# Review 174: In-Context Learning Creates Task Vectors

**Paper: https://arxiv.org/abs/2311.06668v3**

https://huggingface.co/papers/2310.15916

אחד היכולות המדהימות של מודלי שפה ענקיים היא יכולת למידת in-context או ICL בקצרה. ICL היא יכולת של LLM ללמוד מכמה דוגמאות בלי לשנות בכלל את המשקלים שלו. כלומר אנו מעבירים למודל שפה כמה דוגמאות בסגנון (מלון -> צהוב, מלפפון -> ירוק,..) ולאחר המכן אם תזינו למודל ״בננה -> …״, הוא יבין שמדובר בצבע ויענה צהוב.

אבל איך המנגנון הזה עובד? המאמר המסוקר טוען ומראה שמדובר כאן בתהליך דו שלבי:

– הזנה של הדוגמאות (נסמן אותם ב S) המחשבים את הפרמטרים של פונקציה מסוימת (בהמשך נסביר איך היא בנויה) שתופעל על דוגמת הטסט x (בננה במקרה המתואר).

– הפעלה של פונקציה זו על שאילת טסט x. המאמר טוען שהפרמטרים האלו לא תלויים בשאילתת הטסט x עצמו אלא רק ב- S (במאמר זה מנוסח בצורה מתמטית יפה שמאוד אהבתי). ההשערה הזו היא לא לגמרי טריוויאלית כי בארכיטקטורת הטרנספורמרים הייצוג של דוגמאות מתויגות S תלוי גם בשאילתה x.

המאמר מראה שב- ICL ניתן להגיע להפרדה כזו בין ייצוג המשימה (הנגזר מ- S) וייצוג השאילתה x. אוקיי, אז מה זה הפרמטרים האלו שמחושבים רק על דוגמאות S? המאמר טוען הם בעצם הפלטים של שכבה L של הטרנספורמר עבור הטוקנים של S כאשר L אינה השכבה האחרונה של מודל השפה. פרמטרים אלו מגדירים(דרך הזנה) לפונקציה שהיא הפעלה של השכבות הנותרות על פלט זה (= ייצוג המשימה) וגם על השאילתה x.

איך הם בדקו זאת? אוקיי, השאילתה מורכבת מגוף השאלה (בננה בדוגמה שלנו) ובסימן שאלה מאולתר (״->״ במקרה) שלנו המאותת למודל שפה שהוא צריך לפתור אותה. אז המחברים העתיקו את ייצוג של ״->״ בשכבה L עבור דוגמא לא קשורה x' ואז ממשיכים עם השאילתה המקורית לאחר מכן. המאמר מראה שעבור שכבה מסוימת L החלפה כזו לא מובילה לירידה ניכרת בביצועים(יחסית לייצוג של ״->״ הנבנה באופן רגיל).

כלומר הפלט של שכבה L של מודל שפה עבור הטוקנים של S אכן לא תלויה בשאילתה x. מה שמעניין שעבור מודלי שפה בגדלים שונים L האופטימלי יצא בערך 15. מאמר די מעניין שנותן הסבר מסקרן למה ואיך ICL עובד. יהיה מעניין לראות מה קורה במקרים שמודל שפה נכשל ב-ICL אם מופעל בצורה הרגילה. האם ההפרדה הזו תישמר?