המאמר היומי של מייק: 02.08.25

Mixture-of-Recursions: Learning Dynamic Recursive Depths for Adaptive Token-Level Computation

האם כל הטוקנים צריכים את אותה כמות של "חשיבה"? Mixture-of-Recursions אומר שלא - מאמר מתחרה לזה של אתמול

בואו נתחיל עם אמת שכולנו מכירים בעולם ה-AI: הגדלת מודלי שפה פותחת יכולות מדהימות, אבל זה מגיע עם עלות עצומה. הכוח החישובי והזיכרון העצומים הדרושים לאימון והרצה של המודלים המפלצתיים האלה הופכים אותם לנחלתם של מרכזי נתונים ענקיים בודדים. מציאות זו הציתה חיפוש נרחב אחר עיצובי מודלים יעילים יותר. עד כה, החיפוש הזה התנהל בשני מסלולים עיקריים.

המסלול הראשון הוא יעילות פרמטרים, שמטרתו להפיק יותר ביצועים מפחות משקולות מודל. טריק נפוץ כאן הוא שיתוף פרמטרים, שבו אותו סט של משקולות משמש בחלקים שונים של המודל. המסלול השני הוא חישוב אדפטיבי, שבו המודל משקיע יותר כוח חישובי רק על חלקי הקלט שהם באמת קשים, ומאפשר לחלקים פשוטים יותר לעבור מסלול קל יותר.

בעוד ששתי הגישות הצליחו בנפרד, מודל יחיד שעושה את שניהם בו-זמנית היה חסר. טרנספורמרים רקורסיביים, המשתמשים בסט שכבות משותף שוב ושוב, נראו כמו בסיס מבטיח בגלל שיתוף הפרמטרים המובנה שלהם. עם זאת, רובם השתמשו במספר צעדים קבוע לכל טוקן, ולכן לא יכלו באמת להסתגל לקלט.

כאן נכנס לתמונה המאמר שאנו סוקרים, (Mixture-of-Recursions (MoR. הוא מציג מסגרת חדשה ומשולבת המערבבת בחוכמה את שני סוגי היעילות לעיצוב אחד פשוט. בעיקרון, MoR הוא טרנספורמר רקורסיבי. זה אומר שהוא משתמש ב"בלוק רקורסיה" משותף, חבילת שכבות, מספר פעמים כדי לעבד טקסט, מה ששומר על מספר הפרמטרים נמוך. אבל החידוש האמיתי טמון באופן שבו הוא מחליט כמה פעמים להשתמש בבלוק הזה. במקום מספר קבוע לכל הטוקנים, MoR מציג "נתבים" (routers) קטנים שמחליטים בזמן אמת כמה צעדי רקורסיה כל טוקן בודד צריך.

חשבו על זה כך: עבור טוקן פשוט כמו המילה "את", הנתב עשוי להחליט שמספיק מעבר אחד בבלוק. אבל עבור טוקן עשיר יותר במשמעות או מורכב יותר כמו "בהגנתיות", הנתב עשוי לשלוח אותו דרך הבלוק שלוש פעמים, ובכך להעניק לו יותר זמן "חשיבה". זהו השילוב של חיסכון בפרמטרים וחיסכון בחישוב.

עוד חידוש של MoR היא שהוא לא רק מחבר שני רעיונות; הוא יוצר לולאה חיובית שבה יתרון יעילות אחד מאפשר יתרון אחר. את החידוש של המסגרת ניתן לחלק לשלושה חלקים מחוברים שעובדים יחד:

שיתוף פרמטרים באמצעות רקורסיה: הבסיס של MoR הוא שימוש חוזר בבלוק פרמטרים יחיד. זה באופן טבעי מקצץ את מספר המשקולות הייחודיות שהמודל צריך לאחסן, מה שהופך את המודל עצמו לקטן וחסכוני יותר בזיכרון מההתחלה.

עומק "חשיבה" אדפטיבי באמצעות ניתוב: זהו החידוש הארכיטקטוני המרכזי. על ידי אימון נתב מההתחלה כדי להקצות עומקי רקורסיה ספציפיים לכל טוקן, MoR מתקדם מעבר לגישה הנוקשה של "מידה אחת לכולם" שהייתה במודלים רקורסיביים קודמים. זה לא רק תוסף שמוסיפים לאחר האימון, אלא חלק בסיסי מתהליך הקדם-אימון, המאפשר למודל ללמוד כיצד להקצות את תקציב החישוב שלו ביעילות.

אחסון KV-cache: זוהי תוצאה חזקה וישירה של העומק האדפטיבי. בטרנספורמר רגיל, KV-cache הוא צוואר בקבוק משמעותי בזיכרון בזמן אינפרנס. עם MoR, אם טוקן מנותב לצאת אחרי רקורסיה אחת בלבד, המודל לא צריך לחשב או לאחסן את צמדי ה-KV שלו עבור שלבי הרקורסיה העמוקים יותר. אחסון חכם ובזמן אמת זה מקטין את תעבורת הזיכרון, ובאופן מכריע, מצמצם את חישוב הקשב (attention) היקר רק לטוקנים שעדיין פעילים בעומק נתון.

חבילה זו של "שלוש-באחד" מאפשרת ל-MoR לקשור משקולות כדי לחסוך בפרמטרים, לנתב טוקנים כדי לחסוך בחישובים מיותרים, ולאחסן באופן סלקטיבי את צמדי ה-KV כדי לחסוך בתעבורת זיכרון, והכל בתוך מודל אחד ומאוחד. המאמר בוחן דרכים שונות לבנות את הרעיון הזה, תוך התמקדות בשתי החלטות עיקריות:

אסטרטגיות ניתוב: ההחלטה כיצד לנתב טוקנים כרוכה בבחירה. בניתוב מבוסס מומחה (expert-choice routing), כל שלב רקורסיה פועל כ"מומחה" ובוחר את k הטוקנים המובילים להמשך עיבוד. זה מבטיח תקציב חישוב צפוי מראש, אך עלול ליצור בעיות בסדר המידע במהלך האימון.

בניתוב מבוסס טוקן (token-choice routing), כל טוקן מקבל את מסלול החישוב המלא שלו כבר בהתחלה. זה פותר את בעיית הסדר, אך עלול להוביל לחוסר איזון בעומסים, כאשר שלבים מסוימים מקבלים יותר מדי טוקנים ואחרים פחות מדי.

אסטרטגיות אחסון KV-cache: הכותבים מציעים גם שתי דרכים לנהל את ה-KV-cache. אחסון מותאם-רקורסיה (recursion-wise caching) שומר את צמדי ה-KV באופן מקומי רק עבור הטוקנים הפעילים בכל שלב רקורסיה, מה שממקסם את יעילות החישוב. לחלופין, שיתוף רקורסיבי של KV או (recursive KV sharing) מאחסן את כל צמדי ה-KV בשלב הרקורסיה הראשון ועושה בהם שימוש חוזר בכל השלבים העמוקים יותר. זה מקטין משמעותית את טביעת הרגל של הזיכרון ויכול להאיץ מאוד את שלב העיבוד הראשוני של הקלט, מה שהופך אותו לאופציה אטרקטיבית בסביבות עם זיכרון מוגבל.

התוצאות הניסוייות מרשימות. במגוון גדלי מודלים (מ-135 מיליון ועד 1.7 מיליארד פרמטרים), MoR קובע רף חדש של יעילות (פארטו פרונט). תחת תקציב חישוב אימון זהה, מודלי MoR משיגים שגיאת מבחן נמוכה יותר ודיוק גבוה יותר במשימות few-shot בהשוואה למודלים רגילים ומודלים רקורסיביים סטנדרטיים, למרות שיש להם עד 50% פחות פרמטרים. כאשר הם מאומנים על כמות נתונים זהה, מודלי MoR משיגים ביצועים עדיפים תוך שימוש ב-25% פחות חישובים, ומקצרים את זמן האימון וצריכת הזיכרון.

הארכיטקטורה גם מתרחבת היטב לגודל. ככל שגודל המודל עולה, MoR לא רק שמדביק את הפער מול טרנספורמרים רגילים, אלא בסופו של דבר עוקף אותם, וכל זאת תוך שימוש בכשליש מהפרמטרים הייחודיים בלבד.

השיטה המוצעת הוא יותר מסתם פתרון הנדסי חכם. הוא מייצג שינוי תפיסתי באופן שבו אנו חושבים על ארכיטקטורת מודלים ועל חישוב. הוא מתייחס ל"עומק" המודל לא כמספר קבוע וסטטי, אלא כמשאב דינמי שיש להקצות באופן מדויק, ברמת הטוקן הבודד.

מסגרת זו מגדירה מחדש ובאלגנטיות את תהליך ה"חשיבה" של המודל כסוג של חשיבה סמויה, שבה עומק המחשבה מותאם לקושי של המושג המעובד. על ידי איחוד של שיתוף פרמטרים עם חישוב אדפטיבי, MoR מספק נתיב יעיל וסקיילבילי להשגת היכולות של מודלים גדולים במחיר נמוך יותר.

https://arxiv.org/abs/2507.10524