# Review 24: RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning

**Paper: https://arxiv.org/abs/2205.12548v3**

פינת הסוקר:

המלצת קריאה ממייק וקרין: שווה קריאה למתעניינים ב-RL ו-NLP

בהירות כתיבה: בינונית פלוס

ידע מוקדם:

היכרות עם עקרונות מודלי שפה גדולים

הבנה בסיסית בהנדסת פרומפטים

היכרות עם עקרונות של למידה באמצעות חיזוקים (Reinforcement Learning -RL)

יישומים פרקטיים:

פרומפטים משופרים למודלי שפה

פרטי מאמר:

פורסם בתאריך: 22.10.22, בארקיב.

הוצג בכנס: ---.

תחומי מאמר:

מודלי שפה גדולים

הנדסת פרומפטים

למידה באמצעות חיזוקים

כלים מתמטיים, מושגים וסימונים:

מודלי שפה גדולים - LLMs

למידה באמצעות חיזוקים - RL

הנדסת פרומפרטים - PrEng

מבוא:

פרומפט (promptֿ, הנחיה בעברית) היא דרך תקשורת עם LLM-ים מאומנים. למעשה פרומפט היא טכניקת ניסוח של השאילתות ל-LLM-ים בשפה שהם מבינים שמאפשרת לנו לנצל את הידע החבוי בפרמטרים שלהם לביצוע מגוון משימות קצה. יותר ספציפית, פרופמט הוא פיסת טקסט שמכילה לרוב את הגדרת המשימה ולאחר מכן קלט, כך שהמודל שפה יכול לייצר את התשובה בפורמט האהוב עליו - ע"י פתרון בעיית "מיסוך" וכו' (masking problem) שמודלי שפה ״אוהבים״. כלומר המודל מתבקש להשלים את המקומות הממוסכים בקלט. למשל עבור משימת תרגום, הפרומפט יכול להיראות באופן הבא (למשימת few-shot learning):

איור 1: דוגמא של פרומפט (נלקח מבלוג)

נושא של הנדסת פרומפטים הפך להיות מאוד פופולרי בשנים האחרונות. יצאו מאות מאמרים המציעים גישות להנדסת פרומפטים וגם הסברים למה זה עובד. רוב השיטות להנדסת פרומפטים מנסות לאתר פרומפט אופטימלי רך (soft prompt) כלומר שיכון (embedding) של פרומפט. לשיטה זו יש כמה חסרונות בולטים:

קושי ב-explainability

פרומפטים רכים עלולים לא לעבוד עם מודלי שפה שונים

לא ניתן לבנות פרומפט אם אין לנו גישה לגרדיאנטים של מודל שפה (למשל כאשר אנו עובדים עם מודל שפה דרך API)

תמצית מאמר:

המאמר שנסקור היום מציע שיטה המאפשרת חיפוש פרומפטים עצמם ולא וקטור השיכון שלהם. כלומר הפלט של השיטה הוא סדרת הטוקנים הבונים את הפרומפט. שיטה זו לא דורשת ידע על מודלי שפה שעבורו נבנה הפרומפט ועקב כך מאפשרת מציאת פרומפט אופטימלי כאשר מודל שפה נגיש דרך API.

נזכיר כי מציאת פרומפט אופטימלי (ולא ייצוגו) היא בעיית אופטימיזציה דיסקרטית (מרחב החיפוש הוא מרחב הטוקנים). נובע מכך שאנו לא יכולים לחשב גרדיאנטים (לפחות לא באופן ישיר) ולבצע gradient descent וזה די טבעי שהמחברים בחרו להשתמש בשיטה מלמידה באמצעות חיזוקים (RL) למשימה זו. אבל איך בכלל מפעילים טכניקות RL למודלי שפה (למשימות גנרוט). בשביל כך אנו צריכים להגדיר מה היא פעולה, מצב (state), אסטרטגיה ותגמול. הפעולה כאן היא יצירת טוקן הבא, האסטרטגיה היא בעצם פונקציית softmax המגדירה הסתברות של כל טוקן והתגמול (reward) מודד את הדמיון בין פלט לבין ה-ground truth.

כדי ליצור פרומפט אופטימלי הכותבים מאמנים מודל, שהוא LM מוקפא (לא חייב להיות זהה ל-LLM שעבורו נבנה הפרומפט) עם רשת MLP מאומנת בסוף. רשת זו נקראת במאמר Policy LM (מודל שפה לבניית הפרומפטים). המאמר משתמש באלגוריתם RL מסוג on-policy לאימון של Policy LM, כאשר המטרה לאמן את ה-MLP ב-Policy LM כך שהיא תמקסם פונקציית תגמול. באופן טבעי פונקצית התגמול אומדת עד כמה הצלחנו להפיק תוצאה רצויה מהפרומפטים, שנבנו באמצעות Policy LM.

הסבר של רעיונות בסיסיים:

אחד האתגרים המרכזיים לאימון מודל RL היא בחירה של פונקציית תגמול. כאמור במקרה שלנו פונקציה תגמול לפרומפט נתון מודדת את ״איכות הפרומפט״, כלומר עד כמה הצלחנו להוציא איתו תשובות טובות מה-LLM. מרחב התגמולים הינו מאוד דליל כי התגמול מתקבל רק לאחר סיום בניית הפרומפט ולא לגמרי ברור איך לשערך אותו במהלך בניית הפרומפט (אחרי כל טוקן). עקב כך הכותבים מציעים שיטת בניית תגמול ייחודית פר מטלה. למשל למשימת הסיווג few-shot המאמר מנסה לא סתם למקסם את ההסתברות של קטגוריית היעד אלא ממקסם את ההפרש בין הסתברות קטגוריית היעד לבין הקטגוריה בעלת הסתברות מקסימלית בין שאר הקטגוריות. בנוסף הכותבים ״מצ'פרים״ (מגדילים) את Policy LM כאשר הפרש זה הוא חיובי (ראה פרק 3.1 במאמר להסבר יותר מעמיק).

אחרי שהבנו איך נבנית פונקציית התגמול, כעת נפרט איך הכותבים ממקסמים אותה. המאמר בגרסת on-policy של אלגוריתם שנקרא (SQL (Soft Q-Learning. נזכיר כי אלגוריתמי on-policy הם סוג של אלגוריתמי RL שבהם הסוכן משתמש באסטרטגיה (policy) הנוכחית כדי לקבוע את הפעולות ולומד מתגמולים שהוא מקבל. המשמעות היא שהסוכן מעדכן את האסטרטגיה על סמך התגמולים מהסביבה, מה שמביא לאסטרטגיה חדשה שמשמשת אותו לאחר מכן.

בגדול SQL ממקסם את התגמול הכולל אבל יחד עם זאת מנסה למקסם את ״האקראיות״ של האסטרטגיה(שהיא בעצם אנטרופיה של התפלגות הטוקנים) . איך משיגים זאת? פשוט מוסיפים איבר אנטרופיה לביטוי של פונקציית התגמול המקורית. בהמשך SQL מנצל את הקשר בין הקלט לשכבת softmax של מודל שפה לבין Q-value שהיא תגמול ממוצע למצב (כל הטוקנים עד t-1) ופעולה (הטוקן הנבחר בסיבוב t) נתונים כדי לבנות פונקציית תגמול ״נוחה״.

כלומר האימון של RLPrompt מתבצע באופן הבא:

מכניסים את ״הגדרת המשימה״ ל Policy LM (ראה אפנדיקס A2 להסבר מפורט יותר)

מזינים את השיכון (embedding) של המשימה ל- MLP שמוציא לנו פרומפט עבור המשימה

מכניסים פרומפט זה למודל שפה שעבורו אנו מייצרים אותו יחד עם הדוגמאות שיש לנו (few shot learning)

מחשבים ערך פונקצית התגמול (וגם Q-value) עבור הפלט של מודל שפה

מבצעים עדכון של האסטרטגיה (בגדול מעדכנים את המשקלים של ה-MLP) על סמך הערכים שחישבנו בסעיף הקודם.

משימות שנבחנו:

השיטה המוצעת ניתנת ליישום עבור מרחב גדול של משימות. במדידה התרכזו בשתי משימות עיקריות: קלסיפיקציה וג׳ינרוט.

קלסיפיקציה: משימת Few-Shot Text Classification: סיווג של טקסט תוך שימוש במספר קטן של דוגמאות אימון. מדדו על דאטאסטס פופלריים למשימות סיווג סנטימנט וסיווג נושאים בטקסט. משימת הקלסיפיקציה שקולה למשימה של בחירת ה- tokens אשר בפורמט “[Input] [Prompt] [MASK]” ממקסמים את ההסתברות ל MASK token.

ג׳ינרוט: משימת Unsupervised Text Style Transfer Text: בהינתן טקסט מקור, יש לכתוב אותו מחדש בסגנון יעודי. למשל משפט בסנטימנט חיובי שנרצה לכתוב מחדש בסנטימנט שלילי (״האוכל טעים״ ← "האוכל מגעיל״). המדידה נעשית באמצעות הערכה ידנית של מתייגים, וכן באמצעות חישוב ציונים של Content (מידת שימור התוכן), Style (מידת הדיוק בסגנון הרצוי) ו- Fluency (מידת הרהיטות // שטף) עבור הטקסט המג׳ונרט.

ניתוח תוצאות:

עבור משימת קליסיפיקציה, השיטת המוצעת מראה שיפור משמעותי על פני שיטות אחרות כמו Prompt Tuning ו- Instructions, ולמעשה משיגה ביצועים משופרים על כל ה- benchmarks המדווחים.

עבור משימת הג׳ינרוט, השיטה המוצעת מסוגלת ללמוד פרופמטים ללא שימוש במידע מתוייג. בהשוואה לשיטות אחרות, מדד ה- Content השיג ביצועים מעט נמוכים יותר, ואילו במדד ה- Fluency נרשם שיפור משמעותי.

כמו כן, התוצאות מראות כי בהרבה מהמקרים פרומפטים שטובים עבור משימות downstream הם חסרי קוהרנטיות, דוגמא לפרופמט נלמד עבור משימת סנטימנט: “Parameters Comparison )=( Compare either”. חוסר קוהרנטיות של הפרומפטים מקשה באספקט של פרשנות והבנה, ומכאן שמודלי שפה עושים שימוש שונה בפרומפטים מאשר משתמש אנושי.