המאמר היומי של עמרי ומייק: 16.08.25

Large Action Models: From Inception to Implementation

מה זה (Large Action Models (LAM ואיך זה שונה מ-LLM? שורה תחתונה: LAM הוא LLM, אבל כזה שאומן והותאם במיוחד כדי להפיק פעולות ברות־ביצוע בסביבה אמיתית. בעוד שמודל LLM רגיל מאומן להפיק טקסט איכותי ועקבי, LAM מאומן לייצר תוכניות ופקודות שניתן להפעיל בפועל דרך agent, בין אם זה קליק, הקלדה או קריאת API, כך שהוא משפיע ישירות על מצב העולם ולא רק “מדבר עליו”.

מה שהכותבים מציעים הוא שבמקום לחבר LLMים לסביבת agentים, יש לחבר LAM שלמעשה משמש כמנוע קבלת ההחלטות בתוך הלולאה של ה־agent: ה־agent אוסף תצפיות מהסביבה (למשל מצב מסך, רשימת כפתורים זמינים או נתוני API), מזין אותן ל־LAM, וה־LAM מחזיר את הפעולה הבאה לביצוע. ה־agent הוא זה שמבצע בפועל את הפעולה ומחזיר חיווי על התוצאה וזה מה שמאפשר ל־LAM לעדכן את ההחלטות הבאות.

כאן בדיוק טמון ההבדל הקריטי גם ברמת הסיכון, כפי שהכותבים רואים זאת: טעות של LLM “קלאסי” מתבטאת לרוב בתשובה שגויה או בהזיה (hallucination) - פגיעה בהבנה או באמון, אך בלי השלכות ישירות בעולם האמיתי. לעומת זאת, טעות של LAM עלולה לגרום לשינוי ממשי: מחיקת קובץ חשוב, שליחת הודעה לכתובת הלא נכונה, או ביצוע פעולה עסקית לא רצויה.

האינטראקציה עם הסביבה שבה החוקרים פעלו נעשתה ב־Windows בלבד, במשימות ממוקדות ב־Microsoft Word. הם חיברו את ה־LAM אל UFO, סוכן GUI ייעודי ל־Windows. הסוכן קורא את מצב הממשק (status) שזה רשימת הבקרים (Controls) עם סוג, כותרת ואינדקס ומעביר את המידע ל־LAM להכרעה, ולאחר מכן מבצע את הפעולה (action) שנבחרה: לחיצת עכבר, הקלדה, או קריאת API.

התהליך שהחוקרים מציעים בנוי מ-5 שלבים: - Data → Training → Integration & Grounding → Offline Eval → Online Eval. לאורך המאמר ישנה הפרדה בין Task-Planning לבין Task-Action: בשלב איסוף ה־data אוספים קודם Task→Plan, ולאחר מכן הם יוצרים מסלולים (trajectories) שהופכים את הצעדים האלו לפעולות קונקרטיות בסביבת Word: בחירת כפתור ספציפי, הגדרת סוג פעולה ופרמטרים כך שה־agent יכול להריץ אותם בפועל ולבחון הצלחה או כישלון. לתהליך הזה הם קוראים Grounding: עיגון הפלט הטקסטואלי של המודל ל-UI אמיתי ולפעולה אופרטיבית דטרמיניסטית.

ב־LAM1 המודל אומן ב־SFT על Task→Plan בלבד (𝑡ᵢ→𝑃ᵢ). הכותבים מסבירים שהאינטואיציה כאן היא ללמד קודם את המודל לפרק משימות בצורה הגיונית ומסודרת לפני שניגשים לבחירת פעולות בפועל. לשם כך השתמשו בכ־76.7K דוגמאות ממקורות כמו מדריכי עזרה, WikiHow ושאילתות היסטוריות, שעברו ניקוי, עיבוד והבשלה כדי להבטיח עקביות ואיכות.

ב־LAM2 המיקוד עבר ל־State→Action, חיקוי מסלולי הצלחה של (GPT-4o (𝑠ₜ→𝑎ₜ. כאן כל דוגמה מייצגת מצב נוכחי (UI state) כפי שנקלט על־ידי ה־agent שזה רשימת בקרים (Controls) עם סוג, כותרת ואינדקס בצירוף טקסט המשימה, והפעולה המדויקת שבוצעה בפועל: בחירת הבקר הנכון, סוג הפעולה והפרמטרים. את מסלולי ההצלחה יצרו מתוך מאגר ה־Task→Plan של LAM1, תוך הפיכת הצעדים הכלליים לפקודות ממוקדות על רכיבים אמיתיים ב־Word, הרצה ובדיקה בסביבה החיה, וסינון לפי הצלחה בפועל. גם שלב זה אומן ב־SFT, כשהדאהטסט הכיל בסופו של דבר 2,192 מסלולים מוצלחים (trajectories) ששימשו כבסיס לאימון.

ב־LAM3 המשיכו ב־SFT על (State→Action (𝑠ₜ→𝑎ₜ, אך בשלב זה יישמו Self-Boosting: לקחו מסלולי כישלון של GPT-4o, נתנו למודל שנאמן ב־LAM2 לנסות שוב, ואספו את ההצלחות החדשות שיצר. כך נוצר דאטה נוסף ואיכותי ללא אנוטציה ידנית, שהרחיב את כיסוי המודל גם על מקרים קשים יותר.

ב־LAM4 עברו משלב ה־SFT ל־RL, וביצעו Offline PPO המונחה על־ידי Reward Model. את ה־Reward Model בנו על בסיס LAM3, בתוספת שכבה שמחזירה ציון הצלחה לכל פעולה, כשהמודל אומן ב־LoRA על מסלולי הצלחה וכישלון. לצורך האימון, כל צעד במסלול מוצלח קיבל ציון +1 וכל צעד במסלול כושל קיבל ציון −1, וה־RM אומן עם MSE כדי לחזות את הציון הזה.

עם RM מוכן, השתמשו בו כדי לאמן את LAM4 ב־Offline PPO, כשההתמקדות הייתה דווקא על 1,788 מסלולי הכישלון שנאספו ב־LAM3 – במטרה “ללמוד מהטעויות”. כאן הפורמט הוא (𝑠ₜ, 𝑟ₜ)→𝑎ₜ, כאשר ה־RM מספק את ה־𝑟ₜ, והמודל לומד לשפר את בחירת הפעולות מעבר למה שנלמד בחיקוי ישיר.

לאורך המאמר מוצגות שלוש מדידות: תכנון (Planning), פעולות אופליין (Offline Eval) והרצות חיות (Online Eval). בשני הראשונים נבדקו הצלחות ברמת תכנון המשימה והצעדים, וכן דיוקים בבחירת אובייקט ופעולה, והמודלים התקדמו בהדרגה מרמה תחרותית ועד שיפורים עקביים. בשלב השלישי – ההרצות בסביבת Windows ו־Word – נמצא כי LAM טקסטואלי בלבד היה תחרותי מול GPT-4o, ואף עקף אותו בחלק מהמדדים כשהשוו קונפיגורציות טקסטואליות בלבד. לעומת זאת, כאשר ל־GPT-4o נוספה גם יכולת vision, שיעורי ההצלחה היו גבוהים יותר, אך המחיר היה ירידה במהירות וביעילות.

אנו מניחים שככל ש־agents יהפכו ליותר נפוצים ובעלי יכולות, נראה עוד ועוד עבודות בסגנון הזה – כאלה שמחברות מודלים לסביבות אמיתיות ומבצעות אימון עם דאטה ייעודי ואדפטציה למשימות, לא בטוח שאימון LAM בשלושה שלבי SFT ואחריהם שלב RL יחיד הוא המתכון האופטימלי, אבל הכיוון של להפוך LLM ים ליותר ממוקדי־משימה, עם אימון מובנה ומותאם־דומיין, הוא צעד מתבקש בעידן שבו יותר ויותר agents יפעלו בעולם האמיתי.

https://arxiv.org/pdf/2412.10047

לאלף את החיה: הטרנספורמרים סוף סוף תחת שליטה מתמטית.