המאמר היומי של עמרי ומייק 27.06.2025

Agent-as-a-Judge: Evaluate Agents with Agents

כולנו כבר מכירים את הקונספט LLM-as-a-Judge שזה אומר להיעזר במודלי שפה גדולים כדי לבחון מודלי שפה אחרים. צוות מ-Meta מציג כאן חלופה שאפתנית יותר: (Agent-as-a-Judge (AAJ, תפיסה שבה סוכן מבצע אבלואציה לסוכנים אחרים ומספק משוב עשיר ברמת הצעד, לא רק פסק דין סופי.

אחת התרומות המרכזיות של המאמר היא DevAI שזה דאטהסט שהמחברים בנו מאפס: 55 משימות פיתוח AI מורכבות יחסית מקצה-לקצה, שמפורקות לתת דרישות עבור כל הערכה של משימה, סה״כ 365 דרישות. DevAI נולד כתגובה לפער בבנצ’מרקים קיימים, שרובם מסתפקים במדד “עבר/נכשל” סופי; הפירוק היסודי לדרישות־משנה נועד לחשוף את הבאגים והכשלים שמתרחשים באמצע תהליך הפיתוח. זה החלק שבו סוכנים נכשלים הכי הרבה, אך כמעט שלא נמדד עד היום כפי שהם טוענים.

שלושה “סוכני-מתכנת” פופולריים, MetaGPT, GPT-Pilot ו-OpenHands, (נכון לאוקטובר 2024) קיבלו לפתור את כל המשימות. כאן נכנס AAJ: הוא עצמו סוכן עם חמישה כלים או modules כמו שהם קוראים לזה במאמר ask, graph, read, locate, retrieve.

graph – יוצר גרף תלות בין קבצים ופונקציות וכך מבין אילו רכיבים משפיעים זה על זה.

read – קורא ומנתח את תוכן הקבצים והלוגים כדי לבדוק אם המימוש עומד בדרישות.

locate – מאתר במדויק את שורות הקוד או השגיאות הרלוונטיות שמסבירות את הכשל או ההצלחה.

retrieve – שולף קטעים רלוונטיים ממסלולי הרצה ארוכים כדי לגבות את ההחלטה במקרים של חוסר ודאות(כמן ב-RAG).

ask – מקבל את ההחלטה הסופית האם הסוכן עמד בדרישה או לא ומספק נימוק קצר.

הכישורים האלה מאפשרים לו לחטט בקוד שיוצר הסוכן הנבחן, לבנות גרף תלות (dependency graph) בין הקבצים, לאתר שורות שגיאה ולשלוף תיעוד רלוונטי, ורק אז לקבוע אם הדרישה הושלמה. כדי שתהיה אמת-מידה אנושית, שלושה מתייגים מומחים דירגו כל דרישה בנפרד, אחר כך עשו majority-vote ולבסוף דיון שהוליד קונצנזוס לגבי על דרישה בנפרד. הסכמה ראשונית נעה סביב 70%–90%; אחרי דיון היא התייצבה על בערך 95% - זהו ה-ground truth שמולו מודדים את כל השופטים.

AAJ נבדק בשתי רמות מידע:

Black-box: שופט עיוור, רואה רק את הקלט והפלט. מדמה-מציאות המחמיר.

Gray-box: מקבל גם את הלוגים וקבצי הקוד, כך שהשיפוט קל ומבוסס יותר.

ב-Black-box AAJ משתווה כמעט במדויק למתייג אנושי ממוצע, בעוד ש-LLM-as-a-Judge נשאר מאחור בפער ניכר. במצב ה-Gray-box AAJ מתקרב עוד יותר להסכמה האנושית ומצמצם את המרחק עד לכדי אחוזים בודדים, כך שהוא כבר טוב כמעט כמו מתייג אנושי כל זה בזמן ש LLM-as-a-Judge רחוק מהמתייג האנושי .

כאשר מסתכלים על עלות וזמן התמונה חדה: אבלואציה ידנית גובה אלפי דולרים ונמשכת ימים, בעוד LLM-as-a-Judge מבצע זאת בדקות ספורות ובעלות מזערית אך מקריב לא מעט דיוק. AAJ מוסיף רק קמצוץ של זמן וכסף ביחס ל-LLM, ומחזיר כמעט במלואו את רמת האיכות של האבולוציה האנושית. ניתוח האבלואציה מראה שככל שמוסיפים ל-AAJ את היכולות graph, read ו-locate, הוא מתקרב יותר ויותר לרמה של בני אדם.

המחברים מבהירים שההדגמות עדיין מוגבלות לעולמות ג'ינרוט קוד, ושמעבר לתחומים אחרים יצריך בדיקות נוספות. הם גם מציינים כי לסוכן-השופט יש שכבת זיכרון ותכנון מורכב וזה עלול להשתבש בקלות יחסית - המנגנון של האגנטים מאוד עדין ולכן שינוי בפרטים קטנים כגון בפרומפטים עלולים להוביל לפגיעה רצינית בו (במנגנון).

למרות זאת, הם מציעים לראות ב-AAJ בסיס לקו מחקר חדש של למידה חיזוקית מונחית-תהליך: במקום RLHF שמסתמך על התיוגים שמבוצעים על ידי בני אדם, אפשר לדמיין RLAF – Reinforcement Learning from משוב של מערכת האג'נטים. AAJ מאבחן את הטעויות, מזרים את הפידבק בחזרה, וסוגר את לופ השיפור בלי יד אדם.

לסיכום, גישת agents grading agents מוכיחה שאפשר להשיג רמת דיוק אנושית כמעט ללא עלות אנושית, תוך קבלת תובנות מפורטות בהרבה מהשיטה המסורתית של LLM-as-a-Judge. למי שמחפש פידבק תהליכי עשיר לצד חסכון בזמן ובכסף, Agent-as-a-Judge הוא צעד משמעותי קדימה.

https://arxiv.org/abs/2410.10934