصطناعي للنظر فيها عند تحديد ما يجب فعله بعد ذلك، قد تبدأ في إدراك كيف يمكن لهذا النهج تقليل عمل تخطيط المدخلات/المخرجات الذي عادةً ما يكون مطلوباً عند دمج الأنظمة المختلفة مع بعضها البعض.

مراقب المحتوى الذكي

تتطلب تطبيقات وسائل التواصل الاجتماعي عموماً حداً أدنى من مراقبة المحتوى لضمان مجتمع آمن وصحي. يستفيد مكون ContentModerator هذا من الذكاء الاصطناعي لتنظيم سير عمل المراقبة بذكاء، واتخاذ القرارات بناءً على خصائص المحتوى ونتائج خطوات المراقبة المختلفة.

تتضمن عملية المراقبة خطوات متعددة مثل تحليل النص، والتعرف على الصور، وتقييم سمعة المستخدم، والمراجعة اليدوية. يتم تنفيذ كل خطوة كعملية معالجة منفصلة تؤدي مهمة محددة وتعيد النتيجة إلى ContentModerator.

فيما يلي مثال على تنفيذ ContentModerator:

```

1  class ContentModerator
2      include Raix::ChatCompletion
3      include Raix::FunctionDispatch
4
5      SYSTEM_DIRECTIVE = ",manager process moderator content a are You
6      content... user-generated moderating in involved workflow the with tasked  "
7
8      def initialize(content)
9          @content = content
10         @transcript = [
11             { system: SYSTEM_DIRECTIVE },
12             { user: content.to_json }
13         ]
14     end
15
16     def perform
17         complete(@transcript)
18     end
19
20     def model
21         "openai/gpt-4"
22     end
23
24     AI the by called be to available functions of list #
25     brevity for truncated #
26
27     def functions
28         [
29             {
30                 name: "analyze_text",
31                 ... #
32             },
33             {
34                 name: "recognize_image",
35                 description: "images... describe to Invoke",
36                 ... #
37             },
38             {
39                 name: "assess_user_reputation",
40                 ... #
41             },
42             {

```

```

43         name: "escalate_to_manual_review",
44         ... #
45     },
46     {
47         name: "approve_content",
48         ... #
49     },
50     {
51         name: "reject_content",
52         ... #
53     }
54 ]
55 end
56
57 AI the by called be can that functions of implementation #
58 order the of needs the on depending ,discretion its at entirely #
59
60 def analyze_text
61     result = TextAnalysisWorker.perform(@content)
62     continue_with(result)
63 end
64
65 def recognize_image
66     result = ImageRecognitionWorker.perform(@content)
67     continue_with(result)
68 end
69
70 def assess_user_reputation
71     result = UserReputationWorker.perform(@content.user)
72     continue_with(result)
73 end
74
75 def escalate_to_manual_review
76     ManualReviewWorker.perform(@content)
77     @content.update!(status: 'pending', transcript: @transcript)
78 end
79
80 def approve_content
81     @content.update!(status: 'approved', transcript: @transcript)
82 end
83
84 def reject_content

```

```

85     @content.update!(status: 'rejected', transcript: @transcript)
86   end
87
88   private
89
90   def continue_with(result)
91     @transcript << { function: result }
92     complete(@transcript)
93   end
94 end

```

في هذا المثال، يتم تهيئة ContentModerator بكائن محتوى ويحتفظ بسجل إشراف في شكل محادثة. يتمتع مكون الذكاء الاصطناعي بتحكم كامل في سير عملية الإشراف، حيث يقرر الخطوات التي سيتم تنفيذها بناءً على خصائص المحتوى ونتائج كل خطوة.

تتضمن الدوال العاملة المتاحة للذكاء الاصطناعي لاستدعائها image_recognize و text_analyze و reputation_user_assess و review_manual_to_escalate تقوم كل دالة بتفويض المهمة إلى عملية عاملة مقابلة (ImageRecognitionWorker و TextAnalysisWorker وما إلى ذلك) وتضيف النتيجة إلى سجل الإشراف، باستثناء دالة التصعيد التي تعمل كحالة نهائية. وأخيراً، تعمل دالتا content_approve و content_reject أيضاً كحالات نهائية.

يقوم مكون الذكاء الاصطناعي بتحليل المحتوى وتحديد الإجراء المناسب الذي يجب اتخاذه. إذا كان المحتوى يحتوي على مراجع صور، يمكنه استدعاء العامل image_recognize للمساعدة في المراجعة البصرية. إذا حذر أي عامل من محتوى ضار محتمل، قد يقرر الذكاء الاصطناعي تصعيد المحتوى للمراجعة اليدوية أو رفضه مباشرة. ولكن اعتماداً على شدة التحذير، قد يختار الذكاء الاصطناعي استخدام نتائج تقييم سمعة المستخدم في تقرير كيفية التعامل مع المحتوى الذي لا يكون متأكداً منه. واعتماداً على حالة الاستخدام، ربما يتمتع المستخدمون الموثوق بهم بمزيد من المرونة في ما يمكنهم نشره. وهكذا دواليك...

كما هو الحال في مثال مدير العمليات السابق، يعمل سجل الإشراف كسجل لتنفيذ سير العمل، بما في ذلك نتائج كل خطوة والقرارات التي يولدها الذكاء الاصطناعي. يمكن استخدام هذا السجل للتدقيق والشفافية وتحسين عملية الإشراف بمرور الوقت.

من خلال الاستفادة من الذكاء الاصطناعي في ContentModerator، يمكن لتطبيق وسائل التواصل الاجتماعي تكيف سير عمل الإشراف بشكل ديناميكي بناءً على خصائص المحتوى والتعامل مع سيناريوهات الإشراف المعقدة بذكاء. يمكن لمكون الذكاء الاصطناعي اتخاذ قرارات مستنيرة، وتحسين

سير العمل، وضمان تجربة مجتمعية آمنة وصحية.

دعونا نستكشف مثالين آخرين يوضحان جدولة المهام التنبؤية ومعالجة الاستثناءات والتعافي في سياق تنسيق سير العمل الذكي.

جدولة المهام التنبؤية في نظام دعم العملاء

في تطبيق دعم العملاء المبني باستخدام ، تعد إدارة تذاكر الدعم وتحديد أولوياتها أمراً حاسماً لتقديم المساعدة في الوقت المناسب للعملاء. يستفيد مكون SupportTicketScheduler من الذكاء الاصطناعي لجدولة وتخصيص تذاكر الدعم بشكل تنبؤي للوكلاء المتاحين بناءً على عوامل مختلفة مثل إلحاح التذكرة وخبرة الوكيل وعقب العمل.

```

1  class SupportTicketScheduler
2      include Raix::ChatCompletion
3      include Raix::FunctionDispatch
4
5      SYSTEM_DIRECTIVE = ",scheduler ticket support a are You
6      agents... available to tickets assigning intelligently with tasked    "
7
8      def initialize(ticket)
9          @ticket = ticket
10         @transcript = [
11             { system: SYSTEM_DIRECTIVE },
12             { user: ticket.to_json }
13         ]
14     end
15
16     def perform
17         complete(@transcript)
18     end
19
20     def model
21         "openai/gpt-4"
22     end
23
24     def functions
25         [
26             {

```

```

27         name: "analyze_ticket_urgency",
28         ... #
29     },
30     {
31         name: "list_available_agents",
32         description: "agents available of expertise Includes",
33         ... #
34     },
35     {
36         name: "predict_agent_workload",
37         description: "workloads upcoming predict to data historical Uses",
38         ... #
39     },
40     {
41         name: "assign_ticket_to_agent",
42         ... #
43     },
44     {
45         name: "reschedule_ticket",
46         ... #
47     }
48 ]
49 end
50
51 AI the by called be can that functions of implementation #
52 order the of needs the on depending ,discretion its at entirely #
53
54 def analyze_ticket_urgency
55     result = TicketUrgencyAnalyzer.perform(@ticket)
56     continue_with(result)
57 end
58
59 def list_available_agents
60     result = ListAvailableAgents.perform
61     continue_with(result)
62 end
63
64 def predict_agent_workload
65     result = AgentWorkloadPredictor.perform
66     continue_with(result)
67 end
68

```

```

69 def assign_ticket_to_agent
70     TicketAssigner.perform(@ticket, @transcript)
71 end
72
73 def delay_assignment(until)
74     until = DateTimeStandardizer.process(until)
75     SupportTicketScheduler.delay(@ticket, @transcript, until)
76 end
77
78 private
79
80 def continue_with(result)
81     @transcript << { function: result }
82     complete(@transcript)
83 end
84 end

```

في هذا المثال، يتم تهيئة SupportTicketScheduler بكائن تذكرة دعم ويحتفظ بنص الجدولة. يقوم مكون الذكاء الاصطناعي بتحليل تفاصيل التذكرة وجدولة تعيين التذكرة بشكل تنبؤي بناءً على عوامل مثل إلحاح التذكرة، وخبرة الوكيل، وعبء العمل المتوقع للوكيل.

تتضمن الوظائف المتاحة للذكاء الاصطناعي لاستدعائها urgency_ticket_analyze، agents_available_list، workload_agent_predict، وagent_to_ticket_assign. تقوم كل وظيفة بتفويض المهمة إلى مكون محلل أو متنبئ مقابل وتضيف النتيجة إلى نص الجدولة. كما يمكن للذكاء الاصطناعي تأخير التعيين باستخدام وظيفة assignment_delay.

يقوم مكون الذكاء الاصطناعي بفحص نص الجدولة واتخاذ قرارات مدروسة بشأن تعيين التذاكر. يأخذ في الاعتبار مدى إلحاح التذكرة، وخبرة الوكلاء المتاحين، وعبء العمل المتوقع لكل وكيل لتحديد الوكيل الأنسب للتعامل مع التذكرة.

من خلال الاستفادة من جدولة المهام التنبؤية، يمكن لتطبيق دعم العملاء تحسين تعيين التذاكر، وتقليل أوقات الاستجابة، وتحسين رضا العملاء بشكل عام. تتضمن الإدارة الاستباقية والفعالة لتذاكر الدعم تعيين التذاكر المناسبة للوكلاء المناسبين في الوقت المناسب.

معالجة الاستثناءات والاستعادة في خط معالجة البيانات

تعد معالجة الاستثناءات والاستعادة من الأعطال أمراً ضرورياً لضمان سلامة البيانات ومنع فقدانها. يستخدم مكون `DataProcessingOrchestrator` الذكاء الاصطناعي لمعالجة الاستثناءات بذكاء وتنسيق عملية الاستعادة في خط معالجة البيانات

```

1  class DataProcessingOrchestrator
2      include Raix::ChatCompletion
3      include Raix::FunctionDispatch
4
5      SYSTEM_DIRECTIVE = "orchestrator... processing data a are You"
6
7      def initialize(data_batch)
8          @data_batch = data_batch
9          @transcript = [
10             { system: SYSTEM_DIRECTIVE },
11             { user: data_batch.to_json }
12         ]
13     end
14
15     def perform
16         complete(@transcript)
17     end
18
19     def model
20         "openai/gpt-4"
21     end
22
23     def functions
24         [
25             {
26                 name: "validate_data",
27                 ... #
28             },
29             {
30                 name: "process_data",
31                 ... #
32             },
33             {
34                 name: "request_fix",

```

```

35     ... #
36   },
37   {
38     name: "retry_processing",
39     ... #
40   },
41   {
42     name: "mark_data_as_failed",
43     ... #
44   },
45   {
46     name: "finished",
47     ... #
48   }
49 ]
50 end
51
52 AI the by called be can that functions of implementation #
53 order the of needs the on depending ,discretion its at entirely #
54
55 def validate_data
56   result = DataValidator.perform(@data_batch)
57   continue_with(result)
58 rescue ValidationException => e
59   handle_validation_exception(e)
60 end
61
62 def process_data
63   result = DataProcessor.perform(@data_batch)
64   continue_with(result)
65 rescue ProcessingException => e
66   handle_processing_exception(e)
67 end
68
69 def request_fix(description_of_fix)
70   result = SmartDataFixer.new(description_of_fix, @data_batch)
71   continue_with(result)
72 end
73
74 def retry_processing(timeout_in_seconds)
75   wait(timeout_in_seconds)
76   process_data

```

```

77     end
78
79     def mark_data_as_failed
80         @data_batch.update!(status: 'failed', transcript: @transcript)
81     end
82
83     def finished
84         @data_batch.update!(status: 'finished', transcript: @transcript)
85     end
86
87     private
88
89     def continue_with(result)
90         @transcript << { function: result }
91         complete(@transcript)
92     end
93
94     def handle_validation_exception(exception)
95         @transcript << { exception: exception.message }
96         complete(@transcript)
97     end
98
99     def handle_processing_exception(exception)
100         @transcript << { exception: exception.message }
101         complete(@transcript)
102     end
103 end

```

في هذا المثال، يتم تهيئة `DataProcessingOrchestrator` بكائن دفعة بيانات ويحتفظ بسجل المعالجة. يقوم مكون الذكاء الاصطناعي بتنسيق خط أنابيب معالجة البيانات، ويتعامل مع الاستثناءات ويتعافى من الأعطال حسب الحاجة.

تتضمن الدوال المتاحة للذكاء الاصطناعي لاستدعائها `data_validate` و `data_process` و `fix_request` و `processing_retry` و `failed._as_data_mark` تقوم كل دالة بتفويض المهمة إلى مكون معالجة البيانات المقابل وتضيف النتيجة أو تفاصيل الاستثناء إلى سجل المعالجة.

إذا حدث استثناء في التحقق أثناء خطوة `data_validate`، تقوم دالة `exception_validation_handle` بإضافة بيانات الاستثناء إلى السجل وتعيد التحكم إلى الذكاء الاصطناعي. وبالمثل، إذا حدث استثناء في المعالجة أثناء خطوة `data_process`، يمكن للذكاء الاصطناعي أن يقرر استراتيجية التعافي.

اعتماداً على طبيعة الاستثناء الذي تمت مواجهته، يمكن للذكاء الاصطناعي وفقاً لتقديره أن يقرر استدعاء `fix_request`، والذي يفوض إلى مكون `SmartDataFixer` المدعوم بالذكاء الاصطناعي (انظر فصل البيانات ذاتية التصحيح). يحصل مصصح البيانات على وصف باللغة الإنجليزية البسيطة لكيفية تعديل `batch_data` حتى يمكن إعادة محاولة المعالجة. ربما تتضمن إعادة المحاولة الناجحة إزالة السجلات من دفعة البيانات التي فشلت في التحقق و/أو نسخها إلى خط أنابيب معالجة مختلف للراجعة البشرية؟ الاحتمالات تكاد تكون لا نهائية.

من خلال دمج معالجة الاستثناءات والتعافي المدعوم بالذكاء الاصطناعي، يصبح تطبيق معالجة البيانات أكثر مرونة وتحملًا للأخطاء. يقوم `DataProcessingOrchestrator` بإدارة الاستثناءات بذكاء، ويقلل من فقدان البيانات، ويضمن التنفيذ السلس لسير عمل معالجة البيانات.

المراقبة وتسجيل السجلات

توفر المراقبة وتسجيل السجلات رؤية لتقدم وأداء وصحة مكونات سير العمل المدعومة بالذكاء الاصطناعي، مما يمكن المطورين من تتبع وتحليل سلوك النظام. يعد تنفيذ آليات فعالة للمراقبة وتسجيل السجلات أمراً ضرورياً لتصحيح الأخطاء والتدقيق والتحسين المستمر لسير العمل الذكي.

مراقبة تقدم وأداء سير العمل

لضمان التنفيذ السلس لسير العمل الذكي، من المهم مراقبة تقدم وأداء كل مكون من مكونات سير العمل. يتضمن ذلك تتبع المقاييس والأحداث الرئيسية خلال دورة حياة سير العمل.

تشمل بعض الجوانب المهمة للمراقبة ما يلي:

1. وقت تنفيذ سير العمل: قياس الوقت الذي يستغرقه كل مكون من مكونات سير العمل لإكمال مهمته. يساعد هذا في تحديد اختناقات الأداء وتحسين كفاءة سير العمل الإجمالية.
2. استخدام الموارد: مراقبة استخدام موارد النظام، مثل وحدة المعالجة المركزية والذاكرة والتخزين، من قبل كل مكون من مكونات سير العمل. يساعد هذا في ضمان أن النظام يعمل ضمن قدرته ويمكنه التعامل مع عبء العمل بفعالية.
3. معدلات الأخطاء والاستثناءات: تتبع حدوث الأخطاء والاستثناءات داخل مكونات سير العمل. يساعد هذا في تحديد المشكلات المحتملة ويمكن من معالجة الأخطاء والتعافي بشكل استباقي.

4. نقاط القرار والنتائج: مراقبة نقاط القرار داخل سير العمل ونتائج القرارات المدعومة بالذكاء الاصطناعي. يوفر هذا رؤى حول سلوك وفعالية مكونات الذكاء الاصطناعي. يمكن عرض البيانات التي تم جمعها بواسطة عمليات المراقبة في لوحات المعلومات أو استخدامها كمداخلات للتقارير المجدولة التي تُعلم مسؤولي النظام عن صحة النظام.

يمكن تغذية بيانات المراقبة إلى عملية مسؤول نظام مدعومة بالذكاء الاصطناعي للمراجعة والإجراء المحتمل!



تسجيل الأحداث والقرارات الرئيسية

تسجيل السجلات هو ممارسة أساسية تتضمن التقاط وتخزين المعلومات ذات الصلة حول الأحداث الرئيسية والقرارات والاستثناءات التي تحدث أثناء تنفيذ سير العمل.

تشمل بعض الجوانب المهمة للتسجيل ما يلي:

1. بدء وإكمال سير العمل: تسجيل أوقات البدء والانهاء لكل نسخة من سير العمل، مع أي بيانات وصفية ذات صلة مثل بيانات الإدخال وسياق المستخدم.
2. تنفيذ المكونات: تسجيل تفاصيل تنفيذ كل مكون من مكونات سير العمل، بما في ذلك معلمات الإدخال ونتائج الإخراج وأي بيانات وسيطة تم إنشاؤها.
3. قرارات الذكاء الاصطناعي والمنطق: تسجيل القرارات التي اتخذتها مكونات الذكاء الاصطناعي، مع المنطق الأساسي أو درجات الثقة. يوفر هذا الشفافية ويمكن من تدقيق القرارات المدعومة بالذكاء الاصطناعي.
4. الاستثناءات ورسائل الخطأ: تسجيل أي استثناءات أو رسائل خطأ تمت مواجهتها أثناء تنفيذ سير العمل، بما في ذلك تتبع المكس ومعلومات السياق ذات الصلة.

يمكن تنفيذ تسجيل السجلات باستخدام تقنيات مختلفة، مثل الكتابة في ملفات السجل، أو تخزين السجلات في قاعدة بيانات، أو إرسال السجلات إلى خدمة تسجيل مركزية. من المهم اختيار إطار عمل للتسجيل يوفر المرونة وقابلية التوسع والتكامل السهل مع هيكل التطبيق.

فيما يلي مثال على كيفية تنفيذ تسجيل السجلات في تطبيق Rails on Ruby باستخدام فئة ActiveSupport::Logger:

```

1 class WorkflowLogger
2   def self.log(message, severity = :info)
3     @logger ||= ActiveSupport::Logger.new('workflow.log')
4     @logger.formatter ||= proc do |severity, datetime, progname, msg|
5       "#{datetime}[ #{severity} ]#{msg}\n"
6     end
7     @logger.send(severity, message)
8   end
9 end
10
11 example Usage #
12 WorkflowLogger.log(" order for initiated Workflow#{@order.id}")
13 WorkflowLogger.log("successfully completed processing Payment")
14 WorkflowLogger.log(" item for failed check Inventory#{@item.id}", :error)

```

من خلال وضع عبارات التسجيل بشكل استراتيجي في مكونات سير العمل ونقاط قرار الذكاء الاصطناعي، يمكن للمطورين جمع معلومات قيمة لتصحيح الأخطاء والتدقيق والتحليل.

فوائد المراقبة والتسجيل

يقدم تنفيذ المراقبة والتسجيل في تنسيق سير العمل الذكي العديد من الفوائد:

1. **تصحيح الأخطاء واستكشاف المشكلات:** تساعد السجلات التفصيلية وبيانات المراقبة المطورين على تحديد وتشخيص المشكلات بسرعة. وهي توفر رؤى حول تدفق تنفيذ سير العمل، وتفاعلات المكونات، وأي أخطاء أو استثناءات تمت مواجهتها.
2. **تحسين الأداء:** يتيح مراقبة مقاييس الأداء للمطورين تحديد نقاط الاختناق وتحسين مكونات سير العمل لتحقيق كفاءة أفضل. من خلال تحليل أوقات التنفيذ، واستخدام الموارد، والمقاييس الأخرى، يمكن للمطورين اتخاذ قرارات مدروسة لتحسين الأداء العام للنظام.
3. **التدقيق والامتثال:** يوفر تسجيل الأحداث والقرارات الرئيسية سجلاً للتدقيق من أجل الامتثال التنظيمي والمساءلة. يمكن المؤسسات من تتبع والتحقق من الإجراءات التي تتخذها مكونات الذكاء الاصطناعي وضمان الالتزام بقواعد العمل والمتطلبات القانونية.
4. **التحسين المستمر:** تعمل بيانات المراقبة والتسجيل كمداخلات قيمة للتحسين المستمر لسير العمل الذكي. من خلال تحليل البيانات التاريخية، وتحديد الأنماط، وقياس فعالية قرارات الذكاء الاصطناعي، يمكن للمطورين تحسين وتعزيز منطق تنسيق سير العمل بشكل تكراري.

اعتبارات وأفضل الممارسات

عند تنفيذ المراقبة والتسجيل في تنسيق سير العمل الذكي، ضع في اعتبارك أفضل الممارسات التالية:

1. تحديد مقاييس مراقبة واضحة: حدد المقاييس والأحداث الرئيسية التي تحتاج إلى مراقبة بناءً على المتطلبات المحددة لسير العمل. ركز على المقاييس التي توفر رؤى ذات مغزى حول أداء النظام وصحته وسلوكه.

2. تنفيذ التسجيل التفصيلي: تأكد من وضع عبارات التسجيل في النقاط المناسبة داخل مكونات سير العمل ونقاط قرار الذكاء الاصطناعي. قم بتسجيل معلومات السياق ذات الصلة، مثل معلمات الإدخال، ونتائج الإخراج، وأي بيانات وسيطة يتم إنشاؤها.

3. استخدام التسجيل المنظم: اعتمد تنسيق تسجيل منظم لتسهيل تحليل وتصفية بيانات السجل. يتيح التسجيل المنظم إمكانية أفضل للبحث والتصفية وتجميع إدخلات السجل.

4. إدارة الاحتفاظ بالسجلات وتدويرها: قم بتنفيذ سياسات الاحتفاظ بالسجلات وتدويرها لإدارة تخزين ودورة حياة ملفات السجل. حدد فترة الاحتفاظ المناسبة بناءً على المتطلبات القانونية وقيود التخزين واحتياجات التحليل. إذا أمكن، قم بنقل التسجيل إلى خدمة طرف ثالث مثل [Papertrail](#).

5. تأمين المعلومات الحساسة: كن حذراً عند تسجيل المعلومات الحساسة، مثل المعلومات الشخصية القابلة للتحديد (PII) أو بيانات الأعمال السرية. قم بتنفيذ تدابير أمنية مناسبة، مثل إخفاء البيانات أو التشفير، لحماية المعلومات الحساسة في ملفات السجل.

6. التكامل مع أدوات المراقبة والتنبيه: استفد من أدوات المراقبة والتنبيه لمركزة جمع وتحليل وتصور بيانات المراقبة والتسجيل. يمكن لهذه الأدوات توفير رؤى في الوقت الفعلي، وإنشاء تنبيهات بناءً على عتبات محددة مسبقاً، وتسهيل الكشف الاستباقي عن المشكلات وحلها. أداتي المفضلة من بين هذه الأدوات هي [Datadog](#).

من خلال تنفيذ آليات شاملة للمراقبة والتسجيل، يمكن للمطورين الحصول على رؤى قيمة حول سلوك وأداء سير العمل الذكي. تمكن هذه الرؤى من التصحيح الفعال للأخطاء والتحسين والتطوير المستمر لأنظمة تنسيق سير العمل المدعومة بالذكاء الاصطناعي.

اعتبارات قابلية التوسع والأداء

تعد قابلية التوسع والأداء جوانب حاسمة يجب مراعاتها عند تصميم وتنفيذ أنظمة تنسيق سير العمل الذكي. مع زيادة حجم سير العمل المتزامن وتعقيد مكونات الذكاء الاصطناعي، يصبح من الضروري ضمان قدرة النظام على معالجة عبء العمل بكفاءة والتوسع بسلاسة لتلبية المتطلبات المتزايدة.

التعامل مع الأحجام الكبيرة من سير العمل المتزامن

غالباً ما تحتاج أنظمة تنسيق سير العمل الذكي إلى التعامل مع عدد كبير من عمليات سير العمل المتزامنة. لضمان قابلية التوسع، ضع في اعتبارك الاستراتيجيات التالية:

1. **المعالجة غير المتزامنة:** قم بتنفيذ آليات المعالجة غير المتزامنة لفصل تنفيذ مكونات سير العمل. يتيح هذا للنظام التعامل مع العديد من عمليات سير العمل في وقت واحد دون حظر أو انتظار اكتمال كل مكون. يمكن تحقيق المعالجة غير المتزامنة باستخدام قوائم انتظار الرسائل، أو البنى المعمارية القائمة على الأحداث، أو أطر معالجة المهام في الخلفية مثل Sidekiq.
2. **البنية الموزعة:** قم بتصميم بنية النظام لاستخدام المكونات اللاحادية (مثل AWS Lambda أو ببساطة توزيع عبء العمل عبر عدة عقد أو خوادم إلى جانب خادم التطبيق الرئيسي). يتيح هذا قابلية التوسع الأفقي، حيث يمكن إضافة عقد إضافية للتعامل مع أحجام سير العمل المتزايدة.
3. **التنفيذ المتوازي:** حدد فرص التنفيذ المتوازي داخل سير العمل. قد تكون بعض مكونات سير العمل مستقلة عن بعضها البعض ويمكن تنفيذها بشكل متزامن. من خلال الاستفادة من تقنيات المعالجة المتوازية، مثل تعدد مسارات التنفيذ أو قوائم انتظار المهام الموزعة، يمكن للنظام تحسين استخدام الموارد وتقليل وقت التنفيذ الإجمالي لسير العمل.

تحسين أداء المكونات المدعومة بالذكاء الاصطناعي

يمكن أن تكون المكونات المدعومة بالذكاء الاصطناعي، مثل نماذج التعلم الآلي أو محركات معالجة اللغة الطبيعية، كثيفة من حيث الحوسبة وتؤثر على الأداء العام لنظام تنسيق سير العمل. لتحسين أداء مكونات الذكاء الاصطناعي، ضع في اعتبارك التقنيات التالية:

1. التخزين المؤقت: إذا كانت معالجة الذكاء الاصطناعي لديك توليدية بحتة ولا تتضمن عمليات بحث في الوقت الفعلي أو تكاملات خارجية لتوليد إكالات المحادثة، فيمكنك النظر في آليات التخزين المؤقت لتخزين وإعادة استخدام نتائج العمليات التي يتم الوصول إليها بشكل متكرر أو المكلفة حسابياً.
2. تحسين النموذج: قم بتحسين الطريقة التي تستخدم بها نماذج الذكاء الاصطناعي في مكونات سير العمل بشكل مستمر. قد يتضمن ذلك تقنيات مثل تقطير الموجهات أو قد يكون مجرد مسألة اختبار نماذج جديدة عندما تصبح متاحة.
3. المعالجة الدفعية: إذا كنت تعمل مع نماذج من فئة 4GPT، فقد تتمكن من الاستفادة من تقنيات المعالجة الدفعية لمعالجة نقاط بيانات أو طلبات متعددة في دفعة واحدة، بدلاً من معالجتها بشكل فردي. من خلال معالجة البيانات في دفعات، يمكن للنظام تحسين استخدام الموارد وتقليل العبء الإضافي لطلبات النموذج المتكررة.

مراقبة وتحليل الأداء

لتحديد اختناقات الأداء وتحسين قابلية التوسع في نظام تنسيق سير العمل الذكي، من الضروري تنفيذ آليات المراقبة والتحليل. ضع في اعتبارك النهج التالية:

1. مقاييس الأداء: حدد وتابع مقاييس الأداء الرئيسية، مثل وقت الاستجابة، والإنتاجية، واستخدام الموارد، وزمن الاستجابة. توفر هذه المقاييس رؤى حول أداء النظام وتساعد في تحديد مجالات التحسين. يتضمن مجمع نماذج الذكاء الاصطناعي الشهير [OpenRouter](#) مقاييس المضيف¹ والسرعة² في كل استجابة API، مما يجعل تتبع هذه المقاييس الرئيسية أمراً بسيطاً.
2. أدوات التحليل: استخدم أدوات التحليل لتحليل أداء مكونات سير العمل الفردية وعمليات الذكاء الاصطناعي. يمكن لأدوات التحليل المساعدة في تحديد النقاط الساخنة للأداء، أو مسارات التعليمات البرمجية غير الفعالة، أو العمليات كثيفة الموارد. تشمل أدوات التحليل الشائعة Relic New و Scout أو المحللات المدججة التي توفرها لغة البرمجة أو الإطار.

¹المضيف هو الوقت المستغرق لاستلام البايت الأول من التوليد المتدفق من مضيف النموذج، أو ما يُعرف باسم "الوقت حتى البايت الأول".

²يتم حساب السرعة كعدد رموز الإكمال مقسوماً على إجمالي وقت التوليد. بالنسبة للطلبات غير المتدفقة، يُعتبر زمن الاستجابة جزءاً من وقت التوليد.

3. اختبار التحميل: قم بإجراء اختبار التحميل لتقييم أداء النظام تحت مستويات مختلفة من أحمال العمل المتزامنة. يساعد اختبار التحميل في تحديد حدود قابلية التوسع للنظام، واكتشاف تدهور الأداء، وضمان قدرة النظام على التعامل مع حركة المرور المتوقعة دون المساس بالأداء.
4. المراقبة المستمرة: قم بتنفيذ آليات المراقبة والتنبيه المستمرة للكشف الاستباقي عن مشاكل الأداء والاختناقات. قم بإعداد لوحات معلومات المراقبة والتنبيهات لتتبع مؤشرات الأداء الرئيسية (KPIs) وتلقي إشعارات عند تجاوز العتبات المحددة مسبقاً. يتيح ذلك التحديد والحل السريع لمشاكل الأداء.

استراتيجيات التوسع

للتعامل مع أحمال العمل المتزايدة وضمان قابلية التوسع لنظام تنسيق سير العمل الذكي، ضع في اعتبارك استراتيجيات التوسع التالية:

1. التوسع العمودي: يتضمن التوسع العمودي زيادة الموارد (مثل وحدة المعالجة المركزية والذاكرة) للعقد أو الخوادم الفردية للتعامل مع أحمال العمل الأعلى. هذا النهج مناسب عندما يتطلب النظام المزيد من قوة المعالجة أو الذاكرة للتعامل مع سير العمل المعقد أو عمليات الذكاء الاصطناعي.
2. التوسع الأفقي: يتضمن التوسع الأفقي إضافة المزيد من العقد أو الخوادم إلى النظام لتوزيع عبء العمل. هذا النهج فعال عندما يحتاج النظام إلى التعامل مع عدد كبير من سير العمل المتزامن أو عندما يمكن توزيع عبء العمل بسهولة عبر عقد متعددة. يتطلب التوسع الأفقي بنية موزعة وآليات موازنة التحميل لضمان التوزيع المتساوي لحركة المرور.
3. التوسع التلقائي: قم بتنفيذ آليات التوسع التلقائي لضبط عدد العقد أو الموارد تلقائياً بناءً على طلب عبء العمل. يسمح التوسع التلقائي للنظام بالتوسع تلقائياً أو التقليل اعتماداً على حركة المرور الواردة، مما يضمن الاستخدام الأمثل للموارد وكفاءة التكلفة. توفر منصات السحابة مثل Amazon Web Services (AWS) أو Google Cloud Platform (GCP) قدرات التوسع التلقائي التي يمكن الاستفادة منها لأنظمة تنسيق سير العمل الذكية.

تقنيات تحسين الأداء

بالإضافة إلى استراتيجيات التوسع، ضع في اعتبارك تقنيات تحسين الأداء التالية لتعزيز كفاءة نظام تنسيق سير العمل الذكي:

1. تخزين واسترجاع البيانات بكفاءة: قم بتحسين آليات تخزين واسترجاع البيانات المستخدمة بواسطة مكونات سير العمل. استخدم فهرسة قواعد البيانات الفعالة، وتقنيات تحسين الاستعلام، والتخزين المؤقت للبيانات لتقليل زمن الاستجابة وتحسين أداء العمليات كثيفة البيانات.
2. الإدخال/الإخراج غير المتزامن: استخدم عمليات الإدخال/الإخراج غير المتزامنة لمنع التوقف وتحسين استجابة النظام. يسمح الإدخال/الإخراج غير المتزامن للنظام بمعالجة طلبات متعددة في وقت واحد دون انتظار اكتمال عمليات الإدخال/الإخراج، مما يؤدي إلى تعظيم استخدام الموارد.
3. التسلسل وفك التسلسل الفعال: تحسين عمليات التسلسل وفك التسلسل المستخدمة لتبادل البيانات بين مكونات سير العمل. استخدم تنسيقات التسلسل الفعالة، مثل Buffers Protocol أو MessagePack، لتقليل العبء الزائد لتسلسل البيانات وتحسين أداء الاتصال بين المكونات.

بالنسبة للتطبيقات المعتمدة على Ruby، ضع في اعتبارك استخدام ID Universal. يستفيد ID Universal من كل من MessagePack و Brotli (مجموعة مبنية للسرعة وأفضل ضغط للبيانات) عند دمجهما، تكون هذه المكتبات أسرع بنسبة تصل إلى 30٪ وضمن معدلات ضغط 2-5٪ مقارنة بـ Buffers. Protocol



4. الضغط والترميز: تطبيق تقنيات الضغط والترميز لتقليل حجم البيانات المنقولة بين مكونات سير العمل. يمكن لخوارزميات الضغط، مثل gzip أو Brotli، أن تقلل بشكل كبير من استخدام عرض النطاق الترددي للشبكة وتحسين الأداء العام للنظام.
- من خلال مراعاة جوانب قابلية التوسع والأداء أثناء تصميم وتنفيذ أنظمة تنسيق سير العمل الذكية، يمكنك ضمان قدرة نظامك على معالجة أحجام كبيرة من مسارات العمل المتزامنة، وتحسين أداء المكونات المدعومة بالذكاء الاصطناعي، والتوسع بسلاسة لتلبية المتطلبات المتزايدة. تعد المراقبة المستمرة والتحليل والجهود التحسينية أمراً ضرورياً للحفاظ على أداء النظام واستجابته مع زيادة حجم العمل والتعقيد بمرور الوقت.

اختبار والتحقق من صحة مسارات العمل

يعد الاختبار والتحقق من الصحة جوانب حاسمة في تطوير وصيانة أنظمة تنسيق سير العمل الذكية. نظراً للطبيعة المعقدة لمسارات العمل المدعومة بالذكاء الاصطناعي، من الضروري ضمان عمل كل مكون كما هو

متوقع، وأن سير العمل العام يعمل بشكل صحيح، وأن قرارات الذكاء الاصطناعي دقيقة وموثوقة. في هذا القسم، سنستكشف التقنيات والاعتبارات المختلفة لاختبار والتحقق من صحة مسارات العمل الذكية.

اختبار وحدات مكونات سير العمل

يتضمن اختبار الوحدات اختبار مكونات سير العمل الفردية بشكل منعزل للتحقق من صحتها ومتانتها. عند اختبار مكونات سير العمل المدعومة بالذكاء الاصطناعي، ضع في اعتبارك ما يلي:

1. التحقق من صحة المدخلات: اختبار قدرة المكون على التعامل مع أنواع مختلفة من المدخلات، بما في ذلك البيانات الصالحة وغير الصالحة. تحقق من أن المكون يتعامل بسلاسة مع الحالات الحدية ويوفر رسائل خطأ أو استثناءات مناسبة.
 2. التحقق من المخرجات: تأكد من أن المكون ينتج المخرجات المتوقعة لمجموعة معينة من المدخلات. قارن المخرجات الفعلية مع النتائج المتوقعة لضمان الصحة.
 3. معالجة الأخطاء: اختبار آليات معالجة الأخطاء في المكون من خلال محاكاة سيناريوهات خطأ مختلفة، مثل المدخلات غير الصالحة، أو عدم توفر الموارد، أو الاستثناءات غير المتوقعة. تحقق من أن المكون يلتقط ويعالج الأخطاء بشكل مناسب.
 4. الحالات الحدية: اختبار سلوك المكون في ظل الحالات الحدية، مثل المدخلات الفارغة، أو الحد الأقصى لحجم المدخلات، أو القيم المتطرفة. تأكد من أن المكون يتعامل مع هذه الحالات بسلاسة دون أن يتعطل أو ينتج نتائج غير صحيحة.
- فيما يلي مثال على اختبار وحدة لمكون سير العمل في Ruby باستخدام إطار اختبار RSpec:

```

1  RSpec.describe OrderValidator do
2    describe '#validate' do
3      context 'valid is order when' do
4        let(:order) { build(:order) }
5
6        it 'true returns' do
7          expect(subject.validate(order)).to be true
8        end
9      end
10
11     context 'invalid is order when' do
12       let(:order) { build(:order, total_amount: -100) }
13
14       it 'false returns' do
15         expect(subject.validate(order)).to be false
16       end
17     end
18   end
19 end

```

في هذا المثال، يتم اختبار مكون OrderValidator باستخدام حالتي اختبار: واحدة لطلب صحيح وأخرى لطلب غير صحيح. تتحقق حالات الاختبار من أن طريقة validate تُرجع القيمة المنطقية المتوقعة بناءً على صحة الطلب.

اختبار تكامل تفاعلات مسار العمل

يركز اختبار التكامل على التحقق من التفاعلات وتدفق البيانات بين مكونات مسار العمل المختلفة. وهو يضمن أن المكونات تعمل معاً بسلاسة وتنتج النتائج المتوقعة. عند إجراء اختبار التكامل لمسارات العمل الذكية، ضع في اعتبارك ما يلي:

1. **تفاعل المكونات:** اختبار الاتصال وتبادل البيانات بين مكونات مسار العمل. التحقق من أن مخرجات أحد المكونات يتم تمريرها بشكل صحيح كمدخلات إلى المكون التالي في مسار العمل.
2. **اتساق البيانات:** التأكد من أن البيانات تظل متسقة ودقيقة أثناء تدفقها عبر مسار العمل. التحقق من أن تحويلات البيانات والحسابات والتجميعات تتم بشكل صحيح.
3. **انتشار الاستثناءات:** اختبار كيفية انتشار الاستثناءات والأخطاء ومعالجتها عبر مكونات مسار العمل. التحقق من أن الاستثناءات يتم التقاطها وتسجيلها ومعالجتها بشكل مناسب لمنع تعطل مسار العمل.

4. السلوك غير المتزامن: إذا كان مسار العمل يتضمن مكونات غير متزامنة أو تنفيذاً متوازيًا، اختبر آليات التنسيق والتزامن. تأكد من أن مسار العمل يعمل بشكل صحيح في السيناريوهات المتزامنة وغير المتزامنة.

فيما يلي مثال على اختبار تكامل لمسار عمل في Ruby باستخدام إطار الاختبار RSpec:

```

1 RSpec.describe OrderProcessingWorkflow do
2
3   let(:order) { build(:order) }
4
5   it 'successfully order the processes' do
6     expect(Validator).to receive(:validate).and_return(true)
7     expect(InventoryManager).to receive(:check_availability).and_return(true)
8     expect(PaymentProcessor).to receive(:process_payment).and_return(true)
9     expect(ShippingService).to receive(:schedule_shipping).and_return(true)
10
11     workflow = OrderProcessingWorkflow.new(order)
12     result = workflow.process
13
14     expect(result).to be true
15     expect(order.status).to eq('processed')
16   end
17
18 end

```

في هذا المثال، يتم اختبار OrderProcessingWorkflow من خلال التحقق من التفاعلات بين مكونات سير العمل المختلفة. تقوم حالة الاختبار بإعداد التوقعات لسلوك كل مكون وتضمن أن سير العمل يعالج الطلب بنجاح، مع تحديث حالة الطلب وفقاً لذلك.

اختبار نقاط القرار في الذكاء الاصطناعي

يعد اختبار نقاط القرار في الذكاء الاصطناعي أمراً حاسماً لضمان دقة وموثوقية مسارات العمل المدعومة بالذكاء الاصطناعي. عند اختبار نقاط القرار في الذكاء الاصطناعي، ضع في اعتبارك ما يلي:

1. دقة القرار: تحقق من أن مكون الذكاء الاصطناعي يتخذ قرارات دقيقة بناءً على البيانات المدخلة والنموذج المدرب. قارن قرارات الذكاء الاصطناعي مع النتائج المتوقعة أو البيانات المرجعية.

2. الحالات الحدية: اختبر سلوك مكون الذكاء الاصطناعي في الحالات الحدية والسيناريوهات غير المعتادة. تحقق من أن مكون الذكاء الاصطناعي يتعامل مع هذه الحالات بسلاسة ويتخذ قرارات منطقية.

3. التحيز والإنصاف: قيم مكون الذكاء الاصطناعي للكشف عن التحيزات المحتملة وتأكد من أنه يتخذ قرارات عادلة وغير متحيزة. اختبر المكون باستخدام بيانات إدخال متنوعة وحلل النتائج بحثاً عن أي أنماط تمييزية.

4. القابلية للتفسير: إذا كان مكون الذكاء الاصطناعي يقدم تفسيرات أو تبريرات لقراراته، تحقق من صحة ووضوح هذه التفسيرات. تأكد من أن التفسيرات تتوافق مع عملية اتخاذ القرار الأساسية.

فيما يلي مثال على اختبار نقطة قرار في الذكاء الاصطناعي باستخدام إطار اختبار RSpec في Ruby:

```

1 RSpec.describe FraudDetector do
2   describe '#detect_fraud' do
3     context 'fraudulent is transaction when' do
4       let(:tx) do
5         build(:transaction, amount: 10_000, location: 'Country High-Risk')
6       end
7
8       it 'true returns' do
9         expect(subject.detect_fraud(tx)).to be true
10      end
11    end
12
13    context 'legitimate is transaction when' do
14      let(:tx) do
15        build(:transaction, amount: 100, location: 'Country Low-Risk')
16      end
17
18      it 'false returns' do
19        expect(subject.detect_fraud(tx)).to be false
20      end
21    end
22  end
23 end

```

في هذا المثال، يتم اختبار مكون الذكاء الاصطناعي FraudDetector باستخدام حالي اختبار: واحدة لمعاملة احتيالية وأخرى لمعاملة شرعية. تتحقق حالات الاختبار من أن طريقة fraud_detect تُرجع القيمة المنطقية المتوقعة بناءً على خصائص المعاملة.

الاختبار من البداية إلى النهاية

يتضمن الاختبار من البداية إلى النهاية اختبار سير العمل بأكمله من البداية إلى النهاية، مع محاكاة سيناريوهات العالم الحقيقي وتفاعلات المستخدم. يتضمن ذلك أن سير العمل يتصرف بشكل صحيح ويُنتج النتائج المرجوة. عند إجراء الاختبار من البداية إلى النهاية لسير العمل الذكي، ضع في اعتبارك ما يلي:

1. سيناريوهات المستخدم: حدد سيناريوهات المستخدم الشائعة واختبر سلوك سير العمل في هذه السيناريوهات. تحقق من أن سير العمل يتعامل مع مدخلات المستخدم بشكل صحيح، ويتخذ القرارات المناسبة، وينتج المخرجات المتوقعة.

2. التحقق من صحة البيانات: تأكد من أن سير العمل يتحقق من صحة مدخلات المستخدم ويعالجها لمنع تناقضات البيانات أو الثغرات الأمنية. اختبر سير العمل باستخدام أنواع مختلفة من البيانات المدخلة، بما في ذلك البيانات الصالحة وغير الصالحة.

3. التعافي من الأخطاء: اختبر قدرة سير العمل على التعافي من الأخطاء والاستثناءات. قم بمحاكاة سيناريوهات الخطأ وتحقق من أن سير العمل يتعامل معها بشكل سلس، ويسجل الأخطاء، ويتخذ إجراءات التعافي المناسبة.

4. الأداء وقابلية التوسع: قيم أداء سير العمل وقابليته للتوسع تحت ظروف تحميل مختلفة. اختبر سير العمل مع حجم كبير من الطلبات المتزامنة وقيم بقياس أوقات الاستجابة، واستخدام الموارد، واستقرار النظام بشكل عام.

فيما يلي مثال على اختبار من البداية إلى النهاية لسير العمل في Ruby باستخدام إطار اختبار RSpec ومكتبة Capybara لمحاكاة تفاعلات المستخدم:

```

1 RSpec.describe 'Workflow Processing Order' do
2   scenario 'successfully order an places User' do
3     visit '/orders/new'
4     fill_in 'Product', with: 'Product Sample'
5     fill_in 'Quantity', with: '2'
6     fill_in 'Address Shipping', with: 'St Main 123'
7     click_button 'Order Place'
8
9     expect(page).to have_content('Successfully Placed Order')
10    expect(Order.count).to eq(1)
11    expect(Order.last.status).to eq('processed')
12  end
13 end

```

في هذا المثال، يحاكي الاختبار من البداية إلى النهاية مستخدماً يقوم بتقديم طلب من خلال واجهة الويب. يقوم بملء حقول النموذج المطلوبة، وإرسال الطلب، والتحقق من معالجة الطلب بنجاح، وعرض رسالة التأكيد المناسبة وتحديث حالة الطلب في قاعدة البيانات.

التكامل والنشر المستمر

لضمان موثوقية وقابلية صيانة مسارات العمل الذكية، يُوصى بدمج الاختبار والتحقق في خط أنابيب التكامل والنشر المستمر (CI/CD). يتيح ذلك الاختبار والتحقق الآلي من تغييرات مسار العمل قبل نشرها في بيئة الإنتاج. ضع في اعتبارك الممارسات التالية:

1. التنفيذ الآلي للاختبارات: قم بتكوين خط أنابيب CI/CD لتشغيل مجموعة الاختبارات تلقائياً عند إجراء تغييرات على الشفرة البرمجية لمسار العمل. يضمن ذلك اكتشاف أي انحدار أو فشل في وقت مبكر من عملية التطوير.

2. مراقبة تغطية الاختبارات: قم بقياس ومراقبة تغطية الاختبارات لمكونات مسار العمل ونقاط قرار الذكاء الاصطناعي. اهدف إلى تغطية اختبار عالية لضمان اختبار المسارات والسيناريوهات الحرجة بشكل شامل.

3. التغذية الراجعة المستمرة: قم بدمج نتائج الاختبارات ومقاييس جودة الشفرة في مسار عمل التطوير. قدم تغذية راجعة مستمرة للمطورين حول حالة الاختبارات، وجودة الشفرة، وأي مشكلات يتم اكتشافها أثناء عملية CI/CD.

4. **بيئات التجريب:** قم بنشر مسار العمل في بيئات تجريب تحاكي بيئة الإنتاج عن قرب. قم بإجراء اختبارات وتحقق إضافي في بيئة التجريب لاكتشاف أي مشكلات تتعلق بالبنية التحتية أو التكوين أو تكامل البيانات.

5. **آليات التراجع:** قم بتنفيذ آليات التراجع في حالة فشل النشر أو اكتشاف مشكلات حرجية في بيئة الإنتاج. تأكد من إمكانية إعادة مسار العمل بسرعة إلى إصدار مستقر سابق لتقليل وقت التعطل والتأثير على المستخدمين.

من خلال دمج الاختبار والتحقق في جميع مراحل دورة تطوير مسارات العمل الذكية، يمكن للمؤسسات ضمان موثوقية ودقة وقابلية صيانة أنظمتها المدعومة بالذكاء الاصطناعي. يساعد الاختبار والتحقق المنتظم في اكتشاف الأخطاء، ومنع الانحدار، وبناء الثقة في سلوك مسار العمل ونتائجه.

الجزء 2: الأنماط

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

هندسة المطالبات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

سلسلة التفكير

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف تعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

أمثلة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

توليد المحتوى

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

إنشاء الكيانات المهيكلية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

توجيه وكيل نموذج اللغة الكبير

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المزايا والاعتبارات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تبدیل الوضع

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

متى تستخدمها

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تعيين الدور

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

متى تستخدمه

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

أمثلة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كائن المطالبة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

قالب المطالبة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الفوائد والاعتبارات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

متى تستخدمه:

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المدخلات والمخرجات الهيكلية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تحجيم الإدخال والإخراج المنظم

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المزايا والاعتبارات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تسلسل المطالبات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

متى تستخدمه

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال: عملية تأهيل Olympia

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

معيد صياغة المحفزات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تسييج الاستجابة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المزايا والاعتبارات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

معالجة الأخطاء

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

محلل الاستعلامات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التنفيذ

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التوسيم النحوي (POS) والتعرف على الكيانات المسماة (NER)

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تصنيف النوايا

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

استخراج الكلمات المفتاحية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المزايا

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

معيد صياغة الاستعلامات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الفوائد

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

فن تحريك الدمى

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

متى تستخدمه

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المكونات المنفصلة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المسند

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

متى تستخدمه

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

واجهة وسيطة لا API

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الفوائد الرئيسية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

متى تستخدمها

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المصادقة والتفويض

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

معالجة الطلبات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تنسيق الاستجابات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

معالجة الأخطاء والحالات الحدية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

اعتبارات قابلية التوسع والأداء

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مقارنة مع أنماط التصميم الأخرى

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مُفسر النتائج

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

متى تستخدمه

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الآلة الافتراضية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف تعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

متى تستخدمها

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ما وراء السحر

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المواصفات والاختبار

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تحديد السلوك

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كتابة حالات الاختبار

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال: اختبار مكون المترجم

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

إعادة تشغيل تفاعلات HTTP

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التدخل البشري في العملية (HITL)

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الأنماط عالية المستوى

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الذكاء المجهين

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الاستجابة التكيفية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تبادل الأدوار بين الإنسان والذكاء الاصطناعي

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التصعيد

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الفوائد الرئيسية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التطبيق في العالم الواقعي: الرعاية الصحية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

حلقة التغذية الراجعة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف تعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التطبيقات والأمثلة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التقنيات المتقدمة في دمج التعليقات البشرية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

إشعاع المعلومات السليبي

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

عرض المعلومات السياقي

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الإشعارات الاستباقية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الرؤى التفسيرية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الاستكشاف التفاعلي

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الفوائد الرئيسية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التطبيقات والأمثلة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

صنع القرار التعاوني (CDM)

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التعلم المستمر

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

التطبيقات والأمثلة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

الاعتبارات الأخلاقية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

دور التدخل البشري في العملية في تخفيف مخاطر الذكاء الاصطناعي

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

التطورات التكنولوجية والنظرة المستقبلية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تحديات وقيود أنظمة التدخل البشري

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

معالجة الأخطاء الذكية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مناهج معالجة الأخطاء التقليدية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تشخيص الأخطاء السياقي

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

هندسة المحفزات لتشخيص الأخطاء السياقي

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التوليد المعزز بالاسترجاع لتشخيص الأخطاء السياقي

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الإبلاغ الذكي عن الأخطاء

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الوقاية التنبؤية من الأخطاء

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيفية العمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التعافي الذكي من الأخطاء

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيفية العمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التواصل المخصص بشأن الأخطاء

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

سير عمل معالجة الأخطاء التكيفي

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيفية العمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ضبط الجودة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

التقييم

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

المشكلة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

الحل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

كيف يعمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

اعتبارات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>.

فهم المراجع الذهبية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيف تعمل التقييمات بدون مرجعية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الضوابط الوقائية

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

المشكلة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

الحل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

كيفية العمل

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مثال

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

اعتبارات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

حواجز الحماية والتقييمات: وجهان لعملة واحدة

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

قابلية التبادل بين حواجز الحماية والتقييمات بدون مرجع

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

تنفيذ حواجز الحماية والتقييمات مزدوجة الغرض

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مسرد المصطلحات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

مسرد المصطلحات

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

أ

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ب

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ت

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

D

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

هـ

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

و

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ز

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

H

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

I

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

J

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

K

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

L

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

M

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

N

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

O

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

P

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

Q

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

R

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

S

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

T

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

U

هذا المحتوى غير متوفر في الكّاب العينة. يمكن شراء الكّاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

V

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

و

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

ص

هذا المحتوى غير متوفر في الكتاب العينة. يمكن شراء الكتاب على Leanpub على <http://leanpub.com/patterns-of-application-development-using-ai-ar>

فهرس

- 92 persistence,
- 92 preparation,
- 208 Datadog,
- decision
- 84 capabilities, -making
- errors
- 213 ,90 handling,
- 121 Handling, Error Intelligent
- 91 architecture, event-driven
- 78 F#,
- 19 Facebook,
- 177 FitAI,
- 9 7B, Gemma
- 78 GitLab,
- 18 Google,
- 17 Gemini,
- 15 ,14 ,11 Pro, 5.1 Gemini
- Model), Language (Pathways PaLM
- 14
- 19 Pathways), اللغة (نموذج PaLM
- 11 T5,
- 19 AI, Cloud منصة
- 121 ,109 ,84 AI,
- 117 applications,
- 83 model,
- 14 Sam, Altman,
- 211 Services, Web Amazon
- 116 ,109 ,61 ,33 ,18 Anthropic,
- 90 logging, audit
- 19 ,11 BERT,
- 213 conditions, boundary
- 212 Brotli,
- 118 (CoT), Thought of Chain
- 44 ,24 ChatGPT,
- 64 ,35 ,7 Claude,
- 116 ,114 ,109 ,107 ,40 ,3 Claude
- 62 Opus, 3 Claude
- 14 v1, Claude
- 14 v2, Claude
- 18 (مزود النماذج اللغوية الكبيرة), Cohere
- 20 (مزود نماذج اللغة الكبيرة), Cohere
- 212 workflows, concurrent
- 84 Analysis, Sentiment Customer
- data

- 47 8x7B,
- 210 Relic, New
- 20 Ollama,
- 128 ,121 ,109 ,52 ,27 Olympia,
- 61 ,33 ,18 OpenAI,
- 210 ,128 ,23 ,22 OpenRouter,
- 213 verification, output
- 9 , (مزود) Perplexity
- 90 Manager, Process
- 212 Buffers, Protocol
- 91 systems, publish-subscribe
- 20 PyTorch,
- 9 70B, Qwen2
- 163 Rails,
- 79 (ROP), Programming Oriented Railway
- 192 Raix,
- 82 مكتبة,
- 217 ,215 ,213 RSpec,
- 217 ,138 ,79 ,78 Ruby,
- 199 ,192 ,94 Rails, on Ruby
- 19 Alex, Rudall,
- 78 Language), (Programming Rust
- 98 (لغة برمجية), Rust
- 210 Scout,
- 110 Stripe,
- 83 directive, system
- 19 T5,
- 21 Together.ai,
- 211 منصة السحابة,
- 52 واجهة برمجة التطبيقات,
- 14 ,11 GPT-3,
- ,88 ,52 ,40 ,35 ,25 ,17 ,14 ,11 ,5 GPT-4,
- 210 ,171 ,113 ,108 ,101 ,98
- 15 Paul, Graham,
- 91 GraphQL,
- 101 ,21 Groq,
- 212 gzip,
- 79 Honeybadger,
- 127 HTTP,
- input
- 213 validation,
- 212 orchestration, workflow intelligent
- 122 refinement, iterative
- 114 JSON,
- 111 Notation), Object (JavaScript JSON
- 107 JSON (جافا سكريبت أوبيجكت نوتيشن),
- 122 (LLM), Model Language Large
- 41 ,70B-2 Llama
- 9 70B, 3 Llama
- 9 8B, 3 Llama
- 212 MessagePack,
- 19 Meta,
- 20 Mistral,
- 9 7B,
- 172 ,14 Instruct, 7B
- Mixtral
- 9 8x22B,

- اختبار المستخدم والتغذية الراجعة, 164
 اختناقات, 189
 اختيار الأدوات الإجباري, 112
 اختيار الأدوات الديناميكي, 111
 استخدام الأدوات, 104, 105, 126
 استدعاء الأداة, 130
 استراتيجيات التراجع, 92
 استراتيجيات التقسيم والاستهداف, 162
 استراتيجيات تحفيزية, 179
 الأجهزة اللوحية, 183
 الأخطاء
 التعافي, 217
 المعالجة, 92
 المعدلات, 93
 معالجة, 121
 الأخلاق
 الآثار, 167
 الأداء
 التحسين, 112
 المشاكل, 211
 المفاضلات, 4
 تحسين, 164
 الأنماط التاريخية, 189
 الأنماط الرئيسية, 187
 الإجابة على الأسئلة المغلقة والمفتوحة, 43
 الإدخال
 المطالبات, 47
 الإدخال والإخراج المنظم, 172
 الإنتاجية, 22, 159
 الاتساق
 وقابلية التكرار, 112
 الاحتفاظ بالسجلات وتدويرها, 208
 212 ID, Universal
 134, 128, 90, 79 Wisper,
 78 Chad, Wooley,
 114 XML,
 41 Yi-34B,
 آلات المتجهات الداعمة, 102
 آليات إعادة المحاولة, 92
 آليات التراجع, 219
 أجهزة الكمبيوتر المكتبية, 183
 أحداث مرسل من الخادم (SSE), 127
 أخذ العينات النووي, 39
 أخذ العينات بالقيمة العليا-k, 39
 أخطاء بنائية, 111
 أسماء الدوال, 131
 أطر التطوير, 126
 ألباكا, 11
 أنظمة الإجابة على الأسئلة, 6
 أنماط تكامل المؤسسات, 88
 أوبن إيه آي, 3
 أولمبيا, 141
 إدارة المعرفة, 25
 إدارة حركة المرور, 26
 إضفاء الطابع الإنساني, 57
 إعادة الصياغة, 44
 إكمال فائق الأداء, 21
 إمكانية الوصول, 182, 183
 اتخاذ القرار
 حالات الاستخدام, 113
 اتصال الشبكة, 190
 اختبار التكامل, 214

- الاختبار من البداية إلى النهاية, 217, 218
 الاستدلال, 4
 الاستقرار التلقائي, 136
 الاكتشافات الطبية, 85
 الانحدار الخطي, 35
 البرمجة الوظيفية, 77
 البنية الموزعة, 209
 البيانات
 استرجاع البيانات, 92
 التحقق من صحة البيانات, 217
 التحليل, 27, 125
 التدفق, 93
 الخصوصية, 21, 181
 السلامة, 202
 خط المعالجة, 202
 مزامنة البيانات, 92
 مهام المعالجة, 106
 البيانات المتدفقة, 129
 البيانات المنظمة, 114
 البيانات ذاتية التصحيح, 139, 205
 التجارة الإلكترونية, 161, 186
 التجريب
 إطار, 162
 التجزئة إلى رموز, 10
 التجميعات, 98
 التحديات المفاهيمية والعملية, 167
 التحسين التكراري, 63
 التحفيز بالأمثلة القليلة, 52
 التحقق من التأمين, 85
 التحيز
 والإنصاف في الذكاء الاصطناعي, 216
 التخزين المؤقت, 210
 التخصيص, 21, 158, 183, 187
 النصوص المصغرة المخصصة, 173
 النماذج المخصصة, 168
 التدخل البشري في العملية, 151
 التدخل اليدوي, 191
 التدريب على التعليمات, 8
 النماذج المدربة على التعليمات, 40, 43
 التدقيق والامثال, 207
 التدويل, 163
 الترجمة, 13, 164
 الترميز الثنائي المتزاوج, (BPE) 12
 الترميز بنط العلامات, 59
 التسجيل التفصيلي, 208
 التسجيل المنظم, 208
 التصفية التعاونية, 77
 التصفية المعتمدة على المحتوى, 77
 التصنيف, 43, 101
 التصويت بالأغلبية, 98
 التطبيقات التعليمية, 25
 التطبيقات الحديثة, 187
 التعلم أحادي الخطوة, 50
 التعلم الصفري, 49, 50
 التعلم بالأمثلة القليلة, 51
 التعلم غير الموجه, 4
 التفاعلات على نمط لعب الأدوار, 5
 التكامل والنشر المستمر, (CI/CD) 218
 خط الأنايب, 218
 التكميم, 23
 التلخيص, 43
 التنبؤات, 4
 التنفيذ المتوازي, 209
 التوجيه الديناميكي للمهام, 188

- العناد, 22
 الفضاء الكامن, 33, 35
 القابلية للتفسير, 216
 القرار
 الأشجار, 186
 النقاط, 206
 القفل التشاؤمي, 92
 القفل التفاضلي, 92
 القواعد التجارية, 186
 القواميس, 111
 الكتابة الإبداعية, 27, 43
 الكشف التدريجي, 174
 الكفاءة, 187
 اللغة
 اكتشاف اللغة, 94
 المهام المتعلقة, 4
 النماذج, 35, 54, 60
 اللغة الطبيعية
 معالجة اللغات الطبيعية, 101
 معالجة اللغة الطبيعية, 85
 اللوفر, 35
 المحادثة
 الحلقة, 135
 النص, 133, 135
 المحتوى
 التصنيف, 21
 تصنيف المحتوى, 94
 المحتوى المنشأ من قبل المستخدم, 94
 المحول المدرب مسبقاً المولد (GPT), 56
 المحول المدرب مسبقاً التوليدي (GPT), 7
 المراقبة
 والتسجيل, 93
 التوجيهات
 الهندسة, 33
 قالب التوجيه, 172
 التوسع التلقائي, 211
 التوليد المعزز بالاسترجاع, 31, 38
 التوليد المعزز بالاسترجاع (RAG), 25, 66, 106
 التوليد عبر الوسائط المتعددة, 18
 الجبر الخطي, 35
 الحالات الحدية, 48
 الخدمات الخارجية أو واجهات برمجة التطبيقات, 107
 الدالة
 سجل الاستدعاءات, 132
 الذكاء الاصطناعي, 54, 61, 114, 128, 170, 176
 الأنظمة المركبة, 24, 27
 التطبيقات, 106, 126, 138
 المحادثة, 178
 المحادثي, 5, 25
 النموذج, 132, 133, 135
 نقاط القرار, 215
 نموذج, 74, 177
 الرموز, 5, 10
 الزئبق (العنصر), 36
 السلوك الحتمي, 48
 السياق
 اتخاذ القرارات السياقية, 189
 اقتراحات الحقول السياقية, 168
 التعزيز, 38
 توليد المحتوى السياقي, 157, 161, 162, 167
 نافذة, 189
 الشبكات العصبية, 3, 5
 الضبط الدقيق, 66
 العالمية, 26

- النماذج
- متعددة الوسائط, 16, 17
- النماذج الأساسية, 44
- النماذج الاحتمالية, 35
- النماذج البيانية, 35
- النماذج القائمة على الاسترجاع, 6
- النماذج اللغوية الكبيرة, 101, 157
- النماذج اللغوية الكبيرة (LLM), 1, 3, 166
- النمذجة ذاتية الانحدار, 35
- النمطية, 74
- النموذج اللغوي الكبير, 63, 122, 194
- النموذج اللغوي الكبير (LLM), 23, 59, 73, 105, 171, 124, 119
- الهواتف الذكية, 183
- الواجهات التي تعمل بالصوت, 27
- الواجهات الشاملة, 167
- برنامج المحاكاة الصوتية, 149
- بناء السرد, 16
- بنية المحولات, 5
- بيئات التجريب, 219
- بيئات التطوير المحلية, 131
- بيانات التدريب, 34
- تتبع المقاييس الرئيسية, 205
- تجار التجزئة عبر الإنترنت, 172
- تجربة المستخدم, 163
- تجميع المستندات, 101
- تحديد المواضيع, 101
- تحسين الأداء, 207
- تحليل المشاعر, 13, 84, 94-96, 99, 114, 123
- تخصيص ديريكلية الكامن, 102
- تخطيط الاستجابة للطوارئ, 26
- تدوين كائنات جافا سكريبت (JSON), 125
- والتنبه, 190
- المراقبة المستمرة للمخاطر, 87
- المراقبة والتسجيل, 207
- المرونة والإبداع, 164
- المساعدون الافتراضيون, 27
- المشهد الرقمي, 162
- المصفوفات, 111
- المصنفات الترتيبية, 28
- المطالبات
- التحسين, 57
- التصميم, 48, 57
- الهندسة, 37, 47, 56
- تقطير المطالبات, 37
- تقطير المطالبة, 65
- المطالبات البرمجية
- الهندسة, 54
- المعالجة الدفعية, 210
- المعالجة غير المتزامنة, 209
- المعامل الفائق, 38
- المعاملات
- التأثيرات, 109
- المعلومات
- استرجاع, 6
- الاستخراج, 43
- الاسترجاع, 106
- المهام المعقدة, 124
- الموجهات
- تسلسل, 59
- تقطير الموجهات, 60, 210
- كائن الموجه, 61
- الثرة العاطفية, 123
- النظام البيئي, 125

- خدمة البث المدار Kafka, Apache 34
- خصائص ACID, 92
- خوارزمية كي-مينز, 102
- دالة
- استدعاء, 104, 105
- درجة الحرارة, 45
- دعم العملاء, 25
- دعم القرارات السريعة, 87
- دمج النماذج اللغوية الكبيرة, 158
- دوهان وآخرون, 35
- ربط العمال الاصطناعيين, 93
- رسالة التشغيل, 88
- روبوتات دردشة خدمة العملاء, 27
- روبي, 95
- روبي أون ريلز, 1
- زمن الاستجابة, 22
- زمن الرمز الأول, (TTFT) 22
- سطر الأوامر
- واجهة سطر الأوامر, 20
- سلسلة التفكير, 37
- سلسلة التوريد
- التحسين, 26
- سياق
- مدخلات لا نهائية الطول, 13
- نافذة, 12
- سير العمل التكيفي
- تكوين سير العمل التكيفي, 189
- سير عمل متعدد الخطوات, 94
- طريقة finalize, 134, 135
- طريقة الإنهاء, 132
- عديم الحالة, 133
- عطار (الكوكب), 36
- ترميز أزواج البايت, (BPE) 10
- تسييج الاستجابة, 149, 172
- تصحيح الأخطاء, 189
- والاختبار, 112
- تصحيح الأخطاء واستكشاف المشكلات, 207
- تصميم التطبيقات وأطر العمل, 166
- تصنيف المخاطر, 86
- تضييق المسار, 31, 32
- تطبيق روبوت المحادثة, 100
- تطبيقات التجارة الإلكترونية, 77
- تطوير التطبيقات, 185
- تعدد العمال, 141
- تعددية العمال, 100
- تعيين التذاكر, 201
- تغذية راجعة
- حلقة التغذية الراجعة, 49
- تقييم الأعراض وتصنيفها, 85
- تنسيق سير العمل الذكي, 185, 192, 210
- تنظيف النص, 94
- توجيه النظام, 109
- توصيات المنتجات, 77
- توصيات المنتجات المخصصة, 77
- توليد البيانات الاصطناعية, 44
- توليد واجهة المستخدم الديناميكية, 158
- ثقة المستخدم, 182
- جدول التجزئة, 129
- جمع التاريخ الطبي, 85
- جوجل
- واجهة برمجة التطبيقات, 54
- جيسون, (JSON) 141
- حساب, 76
- حقن SQL, 58

- التكامل المؤسسي, 192
 مراقب المحتوى الذكي, 195
 مركز ميموريال سلون كيترينج للسرطان, 34
 مشاكل قابلية الاستخدام, 182
 مطابقة الأنماط, 129
 مطالبات
 تسلسل, 49
 قالب المطالبة, 49
 هندسة, 49
 معالجات التدفق, 128
 معالجة الاستثناءات, 190, 191
 معالجة التدفق, 127, 132, 138
 المنطق, 134
 معامل
 نطاق, 9
 معاملات
 عدد المعاملات, 22
 معاملات الإدخال, 109
 مفسر النتائج, 120
 مقاييس المراقبة, 208
 مقدمي خدمات استضافة النماذج مفتوحة المصدر, 172
 مكتبة, 217 Capybara
 منطق قاطع الدائرة, 137
 موجّهات
 هندسة, 180
 موظفو, 43 Databricks
 ميكوروي (الإله الروماني), 36
 نظارات الواقع المعزز, 183
 نظرية العقل, 33
 نماذج اللغة الكبيرة, 139, 141, 176
 نموذج, 19 OPT
 عقوبات التكرار, 42
 عقوبة الحضور, 39
 علم نفس المستخدم, 181
 علوم الحاسوب, 58, 60
 عملية التقطير, 63
 عوامل المخاطر, 80
 عوامل المخاطرة, 81
 فشل استدعاء الدالة, 113
 قابلية التوسع, 187, 209
 قاعدة معرفة, 76 Olympia
 قفل المفسر العالمي, (GIL), 97
 قواعد البيانات, 104
 استراتيجيات القفل, 92
 كائن مدعوم, 89
 قواعد المعرفة, 6
 قواعد النحو, 3
 كشف الاحتيال
 نظام, 82
 لاما, 11
 لغة C (لغة برمجة), 98
 لغة قابلة للترميز باليونيكود, 12
 مأساة المشاعات, 160
 ماركداون, 124
 مبدأ الامتياز الأدنى, 59
 متحف متروبوليتان للفنون, 35
 متعدد العوامل
 حلل المشكلات, 25
 مجموعات, 99
 مجموعة العمال, 99
 مجموعة, 93 ELK
 مدير العمليات, 88

- نموذج اللغة الكبير, 114
المشهد, 22
نموذج اللغة الكبير, (LLM), 13, 14, 56, 57, 64,
104, 93
نموذج بايز الساذج, 102
هندسة البرمجيات, 2
هندسة الخدمات المصغرة, 75
هندسة تطبيقات المؤسسات, 31
هوبه، غريغور, 88
واجهات المستخدم التوليدية, 183
واجهات برمجة التطبيقات, 59, 104, 130
واجهة المستخدم
أطر العمل, 180
التصميم, 183
تقنيات, 176
واجهات, 180
واجهة المستخدم (UI)
الواجهات, 166
واجهة المستخدم التكيفية, 175
واجهة المستخدم التوليدية, 176, 179
واجهة المستخدم التوليدية, (GenUI), 166, 172,
173
واجهة مرئية, 176
وقت المعالجة, 93
وول، لاري, 2