המאמר היומי של עדן ומייק 23.08.25  
MemOS: An Operating System for Memory-Augmented Generation (MAG) in Large Language Models

## מבוא:

מודלי שפה גדולים (LLMs) נמצאים היום בחזית של מספר רב של תחומים ומגיעים לביצועים חזקים במשימות שונות כגון קידוד, מענה על שאלות מדעיות ועוד. כותבי המאמר טוענים שהעתיד של מודלים אלו הוא ההפיכה שלהם לכלי ששומר על מצב (state) ויודע להפעיל היגיון לאורך זמן ממושך בחיי הסשן שבו הוא פועל. מצב יכול להיות אינטרקציות עבר הכוללות העדפות משתמש, ביצוע משימות ועוד. כפועל יוצא מכך, עולה הבעיה המרכזית עליה כותבי המאמר דנים - כיצד ניתן לשמור את כל המידע (מצב) הזה כך שניתן יהיה לחפש בו ולשחזר בקלות ממנו?

הם טוענים שזיכרון להיות למשהו נחוץ כדי לענות על הצורך הזה. הזיכרון יהפוך את מודל השפה לאחד שיוכל בצורה עקבית לשמור על הזהות וההתנהגות המצופה ממנו לאורך ציר הזמן. הזיכרון צריך לדעת לעמוד בקצבים גבוהים של העברה ואחסון מידע, להיות יעיל ודינמי בהתאם לשינויים שמתרחשים לאורך ציר הזמן. כל אלו גרמו להם לחשוב שזיכרון דמוי זיכרון של מערכת הפעלה הוא המתאים למשימה. זיכרון זה מכיל זיכרון קצר טווח, זיכרון ארוך טווח (שמור על הדיסק) ומאפשר דברים נוספים כמו ביקורת על גישה לזיכרון, תיוג של מי ביקש מה ועוד. לטענת הכותבים, הסוכנים העתידיים שישתמשו בזיכרון זה יוכלו בעצמם להחליט מתי לגשת לזיכרון כדיל לשוף מידע, מתי לסכם פיסות מידע לחוקים בשביל שליפה מהירה ועוד.

## המצב כיום:

כיום הזיכרון של LLMs הוא זיכרון פרמטרי בו כל הידע מקודד לתוך הפרמטרים של המודל (=משקולות). הבעיה שזיכרון זה הוא סטטי ולכן מצריך אימון מחדש (fine tune) כדי להקנות את המודלים ידע חדש, ותהליך עלול להיות יקר ולא יציב. הפתרון שנהוג היום לכך הוא שימוש ב- (Retrieval Augmented Generation (RAG. הוא מאפשר למודלים לגשת למידע עדכני בזמן ריצה מבלי שצריך לאמן אותם מחדש. השיטה אומרת לדחוף את המידע לחלון הקונטקסט של המודל כחלק מהפרומפט.

לטענת הכותבים RAG אינו תחליף לזיכרון שכן הוא לא מתייחס למידע כמשהו שמשתנה עם הזמן. לכן לא יכול לשמש כזיכרון ארוך טווח אלא כפתרון לזיכרון קצר טווח ולכן המודלים מתקשים לזכור מידע בתחילת השיחה כאשר מדובר בשיחה ארוכה. הם מראים ארבעה סוגים של קונטקסט שמודלים מתקשים בו:

מידול תלויות ארוכות טווח: כאשר המודל צריך לעקוב אחרי סגנון הכתיבה של המשתמש. כאשר מדובר בקונקסט גדול של כל השיחה המודל מהר מאוד ישכח את הסגנון ויחזור לכתוב בצורה רגילה.

גמישות למידע משתנה: מידע בעולם האמיתי מתפתח ומשתנה (עדכון מסמכים של החברה, עדכון קוד בגיט ועוד). RAG לא מאפשר שמירה של ציר הזמן בו ניתן לעקוב אחר שינויים אלו דבר שעלול לגרום למצב הוא המידע שהוא מביא למודל כבר לא עדכני.

תמיכה בתפקידים שונים (multi roles): ל-LLM אין אפשרות של לשמירת זיכרון עבור משתמשים שונים, תפקידים שונים ומשימות שונות. כל אינטראקציה או סשן היא דף חלק המתעלם מהעבר. לטענת הכתובים הזיכרון שמציעים כלים כמו ChatGPT הוא נאיבי ולא מאפשר שליטה מבנית במידע.

מעבר בין מערכות (Migration): זיכרון צריך לדעת לעבור ממערכת למערכת וזה לא המצב כיום. למשל זיכרון שיחה ב-ChatGPT לא יוכל לעבור ל-Claude. מעבר של זיכרון ממערכת למערכת היא דרישת בסיס בזיכרון בר קיימא.

## פתרון:

החוקרים מציעים את MemOS, מערכת המדמה זיכרון של מערכת הפעלה עבור מודלי שפה. תפיסה זו מציעה שלושה יתרונות מרכזיים:

שליטה: המערכת מאפשרת תזמון של יצירת זיכרון חדש, עדכון, שילוב זכרונות לזיכרון אחד ומחיקה. בנוסף נותנת שקיפות לשליטה עם הגבלת הגישה (access control) רק למי שיש הרשאה ומתעדת פעולות (auditing).

גמישות: הזיכרון תומך במעבר בין משימות או מטרות בצורה נוחה כך שמודלים יכולים להחליף זיכרון בהתאם למשימה או לעדכן אותו בהתאם לשינויים שהתרחשו.

התפתחות: הזיכרון מאפשר מעבר בין סוגי זיכרון שונים בהתאם לצורך - זיכרון פרמטרי (הזיכרון של המודל) וזיכרון מובנה חיצוני. כך מתאפשר למשל דחיסה של מידע ארוך טווח לתוך הפרמטרים של המודל עצמו.

הקו שמנחה אותם הוא שמערכת הפעלה מאפשרת אבסטרקציה של משאבים, תזמון אחיד ושליטה. כך לטענתם צריך לנהוג גם במודלי שפה בתפקיד התוכנה שמבקשת גישה למשאבים. לכן הם חילקו את המערכת לשלוש שכבות לפי אותם קויים ועשו מיפוי בכל שכבה בין רכיבים במערכת הפעלה רגילה באותה שכבה לבין רכיב במערכת שלהם.

למשל בשכבת הליבה (איפה שהזיכרון יושב) אפשר לראות את 3 סוגי הזיכרון. אם נסתכל למשל על הזיכרון הפרמטרי ,שהוא הזיכרון ארוך הטווח, נראה שבמערכת ההפעלה הינה הרכיב האחראי על כך הם הרגיסטרים במחשב בעוד ב-MemOS הרכיב האחראי על כך נקרא גם הוא זיכרון פרמטרי ומתייחס לפרמטרים של המודל. דוגמא נוספת היא הגישה למערכת. במערכת הפעלה רגילה הגישה למערכת ההפעלה נעשת באמצעות קריאות מערכת (system calls) וב-MemOS קיים גם ממשק (API) שעליו אחראי רכיב ה-MemoryAPI.

החוקרים לוקחים השראה מאימון מודלי שפה ומציינים שהיום כפי שאפשר לאמן את מודל השפה על מידע חדש או משימה חדשה, כך גם את הזיכרון אפשר לאמן. הם קוראים לאימון זה Mem-Training (MT). MT מאפשר לאסוף את כל פיסות הזיכרון, לארגן אותם מחדש ולשחזר אותם כל זה בזמן ריצה. הלמידה מתבצעת באמצעות אינטראקציות חוזרות של המערכת (דרך המודל) עם המשתמש או הסביבה והפיכה של אותן אינטראקציות לפיסות מידע שניתן לשחזר בקלות בהמשך בעת הצורך.

כפי שרואים מהטבלה, הארכיטקטורה של המערכת בנויה משלוש שכבות:

שכבת הממשק / ניהול: זוהי השכבה שבין המשתמש לזיכרון. היא מספקת את ה-Memory API. הממשק הזה תומך בקריאה, כתיבה, עדכון ותשאול של הזיכרון וחלקיו. MemReader מתגרם בקשות בשפחה חופשי לשאילתה מסוג MemQuery. דוגמא לכך היא השאילתא: “Summarize my meeting notes from last month”. הרכיב מחלץ את המידע הרלוונטי כגון טווח זמן (last month), סוג הזיכרון (meeting notes) והפלט (summary). שכבה זו גם בודקת הרשאות ביחד עם MemGov.

שכבת האופרציות (operation): אחראית על ניהול תהליכים ברקע. הרכיב הבסיסי בשכבה זו הוא ה-MemOperator הכולל פעולות שונות כגון בניית גרף הזיכרון, אחזור מהזיכרון הסמנטי ועוד. ה-MemSchedular מבצע אופטימיזציה ותזמון לתהליכים בהתאם לקונטקסט ולכוונה. הרכיב האחרון שפועל כאן הוא MemLifecycle שעוקב אחרי שינויים ומעברי מצבים של רכיבים, ליצירת שקיפות מלאה

שכבת התשתית (Infrastructure): מטפלת באבטחת מידע ואחסון. MemGovernance אוכף כללי גישה ושימוש במידע רגיש. MemVault מנהל תיקיון זיכרון (repositories) של זיכרונות כגון תיקייה של זיכרונות משתמש, זכרונות הקשורות לדומיין ועוד. MemStore מאפשר שיתוף זיכרונות החוצה בין סוכנים שונים.

היחידה הבסיסית של זיכרון ב-MemOS נקראת MemCube והיא אבסטרקציה למשאב זיכרון שנועד לייצג את כל סוגי הזיכרון במערכת. כל קובית זיכרון שכזה מכילה שני רכיבים: Memory Payload - התוכן הסמנטי עצמו ו-Metadata. המאטה דאטה הוא אחד מבין שלושה סוגים:

Descriptive Identifier: מכיל מידע כגון Timestamp שהיא חותמת הזמן האחרונה שבה נעשי שינוי באותו MemCube בזמן החיים שלו, Origin Structure - מאיפה הגיע המידע (למשל שאילתא של משתמש או תשובה של מודל וכו׳) ו-Semantic Type - למה נועד הזיכרון (למשל פרומפט, העדפות משתמש וכו׳).

Governance Attributes: מכיל מידע התורם לאבטחת המידע באמצעות הגדרת חוקים שונים כגון: Access Control - מי יכול לגשת לזיכרון,Traceability - מה הסיווג של הזיכרון, TTL - כמה זמן הזיכרון יכול להיות קיים.

Access Pattern - מתי וכמה ניגשו לקוביית הזיכרון הזו. זה נותן ל-MemOSt את היכולת לתת עדיפות ל-MemCube מסוימים.

מבנה הזיכרון ב-MemOS כולל שלושה סוגים של זיכרון אשר יוצרים היררכיה:

זיכרון טקסט חופשי (Plain Text): זיכרון זה הוא זיכרון חיצוני נפרד שנגיש בצורה דינמית. דוגמא למה שנשמר שם: פרומפטים, פסקאות ועוד. זיכרון זה מקושר לזיכרון האקטיבציה שהוא זיכרון בנגישות מהירה יותר ולכן זיכרון טקסט חופשי שנעשה בו שימוש הרבה, יכול להפוך לזיכרון אקטיבציה בשביל שחזור מהיר יותר (אפשר לחשוב על זה כמעבר מהדיסק ל-Cache). הדרך שבה הזיכרון נשמר היא גרף של task-concept-fact. זיכרון זה מתאים בעיקר עבור משימות של מספר סוכנים (multi agent), פרסונליות ודורשות הסתמכות על המון עובדות (facts heavy tasks).

זיכרון אקטיבציה (Activation): זיכרון ביניים המכיל את ה-hidden states שנוצרו בזמן תהליך ההסקה (inference). מנגנון ה-KV Cache הוא מרכיב מרכזי בזיכרון זה. MemOS יכול להשתמש ברכיב זה כדי להזריק את הזיכרון ישירות לשכבות ה-attention של המודל באמצעות אותו מנגנון KV Cache.

זיכרון פרמטרי (Parametric) - הפרמטרים של המודל אשר מייצגים את כלל הידע והזיכרון שלו. זיכרון זה נועד בעיקר בשביל שהמודל יוכל לזכור את היכולות שלו, למשל כמומחה לסיכום או יועץ לעריכת דין ועוד. MemOS מאפשר לעדכן את זיכרון זה באמצעות אימון קל יותר סגנון המונחה מתאמים (Adapters), למשל LORA.

MemOS יכול להעביר זיכרון בין סוגי זיכרון אלו לפי הצורך. למשל זיכרון טקסט חופשי שניגשים אליו הרבה יכלו לעבור עיבוד ולהוף לזיכרון אקטיבציה כ-KV Cache שיוזרק לתוך המודל. הדרך ההפוכה גם אפשרית בה זיכרון אקטיבציה שלא משתמשים בו הרבה יהפוך חזרה לזיכרון טקסט חופשי.

כדי לתמוך במעברים אלו בין סוגי הזיכרון השונים, המערכת של MemOS תומכת במה שהם קוראים לו Policy-Aware Scheduling. המערכת מתאימה את סוג הזיכרון בהתאם לצריכה שלו, התאמת התוכן שלו למשימה. זה נעשה באמצעות Contextual Fingerprint שהוא חתימה סמנטית של פיסת זיכרון על ידי וקטור, זה מאפשר שחזור מהיר או התאמה למשימה (חיפוש וקטורי / סמנטי). בנוסף עבור כל זיכרון נשמר לאורך ציר הזמן מתי הוא עודכן מה שמאפשר שקיפות מלאה.

https://arxiv.org/abs/2505.22101

מאמר כחול לבן שאין לא מילה אגנט בפנים: תענוג!