**להכניע את הדרקון הריבועי באמצעות פילוח ליניארי, סקירה 522**

**סקירת המאמר היומית של מייק: 10.10.25  
COMPLLM: COMPRESSION FOR LONG CONTEXT Q&A**

עקב אכילס של ארכיטקטורת הטרנספורמר הוא, ותמיד היה, מנגנון self-attention שלה. העלות החישובית שלו גדלה באופן ריבועי ביחס לאורך רצף הקלט. זהו לא רק מטרד תיאורטי; זהו מחסום מעשי, המרוקן את הארנק, שהופך את העיבוד של הקשרים ארוכים באמת: ריפואים שלמים, מסמכים משפטיים ארוכים, או רומנים מלאים, ליקר באופן בלתי אפשרי. עבור רצף באורך *N*, העלות פרופורציונלית ל-*N* בריבוע. הכפילו את ההקשר, ואתם מרבעים את עלות שלב העיבוד הראשוני.

התחום התמודד עם בעיה זו על ידי חקר "דחיסה רכה", שבה ההקשר הארוך נדחס לקבוצה קטנה יותר של וקטורים חבויים לפני שהוא מוזן ל-LLM. למרות שמדובר בשיטות די מתוחכמות, רובן חולקות פגם מהותי: הן מתייחסות להקשר כאל גוש מונוליטי יחיד. הן מנסות ליצור ייצוג דחוס הוליסטי אחד של כל הקלט. בכך, שלב הדחיסה *עצמו* הופך לעתים קרובות לבעיה ריבועית, ובכך הוא רק מסיט את צוואר הבקבוק במקום לפתור אותו.

המאמר הזה, CompLLM, מציג שינוי תפיסתי מרענן, פרגמטי ועוצמתי. החידוש המרכזי אינו ארכיטקטורה חדשה ומסובכת, אלא בחירת עיצוב פשוטה ואלגנטית: **אל תדחסו את ההקשר כמכלול.** במקום זאת, המודל מחלק תחילה את ההקשר הארוך לסדרה של מקטעים קטנים ועצמאיים, ולאחר מכן דוחס כל אחד מהם בנפרד.

החלטה זו, שנראית שולית, היא המפתח שפותח שלוש תכונות קריטיות החיוניות לפריסה בעולם האמיתי.

הראשונה היא **יעילות**. על ידי חלוקת ההקשר לנתחים קטנים בגודל קבוע (למשל, 20 טוקנים כל אחד), עלות הקשב הריבועית מוגבלת למקטעים זעירים אלה. העלות הכוללת של שלב הדחיסה, אם כן, גדלה באופן לינארי עם אורך ההקשר. במקום שכל טוקן יצטרך לשים לב לכל טוקן שקדם לו במסמך של 100K טוקנים, הוא שם לב רק לקומץ שכניו במקטע המקומי שלו. זה הופך את עלות הדחיסה מעקומה ריבועית מאיימת לקו ישר וניתן לניהול. המעבר הראשוני (כמה שכבות בודדות) של CompLLM (ה-prefill) עדיין מתמודד עם עלות ריבועית ביחס לאורך הסדרה, אך אם מקטינים את אורך הסדרה בפקטור דחיסה *C*, מקטינים את עלות ה-prefill הדומיננטית הזו בפקטור של *C* בריבוע: רווח תיאורטי עצום.

השנייה היא **יכולת סקיילביליות**. מודל הדחיסה נחשף אך ורק למקטעי טקסט קצרים במהלך האימון. משמעות הדבר היא שניתן לאמן מודל על סדרות קצרות יחסית (למשל, 2K טוקנים) ולאחר מכן, בזמן ההסקה, ליישם אותו בביטחון על הקשרים גדולים בסדרי גודל (100K טוקנים לטענת המאמר). המודל לא צריך ללמוד על תלויות ארוכות טווח במהלך הדחיסה; הוא פשוט לומד מיפוי מקומי מסדרת שיכוני(אמבדינגס) טוקנים סטנדרטיים לסדרה קצרה יותר של "שיכוני קונספט" סינתטיים. משימת ההיגיון על פני ההקשר הדחוס המלא נותרת בידי ה-LLM הראשי.

השלישית, ואולי המשפיעה ביותר ליישומים מעשיים, היא **יכולת שימוש חוזר**. מכיוון שכל מקטע נדחס באופן עצמאי, ניתן לשמור את הייצוג הדחוס שלו בקאש. דמיינו מערכת שבה משתמש שואל שאילתות על מאגר ידע גדול. אם מסמך א' הוא חלק מההקשר בעשר שאילתות שונות, יש לחשב את המקטעים הדחוסים שלו פעם אחת בלבד. בעוזר קידוד, אם מפתח משנה פונקציה אחת, רק המקטעים המתאימים לאותה פונקציה צריכים להידחס מחדש, ולא כל בסיס הקוד מרובה הקבצים. זה בלתי אפשרי בתכניות דחיסה הוליסטיות, שבהן שינוי בטוקן בודד עלול לשנות את כל הפלט הדחוס, מה שמחייב חישוב מחדש מלא.

תהליך האימון עצמו גם הוא חדשני. במקום פשוט לאלץ את המודל להפיק את אותו טקסט היעד, הוא משתמש בתהליך זיקוק (דיסטילציה)מתוחכם המתמקד בחישובים הפנימיים של ה-LLM. המטרה אינה להתאים את הפלט הסופי, אלא ליישר את האקטיבציות הפנימיות שכבה אחר שכבה. עבור כל שכבה ב-LLM, מחושבת פונקציית לוס מסוג Smooth L1 בין אקטיבציות שנוצרו באמצעות ההקשר הדחוס לבין אלו מההקשר המקורי, הלא דחוס. לוס זה מחושב *אך ורק* על הטוקנים התואמים לתשובה, מה שמספק אות אימון צפוף ורלוונטי ביותר. כדי להתמודד עם השונות הרחבה בגודל ההפעלות על פני עומק הרשת, הלוס עבור כל שכבה מנורמל על ידי סטיית התקן של אקטיבציות מודל המורה. יישור רב-שכבתי ומנורמל זה מבטיח שהדוחס ילמד לשמר את הייצוגים הפנימיים המדויקים שה-LLM זקוק להם כדי לגבש תשובה נכונה.

בעיקרו של דבר, החידוש של CompLLM טמון במסגור מחדש של הבעיה. הוא נוטש את החיפוש אחר דחיסה מושלמת ויחידה של מסמך שלם. במקום זאת, הוא יוצר ספרייה של חלקים דחוסים, עצמאיים וניתנים לשימוש חוזר, ומשאיר את הסינתזה הסופית ל-LLM.

https://arxiv.org/abs/2509.19228