



סמינר בנושא מיוחד במדעי המחשב - האוניברסיטה הפתוחה

Large Language Model Agents

מנחה: ד"ר מיה הרמן

מגיש: בוריס ברזנר

תאריך: 25/08/2025



איור 1 – תמונה שהופקה ע"י מודל גנרטיבי Dall-E 2 של חברת OpenAI עבור ההנחיה:

"Depict an LLM agent running a software company; setting meetings, building software, running finance department, all while making coffee and chaos is in the office"

תוכן עניינים

2	תוכן עניינים.....
3	1. מבוא.....
4	2. חלק תאורטי: המרכיבים של סוכן מודל שפה גדול.....
4	2.1 מודל שפה גדול (LLM).....
4	2.1.1 תקציר ההתפתחות של מודל שפה גדול.....
5	2.1.2 מרכיבי מודל שפה גדול.....
6	2.1.3 המקודד (Encoder).....
6	2.1.4 המפרש (Decoder).....
6	2.1.5 השנאי (Transformer).....
9	2.1.6 מנגנון הקשב (Attention Mechanism).....
12	2.2 כלים חיצוניים.....
12	2.2.1 מוטיבציה.....
13	2.2.2 שיפור ביצועי מודל בעזרת כלים (ToolFormer).....
16	2.2.3 מפרש קוד (Code Interpreter).....
20	2.3 חשיבה עמוקה.....
20	2.3.1 שרשרת מחשבה (Chain of Thought – CoT).....
21	2.3.2 עץ מחשבה (Tree of Thoughts – ToT).....
22	2.3.3 גרף מחשבה (Graph of Thought - GoT).....
25	2.4 מערכת מרובת סוכנים (Multi-Agent System).....
25	2.4.1 מוטיבציה.....
25	2.4.2 ארכיטקטורות.....
	3. חלק מעשי: הדגמה – פיתוח מערכת מסחר אלגוריתמי במטבעות קריפטוגרפיים באמצעות
27	מערכת מרובת סוכנים.....
27	3.1 מבוא.....
27	3.2 המערכת AlgoTrader.....
28	3.3 תהליך הפיתוח וגישות שונות למערכות מרובות סוכנים.....
32	4. סיכום ומסקנות.....
33	מקורות.....

1. מבוא

המצאת הגלגל בשנות ה-4000 לפני הספירה שינתה את פני האנושות באופן משמעותי. במקום לסחוב את הסחורה על הגב, או על חמור, האדם הרכיב סחורה כבדה יותר למרכבה והתאמץ פחות במסעו. מה היא ההברקה הזו שבעזרת מחשבה יצירתית ומעט הנדסה, חייהם של כלל בני האדם הושפעו, והוקלו כתפיהם של נושאי המשקל? כמו שהמצאה זו השפיעה על תפוקתו של האדם, גם היום ישנם פיתוחים של טכנולוגיה חדשנית המשפיעה בצורה זו – סוכן מודל שפה גדול.

סוכן מודל שפה גדול הינו ישות אוטונומית המבצעת הנחיות מורכבות של משתמש, כאשר התקשורת בין המשתמש לסוכן מתבצעת בשפה טבעית. הסוכן מסוגל לקבל ולהבין הנחיות מורכבות מהמשתמש, לתכנן את אופן הפעולה של עצמו ע"י חלוקת ההנחיה לתת משימות, לבצע את המשימות ע"י שימוש בכלים הקיימים לרשותו, ואם אין לו אותם אז יפתח אותם לעצמו, וכל זאת תוך שהסוכן מודע להקשר בו הוא פועל ומאמת את רלוונטיות הפעולות שלו למשימה. כך, הסוכן הינו עוזר דיגיטלי בעל יכולות גבוהות בתחומים רבים ומגוונים, ביניהן הנדסת תוכנה, ניתוח נתונים, ניהול יומן פגישות, סיעור מוחות, ניהול השקעות פיננסיות ועוד.

בסמינר זה אציג את הטכנולוגיה העומדת בבסיס הסוכן, דוגמאות לשימושים עכשוויים, ואבצע הדגמת הרצה של מערכת מרובת סוכנים שפותחה על ידי קבוצת חוקרים במטרה להחליף חברת הייטק בקבוצת סוכנים מומחים הפועלים בעצמאות מלאה ובעלות מסוגלות לפתור משימות מורכבות בתחום מדעי הנתונים וכאמור מניבה תוצאות המתחרות עם פיתוחים בקדמת התחום. פרויקט ההדגמה - מערכת מסחר אלגוריתמי במטבעות דיגיטליים.

2. חלק תאורטי: המרכיבים של סוכן מודל שפה גדול

2.1 מודל שפה גדול (LLM)

סוכן מודל שפה גדול (Large Language Model Agent – LLM Agent) או כפי שנקרא לו "סוכן" בסמינר זה, הינו אוסף של טכנולוגיות החבורות זו בזו בצורה חכמה ואלגנטית המפיקות משילובם יכולת פתרון בעיות גבוהה, אותה אנו רותמים למען ייעול התפוקה שלנו בעבודה ובחיים האישיים. בין אם אנו מודעים לכך או לא, כמעט בוודאות שמרכיבים מסוימים בחיינו מושפעים מחלק או מכלל הטכנולוגיות שיוזכרו בסמינר זה. השפעה זו באה לידי ביטוי בתחומים כמו שימוש במנועי חיפוש, ניהול יומן בעזרת פקודות קוליות, בהתקשרות מול תמיכת לקוחות של חברת הטלפון שלנו ועוד.

אבני היסוד של הסוכן:

1. מודל שפה גדול (LLM) – רשת נוירונים גדולה אשר בהינתן קלט של רצף מילים מפיקה את המילה הבאה ברצף בהסתברות גבוהה
2. כלים חיצוניים למודל השפה (Tools) – כל משאב הניתן להפעלה למען קבלת נתונים אשר ישמשו את מודל השפה למתן תשובה איכותית יותר לבקשת המשתמש
3. מחשבה עמוקה (Reasoning) – חלוקת תהליך המענה לשלבים אשר בכל אחד מהם מוודא מודל השפה את תקינות ורלוונטיות הפלט להנחיה, זאת על ידי ניצול יכולות ההסקה שתמונות בטבע המודל

בסמינר זה נצלול לעומק של כל אחת מאבני היסוד ונפרט את המדע והטכנולוגיה העומדות בבסיסן. לאורך הסמינר נשלב את אותן אבני יסוד לכדי בנייה הדרגתית של קונספט הסוכן. לאחר מכן, נדון במערכת מרובת סוכנים, ממ"ס (Multi-Agent System, MAS), ולבסוף נבצע הדגמה של ממ"ס ייעודית על מנת לבנות מערכת למסחר אלגוריתמי במטבעות.

2.1.1 תקציר ההתפתחות של מודל שפה גדול

בתחילת המאה ה-20 ניסו מדענים בתחום הפנומנולוגיה (תחום מחקר פילוסופי שניסה להעניק פירוש לוגי מתמטי לתחומי הדעת) באוניברסיטת ג'נבה השוויצרית להביע סמנטיקה של שפה אנושית ע"י חוקים לוגיים. תחום מחקר זה קיבל את התואר "עיבוד שפה טבעית" (Natural Language Processing – NLP) ובכך הזניק תחום שלם של פיתוח ומחקר בנושא זה.

בסוף מלחמת העולם השנייה (1945) ניסו הוגי דעות ומדענים להבין כיצד מחזקים את הסחר הבינלאומי. בעקבות כך, החלה עבודה אקדמאית רבה ליצירת מכונה המסוגלת להבין את השפה האנושית, לתרגם אותה לשפת מחשב ובסוף גם לשפות זרות. אך כפי שהתברר זו אינה משימה קלה לביצוע. השפה האנושית היא כאוטית (chaotic) ורוב המילים תלויות בהקשרן במילים אחרות במשפט ואינן נושאות פירוש חד משמעי וברור.

לעומת השפה האנושית, המתמטיקה ניחנת בחוקיות עקבית, משמעות ברורה ובלתי ניתנת לוויכוח. בשנות ה-80 פותח במעבדת מחקר של חברת IBM מודל השפה הסטטיסטי הראשון הפועל באופן איטרטיבי לחיזוי המילה הבאה במשפט נתון בעזרת אוצר מילים מוכן מראש. מודל זה אומנם היה מהפכני באותה התקופה, אך ביצעו נחשבים חלשים מול מודלים מבוססי רשת הקיימים היום. האתגר המרכזי בשנות ה-80 היה שאין מספיק מידע דיגיטלי הניתן לשימוש בבניית מודל שפה מורכב יותר.

האינטרנט העולמי (World Wide Web – WWW) הפך זמין לקהל הרחב בתחילת שנות ה-90 והחל לאגור מאמרים מדעיים מכלל אוניברסיטאות העולם הניעה תופעה של הצטברות מידע ברשת. זאת, עם הכניסה של יחידות עיבוד גרפיות (Graphical Processing Unit – GPU) לשוק ביחד עם כוח העיבוד הכללי הגובר פי שתיים מדי שנתיים (חוק מור) יצרו לראשונה את התנאים ההכרחיים לאימון מודל שפה מורכב המבוסס על רשתות נוירונים עמוקות – מודל השפה הגדול.

2.1.2 מרכיבי מודל שפה גדול

המרכיב העומד ביסודות הסוכן הוא מודל שפה גדול, ולכן העמקה בראשון דורשת דיון בשני. טכנולוגיות מודלים לשפה גדולים מקבלות תאוצה פנומנלית בשנים האחרונות, וכל מי שהשתמש בפעם הראשונה בוודאי הופתע מהיכולות של מודלי שפה מפורסמים כמו ChatGPT של OpenAI או Claude של Anthropic להבין את ההנחיה שלנו כמה כללית, מורכבת, או לא מדויקת שתהיה ולפלוט מענה מדויק.

ההתפתחויות המובהקות בתחום הבינה המלאכותית התאפשרו בעקבות פריצת דרך בשיטת האימון של מודלי שפה גדולים והתפרסמה במאמר המפורסם "Attention Is All You Need" [1]. במאמר, השתמשו חוקרים מגוגל בשיטה ידועה לקודד ולפענח משמעות של טקסט אנושי ע"י מנגנון "תשומת לב" או "קשב" (Attention mechanism), בתוך הארכיטקטורה המוצעת במאמר. ארכיטקטורה זו הביאה לשני פריצות דרך משמעותיות:

1. זניחה של פתרונות קודמים כמו רשתות נשנות RNN (Recurrent Neural Networks) או רשתות קונבולוציה, בהם נדרש כוח חישובי משמעותי והכרחיות בחישוב סדרתי

2. מנגנון קשב רב-ראשי (Multi-headed attention mechanism), שהוצע במאמר מאפשר חישוב מקבילי

לארכיטקטורת רשת זו קוראים **השנאי (Transformer)**, והוא עומד בבסיס ההצלחה של מודלי שפה גדולים, ובפרט בסוכנים. אך לפני שנציגו, נציין שני מרכיבים חשובים המופיעים כמעט בכל ארכיטקטורת מודל שפה (ובפרט בשנאי): **המקודד (Encoder)** וה**מפרש (Decoder)**.

2.1.3 המקודד (Encoder)

הרעיון העומד מאחורי המקודד הינו להמיר רצף מילים בשפה אנושית לייצוג וקטורי הכולל בתוכו מידע על משמעות ההקשר בין המילים. לדוגמא, מקודד טוב ידע לייצג את המילה¹ "קציצה" באמצעות וקטור קידוד שונה עבור כל משפט:

1. אני אוהב לאכול קציצה
2. הפסקתי להתאמן ונהייתי קציצה

במשפט הראשון הכוונה היא לקציצת בשר, ובשני לשון הגה לאדם שהעלה משקל בעקבות היעדר פעילות גופנית.

המקודד בלבד, באמצעות ייצוג וקטורי מקודד משמעות עבור המילים ברצף הקלט, שימושי לביצוע מספר פעולות וביניהן סיווג טקסט, זיהוי סנטימנט, הערכת דמיון בין טקסטים שונים ועוד.

2.1.4 המפרש (Decoder)

באופן משלים למקודד, המפרש ממיר רצף של ייצוגים וקטוריים מקודדי משמעות לשפה אנושית. הוא מבצע זאת באופן איטרטיבי, כך שבכל איטרציה פולט המפרש את המילה בעלת היתכנות הסתברותית גבוה יותר מכל המילים במילון² להופיע לאחר פלט הרצף באיטרציה הקודמת. תרגום בין שפות טבעיות, מחולל שפה טבעית, קוד והשלמת טקסט נמנים בתור חלק מהיישומים של רכיב זה.

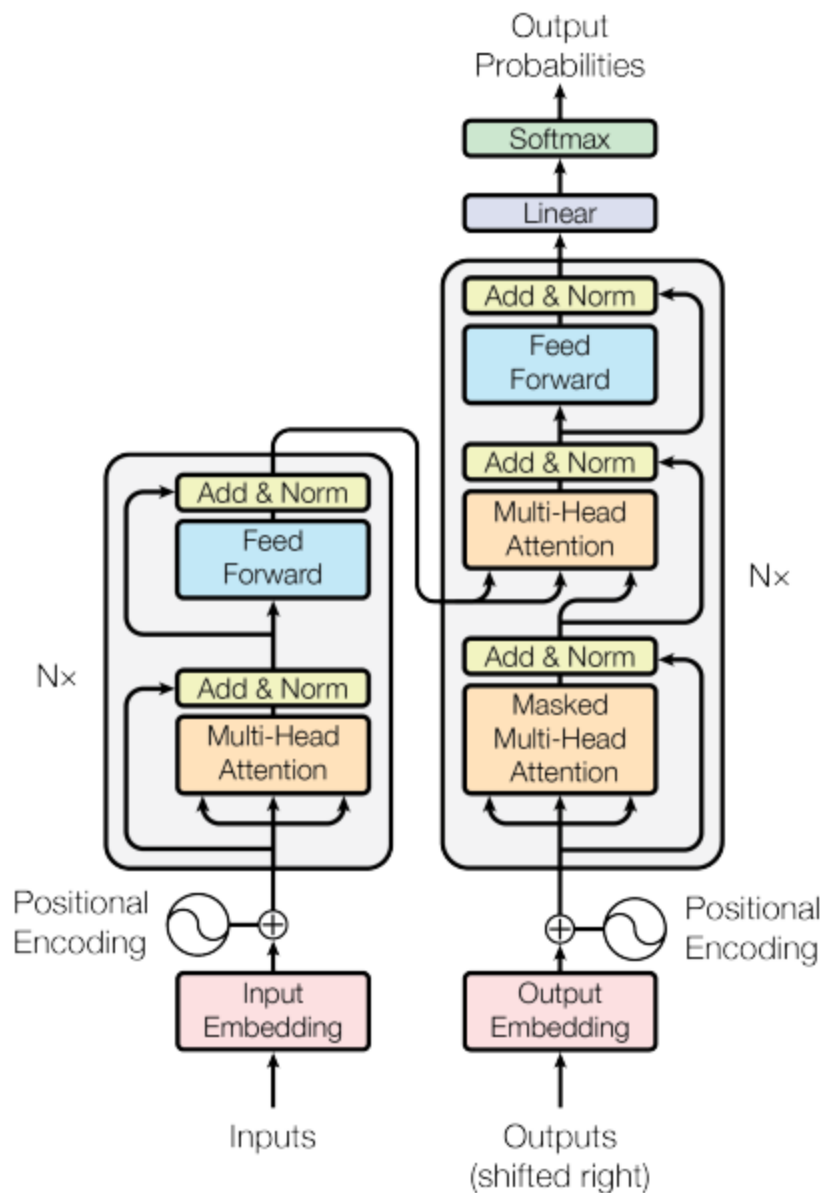
2.1.5 השנאי (Transformer)

השנאי מהווה ארכיטקטורה הכוללת בתוכה מקודד ומפרש. ניתן לחשוב על השנאי בתור פונקציה $t(seq) = seq'$ המקבלת רצף סמלים $seq = (x_1, x_2, \dots, x_t)$ בקלט ופולטת רצף סמלים $seq' = (y_1, y_2, \dots, y_s)$ אחר בהתאם לאימון הרשת, לתהליך זה קוראים התמרת רצף (sequence transduction).

להלן דיאגרמה של ארכיטקטורת השנאי:

¹ בפועל קיים מנגנון הממיר מילים שלמות למילים חלקיות הנקראים סמלים (tokens), אך בסמינר זה נתייחס למילים שלמות בתור סמלים.

² מילון הינו אוסף כל המילים (סמלים) המופיעים בטבלאת ההטמעה (embedding table) של המודל.



איור 2 – השנאי כפי שתואר במאמר [1]

פעולתו של השנאי מתחלקת שני שלבים, שלב הקידד ושלב הפירוש, ותיאורם מופיע באיור 2 בחלק השמאלי והימני בהתאמה. נפרט את פעולתם:

שלב הקידוד

1. הטמעה (embedding): המילים עוברות פונקציה הממירה מילה בשפה אנושית לוקטור d -מימדי הנשלף ממילון שהוכן מראש

$$x'_i = \text{Embed}(x_i)$$

2. קידוד מיקום (positional encoding) – הוקטור x'_i מקודד יחד עם מידע על המיקום שלו ברצף
3. המקודד – מחולק לשתי תת שכבות, האחת מנגנון הקשב והשנייה רשת הזנה קדמית. באמצעות השניים מפיק המקודד וקטור חדש z_i הנושא את משמעות המילה x_i בהקשר של הרצף בו היא מופיעה (ראה סעיף המקודד)

$$z_i = \text{Encoder}(x'_i)$$

נדון במנגנון הקשב בהמשך הסעיף.

שלב הפירוש

שלב זה מתחיל עם הזנה ימינה של **פלט המפרש**, פעולה הכרחית להתנעת המפרש. הזנה ימינה **באיטרציה הראשונה** של המפרש מייצרת סמל תחילת משפט הנקרא <BOS> (Beginning Of Sentence) והוא תוחם צד אחד של הרצף, כאשר אחריו כבר מופיעה המילה הראשונה בחיזוי בעקבות התוצר של המקודד. נציין שבסיום האיטרציה האחרונה של המפרש נוצר תו משלים הנקרא <EOS>, והוא תו חוקי בעל היתכנות הסתברותית.

1. פלט המפרש מוטמע (embedded) ומקודד מיקום בדומה לפעולות הראשונות בשלב הקידוד
2. תת שכבת קשב עם מסכה. מטרתה להתמיר את רצף הפלט של המפרש לכלול מידע על ההקשר של המילים בו. המסכה מונעת מתת שכבה זו לחשב הקשרים של מילים שעתידין להיכנס למפרש – במילים אחרות בכל איטרציה מחושבת טבלאת ההקשרים רק בין המילים בפלט המפרש
3. תת שכבת קשב ללא מסכה (כמו במקודד) המחשבת שוב את רמת הקשב, כעת בין המילים בפלט המפרש לבין פלט המקודד
4. שכבת הזנה קדמית לטיוב הנתונים
5. שכבה לינארית
6. פונקציית softmax הידועה שפולטת הסתברויות בין כל הסמלים האפשריים.

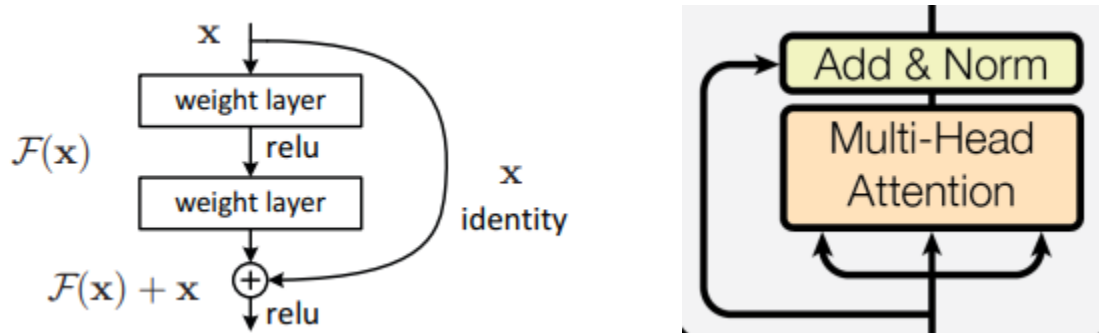
נסכם את פלט שלב הפירוש בכתיב מתמטי:

$$y_{i+1} = \text{Decoder}((y_1, \dots, y_i), (z_1, \dots, z_t))$$

כאשר y_i היא מילה ברצף הפלט השנאי, ו- z_i מילה ברצף הקלט של המפרש.

חיבור שיורי (residual connection)

בארכיטקטורת השנאי ניתן לראות חיבורים מהסוג הזה:



איור 3 חיבור שיורי בשנאי

זהו חיבור שיורי, מטרתו לפתור את בעיית הגרדיאנט הנעלם על ידי סיפוק נתיב ישיר לדילוג הגרדיאנט מעל הפונקציה הנתונה. בנוסף הוא מספק אופטימיזציה לתהליך הלמידה – פונקציית היחידה טריוויאלית ללמידה במקרה זה, נמחיש על ידי הצגת בעיה:

המקרה הלא שיורי – המודל מוצא קירוב טוב לפונקציה $y = f(x)$. נניח כי השגיאה של המודל מינימלית עבור פלט $y = x$, זאת אומרת במקרה שבו f הינה פונקציית היחידה. המודל מאתחל משקולות (פרמטרים) בסביבה קרובה ל-0, ולאחר הרבה איטרציות מחפש איזון של המשקולות שיהווה קירוב לפונקציית היחידה – וקרוב לוודאי לא יתכנס אליה.

החיבור השיורי מחפש קירוב לפונקציה $y = f(x) + x$, ומוצא בנקל את פונקציית המטרה עבור איפוס הפונקציה $f(x) = 0$.

2.1.6 מנגנון הקשב (Attention Mechanism)

בלב השנאי, מודל השפה והסוכן, נמצא מנגנון הקשב והוא אחראי על התהליך המתמטי שמשלב סינטקטיקה, משמעות, והקשר בתוך וקטור, כך שהאינטראקציה אתו יכולה להרגיש כמו עם בן אדם כאשר בפועל המנגנון הוא אינו אלה מכפלה של מטריצות והרכבה של פונקציה לא ליניארית.

תחילה נציג את היחידה הבסיסית של מנגנון הקשב מבוסס מכפלה פנימית ממושקלת (scaled dot product attention), ואחריה את הגרסא המקבילית שלה כפי שהוצגה במאמר [1].

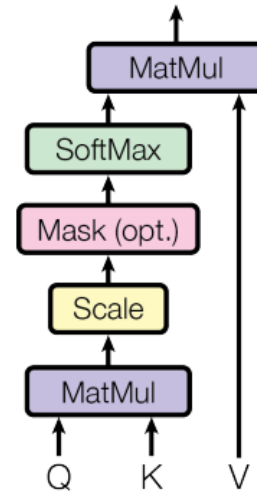
Scaled Dot-Product Attention

מטרת המנגנון היא למצוא מערכות יחסים בתוך הרצף על ידי חישוב "רמת הקשב" (attention score) בין כל זוג מילים ברצף. המטריצות Q, K ו- V (שאלתה, מפתח, וערך) מחושבות באופן הבא:

$$Q = W_Q \cdot E, \quad K = W_K \cdot E, \quad V = W_V \cdot E$$

כאשר E היא מטריצת המילים המוטמעת ברצף הקלט ומטריצות ה- W_x הם הפרמטרים של המנגנון. המנגנון מקיים את המשוואה הכללית הבאה:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V$$



איור 4 - קשב מכפלה פנימית ממושקלת [1]

משמעות המטריצות:

- Q , שאלתה. עמודה i במטריצה מייצגת את הדרישה של מילה i בחיפוש מילים התומכות במשמעות שלה. ניתן לתאר זאת גם ע"י הצורך שבמידה ומילה אחרת נענת אליו, המכפלה הפנימית של שתי העמודות (המילים) תניב ערך מספרי גבוה.
- K , המפתח. עמודותיה מייצגות את הרלוונטיות שיש למילים להציע. לדוגמא אם העמודה K_i מקודד משמעות של שם תואר, ועמודה Q_{i-1} כלשהי היא שם עצם, תהיה רמת קשב (מידת רלוונטיות) גבוהה בין השניים.
- V , הערך. טומן בתוכו את ההטמעה המקורית של המילה ומיקומה, לאחר התמרה לינארית.

השלבים בחישוב:

1. המכפלה QK^T היא מטריצת דירוג רלוונטיות בין מילים, ערכים גבוהים יותר מסמלים רמת קשב חזקה.
2. המכפלה בסקלר $\frac{1}{\sqrt{d_k}}$ מתמודדת עם בעיית הגרדיאנט המתפוצץ\נעלם.
3. פונקציית $softmax$ מניבה עבור כל מילה את התפלגות ההסתברויות של מידת הרלוונטיות של מילים אחרות כלפיה.
4. לבסוף, המכפלה במטריצת הערכים V מפיקה רצף סופי בו רמת ההקשר בין כל זוג מילים ברצף משולבים בתוכו.

לסיכום, בתום פעולת מנגנון הקשב מתקבלים אוסף של וקטורים המייצגים את המשמעות ההקשרית ומידתה בין סמלי הרצף. זהו ייצוג מתמטי של שפה השואף להתחקות להבנה האנושית.

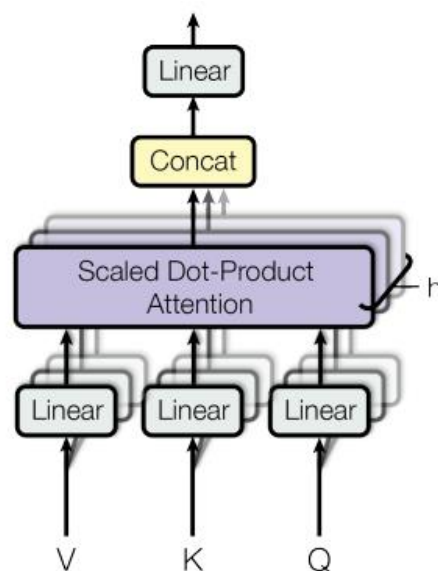
Multi-Head Attention

בחידוש שהוצג במאמר [1] שיכפלו החוקרים את קשב המכפלה הפנימית הממושקלת על מנת לאפשר מקביליות.

בשונה ממנגנון הקשב הבודד שתואר לעיל, שלושת מטריצות הפרמטרים Q , K ו- V מתחלקות בגודלן ל- h מטריצות קטנות יותר, כאשר h מייצג את מספר הראשים שיש במנגנון, קרי "מרובה ראשים".

כל ראש של המנגנון מפיק רמות רלוונטיות שונות בין המילים, והפלט של כלל הראשים משורשרים יחדיו לפני שנכנסים לשכבה לינארית אחרונה.

החוקרים הראו שחלוקה זו משווה ביצועים למנגנון הקשב הבודד ואפילו מייצבת את תהליך הלמידה, כל זאת תוך שמתאפשרת הרצה מקבילית.



איור 5 - קשב מרובה ראשים [1]

לסיכום, מודל שפה גדול הינו כלי לעיבוד שפה טבעית הפולט טקסט בהינתן טקסט. בהינתן מידע טקסטואלי בכמות מספקת, ניתן לאמן את המודל לרמות גבוהות של הבנה, הסקה ותקשורת עד כדי רתימתו לביצוע משימות יום-יומיות בעבודה ובחיים האישיים (כמו ניסוח מיילים והודעות).

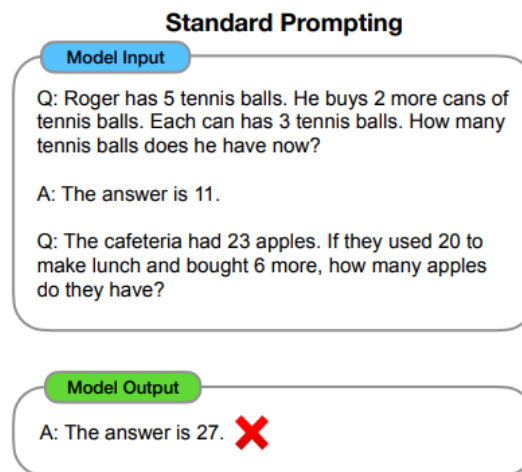
אך בהסתכלות רחבה יותר המודל הוא כשלעצמו אינו אלה מכונת טקסט משוכללת המתחיל ומסיים במילה. בשאלה על הצעד הבא – האם נוכל לחבר למודל כלים כך שיוכל לבצע בהם שימוש? ובמידה וכן, כיצד? האם ניתן להתמיר מילה בפעולה?

התשובה היא חיובית. במרכיב הבא של הסוכן, **כלים חיצוניים**, נדון בהתממשקות המודל לכלים ונדגים מספר שיטות לכך.

2.2 כלים חיצוניים

2.2.1 מוטיבציה

למרות היכולות של המודל לייצר טקסט באופן איכותי, הוא ניחן בתופעות לוואי לא רצויות. אחת מן התופעות הבולטות במודל היא הזיה (hallucination) – תופעה שבה המודל פולט מידע שגוי ואינו "מודע" לכך. לדוגמא, בדצמבר של שנת 2022 כשנשאל מודל ה-ChatGPT של OpenAI מיהו ראש ממשלת ישראל, היה עונה בבטחה נפתלי בנט, למרות שבנימין נתניהו כיהן בתפקיד באותה עת. דוגמא נוספת להזיה הינה תרגיל מילולי במתמטיקה, ראו איור 6 להלן:



איור 6 הלקוח ממאמר [2] הממחיש שגיאה של המודל בחשיבה אריתמטית

שתי הדוגמאות שתוארו לעיל נובעות מסיבות שונות.

- הראשונה – לא היה ברשותו של המודל את המידע העדכני הנדרש על מנת לענות על השאלה בדבר ראש ממשלה מכהן. המודל אומן על מידע מילולי מהאינטרנט עד שנת 2021, ולכן לא ידע על התחלפות ראשי הממשלה.
- השנייה – המודל לא הצליח לבצע את פעולות האריתמטיקה כנדרש ולכן ענה 27 במקום התשובה הנכונה 9, דבר המעיד על העדר יכולות אריתמטיות.

במילים אחרות המודל אינו מסוגל להתמודד עם כל סוגי הבעיות באופן עצמאי וקיים פער ברור בין היכולות של המודל לציפיות שלנו מהביצועים שלו. בדיוק למען גישור פער זה קיים תחום מחקרי רחב באקדמיה ובתעשייה העוסק בהעשרת מודל שפה בכלים חיצוניים על מנת למזער את בעיית ההזיה והמידע החסר.

על מנת לכפר על חולשות המודל נבצע אינטגרציה בינו לבין מודלים פשוטים בהרבה ממנו למשימות ספציפיות בהן הוא לוקה: מחשבון, גישה למאגר מידע עדכני (כמו ויקיפדיה), גישה לשעון (זמן) ועוד.

כעת נדון בפתרון מעניין ופופולרי הקרוי בשם [ToolFormer](#) להתגברות על חולשות המודל בעזרת כלים חיצוניים.

2.2.2 שיפור ביצועי מודל בעזרת כלים – ToolFormer

המאמר [2] מציג את הפרדוקס שבין היכולות הגבוהות של מודל שפה גדול לפתרון בעיות מורכבות לבין החולשה שלו בפתרון בעיות פשוטות וכפתרון משלב את מודל השפה והמודל הפשוט.

המודל המוצע, ToolFormer, משתמש בקריאות API (Application Programming Interface) לשירותים ייעודיים כמו מחשבון, מאגר מידע, לוח שנה (ועוד) במהלך יצירת פלט התשובה ומשלב את התשובות שהתקבלו בפלט. המודל ToolFormer עושה זאת על ידי יצירת סמלים מיוחדים לאזורים בהם דרושה התערבות של כלי חיצוני, ובהיתקלות בסמלים אלה במצע המודל קריאה לשירות חיצוני בתרם ימשיך לחולל את המשך הרצף. הסמלים המיוחדים הם [] ו- > המייצגים את תחילה, סוף ופלט הקריאה בהתאמה.

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

איור 7 מתוך מאמר [2] – דוגמאות לחיזוי של המודל ToolFormer, הטקסט הצבעוני בסוגריים מרובעים אינו נראה למשתמש, אלא נמצא על מנת להורות למודל להשתמש בכלי חיצוני. ניתן לראות את שילוב פלט הכלי משולב בהמשך הרצף.

העובדה שחל שיפור בביצועי מודל השפה כתוצאה משילובו עם כלים חיצוניים אינטואיטיבית בהחלט, בהמשך נראה את ההיגיון המתמטי העומד מאחורי שיטה זו.

עד כה הכרנו את פעולתו של מודל ToolFormer בפשט וכעת נעבור לדון בעומק הארכיטקטורה, תהליכי הכנת הנתונים, האימון וההסקה.

שלב 1: הכנת נתונים

אחת מן ההברקות במאמר [2] נמצאת בשלב הכנת הנתונים והיא הניצול של תכונת הלמידה התוך-הקשרית (in-context learning) של מודל השפה להכנת סט נתוני הלמידה. בהינתן מאגר מידע טקסטואלי X המשמש לאימון של מודל שפה רגיל, נפיק ממנו את המאגר הערוך X^* המכיל את קטעי הטקסט המקוריים בתוספת הסמלים המיוחדים המורים על קריאת API, סוגה ואת הקלט והפלט של הקריאה (דוגמאות באיור 7).

הכנת המאגר X^* מורכבת מהשלבים הבאים:

דגימת קריאות API – מנוסחת שאילתה למודל שפה רגיל בבקשה להכין דוגמאות לטקסט משולב הוראות API בהינתן מספר דוגמאות לאופן שבו המודל נדרש להכיןם. זהו השלב שמאפשר לאמן את מודל ה-ToolFormer מבלי השקעה אדירה של בני אדם בהכנת דוגמאות ע"י ניצול יכולות הלמידה התוך-הקשרית של מודל שפה. בפלט יופיע הטקסט המקורי משולב עם הקריאות API יתכן ריבוי של קריאות עבור סמל יחיד ברצף.

Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

Input: Joe Biden was born in Scranton, Pennsylvania.

Output: Joe Biden was born in [QA("Where was Joe Biden born?")] Scranton, [QA("In which state is Scranton?")] Pennsylvania.

Input: Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

Output: Coca-Cola, or [QA("What other name is Coca-Cola known by?")] Coke, is a carbonated soft drink manufactured by [QA("Who manufactures Coca-Cola?")] the Coca-Cola Company.

Input: x

Output:

איור 8 – דוגמא לשאילתת הכנת דוגמאות לקריאות API, מתוך מאמר [2]

ביצוע הקריאה ל-API

כמשתמע לאחר שנאספו קבוצה של קריאות API עבור כל סמל מתאים, מתבצעת הקריאה אליהם והתוצאה משתלבת בתחום הסמלים המיוחדים לאחר סמל החץ ">" (איור 7).

סינון הקריאות

כעת נמדוד את אפקטיביות התוצאות בהורדת האנטרופיה המשולבת של המשך חיזוי הטקסט, ונסנן את כל הקריאות שלא עומדת בסף סינון τ_f (filtering threshold). עבור רצף תווים נתון נסמן את הקריאה המועמדת לסינון מספר i בתור: $e(c_i, r_i)$ כאשר c_i זוג סדור עם מידע על שם הכלי (מחשבון, תאריך..). והקלט לקריאה, r_i הפלט של הקריאה. M מודל השפה, w_i פרמטרים נלמדים (משקולות) ו- P_M פונקציית הסתברות מותנת.

נשתמש בפונקציית אנטרופיה צולבת ממושקלת הבאה:

$$L_i(\mathbf{z}) = - \sum_{j=i}^n w_{j-i} \cdot \log P_M(x_j | \mathbf{z}, x_{1:j-1})$$

למען חישוב ה- $loss$ במצבים בהם הקריאה מתבצעת והפלט משולבת ברצף, הקריאה לא התבצעה (רק הסמלים של הקריאה מופיעים, ללא הפלט מהקריאה) ובהם הקריאה לא שולבה

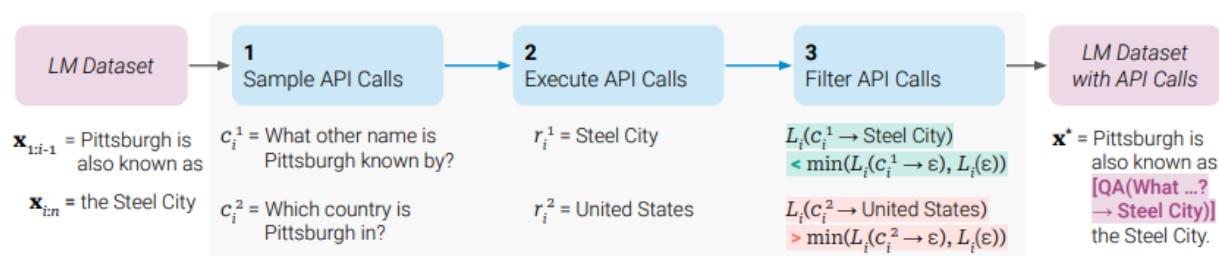
כלל בטקסט. משתמשים במופעים הבאים של פונקציית הloss לקבלת אומדן שיפור חיזוי הרצף ע"י קריאה e :

$$L_i^+ = L_i(e(c_i, r_i)), \quad L_i^- = \min(L_i(\varepsilon), L_i(e(c_i, \varepsilon)))$$

בהינתן ומתקיים התנאי:

$$L_i - L_i^+ \geq \tau_f$$

הקריאה $e(c_i, r_i)$ תוגדר בתור יעילה ותשולב בתוך המאגר כסיפא של רצף התווים הרלוונטיים, ראו איור 9 הבא המסכם את שלב הכנת הנתונים.



איור 9 סיכום שלבי הכנת הנתונים מתוך מאמר [2]

לסיכום, מאגר המידע המתקבל X^* מכיל את המאגר המקורי בתוספת רצפים של קריאות לכלים חיצוניים אשר נבדקו להטיב את איכות הפלט.

שלב 2: האימון (training)

נבצע טיוב המודל (fine tuning) עד לקבלת דיוק רצוי ויתקבל מודל שפה M אשר מחליט באופן עצמאי מתי ואיזה כלי לשלב בתשובה בזמן חילול רצף הפלט. מכיוון שהקריאות במאגר האימון המעודכן נבחרו לפי תרומתם לדיוק חיזוי המשך הרצף, המודל למד (בשאיפה) באיזו עת ברצף הפלט קריאה לכלי חיצוני תשפר את תשובתו.

שלב 3: ההסקה (inference)

בתהליך הסקה המודל M מחולל רצף שגרתי עד לקבלת הסמל המיוחד ">" בו עוצר את פעולתו וימשיך לחולל רק לאחר קבלת תשובה מהכלי החיצוני אליו פנה כאשר פלט הכלי משולב במנגנון הקשב של המודל.

לסיכום, ToolFormer פותר את בעיית ההזיה במודלי שפה, ואף יותר מזה, משפר את הביצועים הכלליים של המודל. בעזרת לימוד מונחה עצמי (self-supervised) עם התערבות מינימלית של בן אדם לומד המודל להשתמש בכלים חיצוניים ולשלב את התוצאה שלהם בפלט שלו, בכך משפר את איכות התוצאה, בפתרון בעיות אריתמטיות ומענה על שאלות המשתמש מבלי לגרוע מיכולות השפה הבסיסית של המודל המקורי.

כעת נתבונן בפרספקטיבה נוספת של כלי חיצוני העוזר בפתרון בעיות סימבוליות, אלגוריתמיות ומתמטיות ע"י המרת הבעיה משפה טבעית לקוד אותו מריץ המודל ומשלב את פלט הריצה בתשובתו לשאילתה.

2.2.3 מפרש קוד – Code Interpreter

ניקח את המודל שלב אחד קדימה בדרך להפיכתו לסוכן – נוסיף לו את היכולת לכתוב ולהריץ קוד ולשלב את פלט ההרצה בתשובתו למשתמש. יכולת זו מקדמת מספר אינטרסים בשיפור יכולות המודל, בהם:

1. שיפור יכולת פתרון הבעיות של המודל
2. יצירת כלים ייעודיים לשימוש עתידי חוזר
3. התממשקות עם כלים חיצוניים שלא אומן עליהם מראש

מעבר לקידום יכולות המודל לתת מענה איכותי לשאילתת המשתמש, פעולות אלה מפתחות את ה**עצמאות** של המודל לבצע משימות מורכבות יותר ללא הסתמך על הנחיה מפורשת מהמשתמש.

לפני שנדגים את מפרשי הקוד נציג רעיון שעוזר למנף את יכולות מודל השפה באמצעות שאילתה שמורכבת מדוגמאות של פתרונות לבעיות דומות לאחת הנתונה ושמו few shot prompting.

הנחיה מעושרת-דוגמאות – Few shot prompting

בשיטת הנחיה זו מציגים למודל מספר קטן של דוגמאות קלט-פלט בתוך השאילתה המטרה להכווין את המודל לפלט הרצוי ללא אימון מחדש או כוונן עדין (fine tuning) שלו. השיטה מאפשרת למודל למנף את יכולותיו החבויות בין מיליארדי הפרמטרים של שנלמדו בשלב האימון.

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✓

איור 10 – הנחיה מעושרת דוגמאות כאשר המקטע המסומן בכחול הינה הדגמה לאופן החשיבה הנדרש מהמודל (מתוך מאמר [4])

השוו את איור 10 לאיור 6 מוקדם יותר בסמינר, ניתן לראות שמודל השפה הצליח לפתור בעיה אריתמטית לאחר שכשל בה כאשר סופקה לו הנחיה נטולת תיאור אופי החשיבה הנדרש. שיטה זו נקראת שרשרת מחשבה (CoT – Chain of Thought) והצלחתה ידועה בתחום מינוף יכולות המודל, כפי שהצליחה להשוות מודלים לביצועי state of the art בזמן הכתיבה (2023).

שיטות רבות המשלבות הנחיה מעושרת התפרסמו בעקבות המאמר [4] לעיל, באחת מהן נדון מיד ובזאת נחזור לנושא הנוכחי "מפרש קוד".

במאמר [3] מוצגת שיטת המודל הנתמך תכנות (PAL – Programming Aided LM), אשר בהינתן הנחיה מעושרת-דוגמאות כותבת את שלבי הביניים לפתרון הבעיה בתור קוד פייתון. לאחר פריסת כל שלבי הפתרון מריץ המודל את המפרש ומציגה למשתמש את פלט התוכנית בתור התשובה הסופית.

כמעט ולא תיתכן טעות חישובית עבור קוד שנכתב בצורה תקינה (למעט זליגת זיכרון או בעיות בזמן ריצה), לכן בהנחה ש-PAL הפיק שלבי ביניים נכונים לפתרון הבעיה תשובת המפרש להנחיה גם כן תהיה נכונה. לא ניתן לטעון כך עבור מודל CoT כיוון שאפילו אם פירק את הבעיה באופן תקין, המנוע המילולי של מודל השפה עלול לטעות בחיזוי התשובה הנכונה – הוא אינו מדויק כמו מפרש שפות התכנות.

Chain-of-Thought (Wei et al., 2022)

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold $93 + 39 = 132$ loaves. The grocery store returned 6 loaves. So they had $200 - 132 - 6 = 62$ loaves left. The answer is 62.



Program-aided Language models (this work)

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls.

`tennis_balls = 5`

2 cans of 3 tennis balls each is

`bought_balls = 2 * 3`

tennis balls. The answer is

`answer = tennis_balls + bought_balls`

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves

`loaves_baked = 200`

They sold 93 in the morning and 39 in the afternoon

`loaves_sold_morning = 93`

`loaves_sold_afternoon = 39`

The grocery store returned 6 loaves.

`loaves_returned = 6`

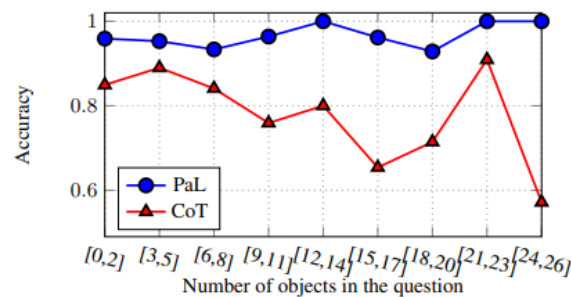
The answer is

`answer = loaves_baked - loaves_sold_morning - loaves_sold_afternoon + loaves_returned`

`>>> print(answer)`
74



איור 11 – PAL פותר בעיה בהצלחה לעומת CoT הנכשל בה (מתוך [4])



איור 12 – ביצועי PAL גוברים על CoT עבור בעיות במורכבות הולכת וגוברת באמת מידה (benchmark) של זיהוי אובייקטים צבעוניים ומענה על שאלות הסקה לגביהם (מתוך [4])

מאיורים 11 ו-12 נסיק שיכולת חישובית של מודל שפה משתפרת משמעותית כאשר משלבים שרשרת מחשבה CoT עם מודל נתמך תכנות PAL. הרי הראשון תומך בהבנת ההנחיה והשני במיקור חוץ של פתרונה.

שילוב זה אינו אלה הרמוניה הנובעת מעוצמת מודל השפה להבין כוונות והקשר תוך-טקסטואלי וכתוצאה פירוק הנחיה לתת-בעיות, והיכולת של שפות תכנות לפתור בעיות סימבוליות (אריתמטיקה, הגיון ואלגוריתמיקה).

סיכום ביניים

בחלק זה הצגנו את הכלים בהם נעזר מודל שפה גדול למען שיפור דיוק הפלט, ובכך השתפר בפתרון בעיות שבהן דיוק אריתמטי ולוגי חשובות. מודל השפה נלקח צעד אחד קדימה המסע הפיכתו לסוכן במיוחד כעת כשבעיית ההזיה פחתה משמעותית.

כעת נדון בהעמקת יכולת החשיבה של המודל ובכך נשפר עוד יותר את האוטונומיות שלו במענה על הנחיות מורכבות יותר.

2.3 חשיבה עמוקה

עד למשפט זה בסמינר מודל השפה שלנו צבר יכולות איכותיות ששיפרו את ביצועיו משמעותית ביחס למודל השפה נטול הכלים. בעזרת מודלי עזר פשוטים ([כלים חיצוניים](#)) הוגברה יכולת הדיוק שלו לצד צמצום בעיית ההזיה ועל ידי [מפרש הקוד](#) פותר בעיות סימבוליות טוב יותר. כלומר, עד כה נוספו למודל השפה יכולות חיצוניות – ובהחלט התוצאות מרשימות: המודל כעת מסוגל לא רק לפלוט טקסט איכותי ומדויק, אלה גם לבצע פעולות בשמנו.

אך היכולות שאספנו עד כה, עם כל הכבוד, אינן מספיקות בכדי לשמש לסוכן עצמאי. המודל כעת לא מסוגל לבצע את המשימה שהגדרנו לחלק המעשי של סמינר זה: [פיתוח מערכת מסחר אלאגוריתמי במטבעות קריפטוגרפים](#), הטומנת בתוכה מורכבות רבה וידע במגוון תחומים כמו ידע בטכניקות מסחר, עיצוב מוצר, ארכיטקטורת web, פיתוח הקוד ותיעודו ועוד.

כעת נרצה להעשיר את המודל ביכולות עצמאיות, והעיקרית שבהן היא יכולת החשיבה העמוקה. בחלק זה נעסוק ביכולת מודל השפה לפרק הנחיה מורכבת של המשתמש ל**מחשבות** ביניים כך שכל מחשבה תורמת לדיוק הפלט הסופי. נציג התפתחות היסטורית קצרה של גישות לחשיבה עמוקה ואת תרומתן לעצמאות הכללית של המודל (ובהמשך, הסוכן).

להלן מספר גישות שנבנו אחת מעל השנייה וכל אחת בתורה קידמה את יכולת החשיבה העמוקה של מודל השפה.

2.3.1 שרשרת מחשבה (Chain of Thought – CoT)

כפי שהוצגה במקטע [few shot prompting](#) היא טכניקה לכוון את מודל השפה לפרק את החשיבה לשלבים בהינתן דוגמאות מוליכות בהנחיה.

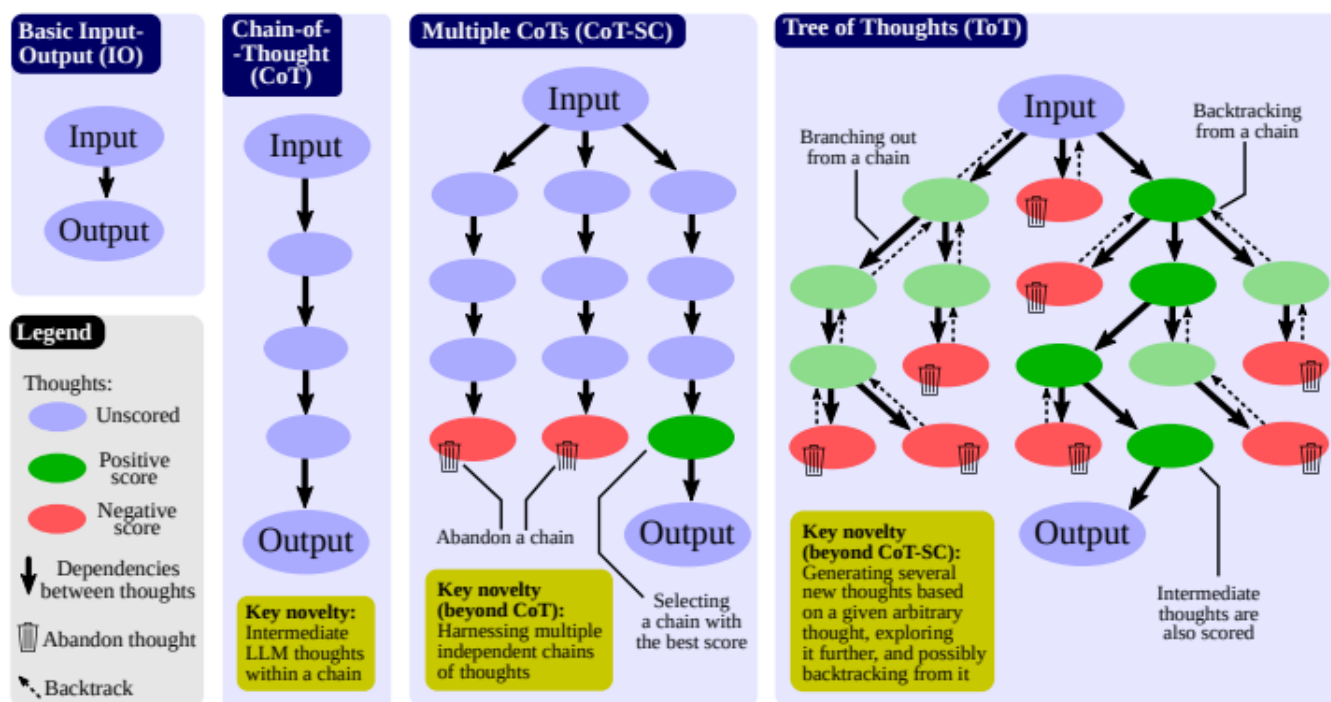
מעל טכניקה נבנתה שיטה בשם שרשרת מחשבה עם עקביות עצמית (CoT-SC (Chain of Thought Self Consistency) המחוללת מספר שרשראות מחשבה כמו בCoT ובפלט נבחרת התשובה התדירה ביותר. ההצלחה שלה על פני שרשרת מחשבה בודדת מיוחסת לעובדה שלבעיות חשיבה קיימות מספר דרכים לפתרון והפתרון הוא זהה בכל הדרכים השונות, לכן הפתרון הנפוץ ביותר בין כל שרשראות המחשבה הוא הסביר להיות נכון.

לא נפרט על ארכיטקטורת הפתרון של שיטות אלו, ובמקום נמקד את ההרחבה על שני השיטות המודרניות הבאות: עץ מחשבה וגרף מחשבה.

2.3.2 עץ מחשבה (Tree of Thoughts – ToT)

במאמר [5] מתווכחים החוקרים שהשיטות הקודמות לעידוד חשיבה רב-שלבית של מודל שפה לא תואמת את החשיבה האנושית. יכולת החשיבה העמוקה של מודל שפה סטנדרטי (או שימוש בהנחיות CoT) מוגבלת על ידי האופן הרציף בו הוא מחולל את הפלט – סמל אחר סמל שהיא שרשרת הסתברויות סדרתית, בעוד שהחשיבה האנושית דומה יותר לחיפוש במרחב בעיה קומבינטורית כלומר חיפוש בעץ בו הצמתים מייצגים פתרונות חלקיים והקשתות פעולות המשנות את הפתרון החלקי בכיוון הפתרון.

הפתרון המוצע במאמר הוא אלגוריתם חיפוש בעץ, כאשר החזקה שלו מתבטאת באופן חילול "מחשבות ביניים" (הצמתים) והערכתם ע"י מודל השפה עצמו. בכל שלב נבחרים עד t צמתים (במאמר $t \leq 3$) שהוערכו בציונים הגבוהים ביותר על ידי המודל ואליהם מחבר האלגוריתם עד k מחשבות שונות (במאמר $k \leq 5$) אשר חילל מודל השפה, וכך חוזר חלילה עד לפלט.



איור 13 – השוואה בין פרדיגמות חשיבה עמוקה [6]

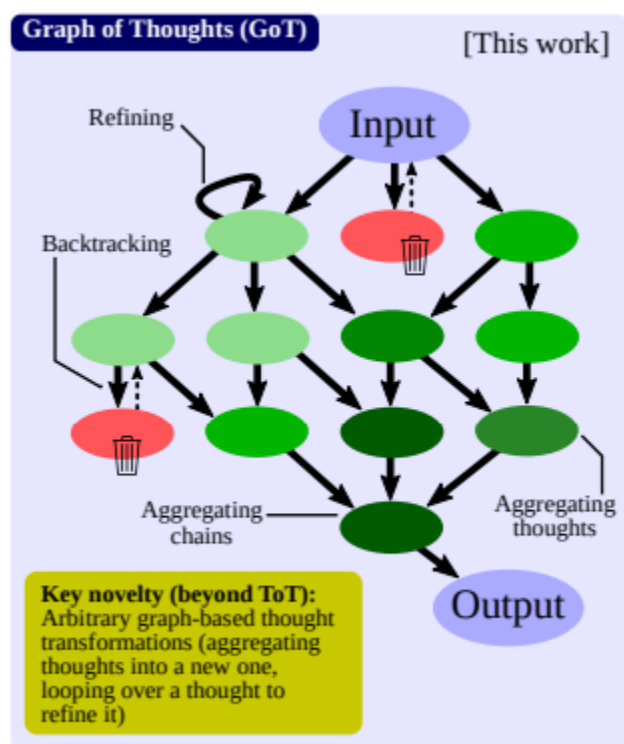
שיטה זו חזקה מקודמותיה בשני אספקטים:

1. השיטות הקודמות (בעיקר CoT-SC) לא מחברות מחשבות בין ענפים שונים בעץ ובכך מאבדות פוטנציאל לשלב איכויות נבדלות משני שרשראות מחשבה שונות.
2. ToT מקיימת אלמנט של תכנון, חזרה לאחור (*backtracking*), הערכה וסינון מחשבות ועל ידי כך מאפשרת חקירה מיטבית של מרחב הפתרונות.

חשוב לציין כי בשונה מקודמותיה, שיטה זו משתמש במודל השפה בתור מודול (module) באלגוריתם כאשר הפניות אליו מתבצעות מתוך אחת הדרישות: חילול מחשבה והערכת מחשבה. עץ המחשבה נבנה ונשמר בזיכרון חיצוני למודל על ידי ריצה של קוד האלגוריתם.

2.3.3 גרף מחשבה (Graph of Thought- GoT)

כעת נציג גישה חזקה יותר בשם "גרף מחשבה". החוקרים שעומדים מאחורי פיתוחה מתווכחים על עליונות הביצועית שלה הנובעת מתוך מודל גרפי דינמי שרירותי הדומה ביותר לאופן החשיבה האנושית, על ידי הסרת המגבלות ההדוקות של מבנה העץ או השרשרת. גישת גרף המחשבה נבנית ישירות מעל הגישות שהוזכרו לעיל (*CoT*, *CoT-SC*, *ToT*) ומרחיבה אותם ע"י הוספת גמישות לתהליך התפתחות החשיבה העמוקה וכתוצאה משפרת את ביצועי המודל תוך שימוש מופחת במשאבים – מאפשרת דפוסי חשיבה מורכבים יותר.



איור 14 – תיאור חזותי של GoT, בהמשך לאיור 13 [6]

החידושים של גישת הגרף על פני עצים מתבטאת במספר אופנים חשובים המקרבים אותה לעליונות ביצועית. תחילה נשים לב שהגרף מאפשר לקשתות לחבר צומת לעצמו ולכן מאפשרת השבחה של מחשבות קיימות. שנית, מחשבות שהובילו לעלה (מחשבה סופית/פלט) שגוי עלולות לתרום לרצף מחשבות אחר שבו לא השתתפה כלומר שרשראות מחשבה נבדלות משתפות מחשבות ובכך מועשר מרחב הפתרונות. לבסוף, מתנקזות כל שרשראות המחשבה לעלה בודד וסופי, המרכז את כל המחשבות התורמות ומזקק אותן לכדי פתרון סופי יחיד.

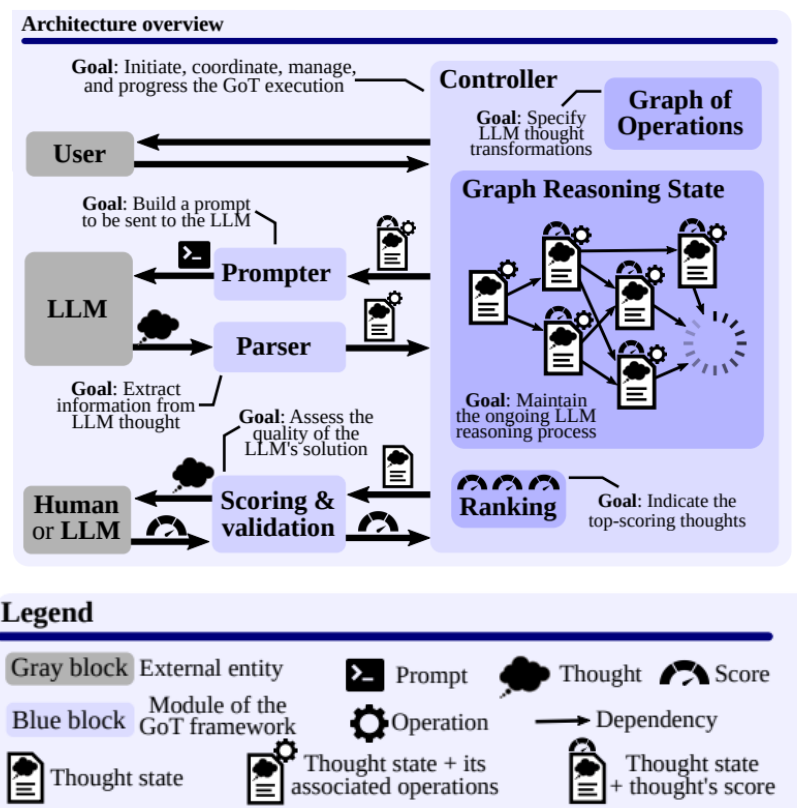
ארכיטקטורת המודל משלבת מספר מרכיבים שפעולתם המשותפת ממנפת את יכולות מודל השפה. כמו בשיטת העץ, גם

הגרף משתמש במודל השפה בתור רכיב חיצוני ומבצע אליו קריאות במהלך זמן הריצה.

מרכיבי הארכיטקטורה הם:

1. Controller – מתזמר (orchestrates) את תהליך החשיבה. עוקב אחר גרף הפעולות (GoO), בורר אילו מחשבות לקדם, בוחר באופן הפעולה (האם לייצר מחשבה חדשה, לרכז מספר מחשבות, לשפר מחשבה וכו'), מחליט האם לעצור את הפעולה של גרף

- המחשבות בהגעתו למחשבה סופית מספקת או להמשיך. מטרתו לקדם את גרף המחשבות לכיוון הפתרון באופן יעיל.
2. גרף הפעולות (Graph of Operations – GoO) – מייצג את הליך המחשבה הכללי חד פעמית וסמטי מתחילת ריצת מודל הגרף. מגדיר את אופן פירוק בעיה לגורמים ואיחודם למען הפתרון.
3. Prompter – מייצר הנחיות למודל השפה בהתאם למבנה הגרף הנוכחי באמצעות שורת פקודות. באחריותו להנחות את מודל השפה לייצר מחשבות חדשות, לשפר מחשבות קיימות, לוודא תקינות או לרכז מספר מחשבות.
4. Parser – מחלץ מידע מבני מהפלט הטקסטואלי של מודל השפה ומייצר מצב של מחשבה. מעדכן את מצב הנמקת הגרף (Graph Reasoning State – GRS).
5. דירוג ואימות (Scoring and Validation) – מעריך את הנכונות והאיכות של כל מחשבה בעזרת מספר כלים שונים כמו פונקציה מקומית שהוגדרה מראש, היעזרות במודל השפה או בן אדם. תוצאת דירוג המחשבה תתמוך בהחלטות הController לגבי חידוד או הזנחה של מחשבות קיימות.
6. מצב הנמקת הגרף GRS – מתחזק מצב דינמי של החשיבה (או הנמקה). מייצג במבנה נתונים את כל המחשבות שחוללו עם דירוגן ואיכויותיהן ובנוסף עוקב אחר משימות גרף פעולות GoO שבוצעו. מציג את ההקשר הנוכחי לביצוע החלטות מודל הגרף.



סיכום ביניים

הצגנו מספר גישות הממנפות את מודל השפה לפתרון מגוון רחב של בעיות בתחומים שונים וביניהם כתיבה יצירתית, בעיות הגיון, אריתמטיקה אלגוריתמיקה ועוד. עובדה אחת הבולטת בהתפתחויות הללו היא שלצד התפתחות פתרונות חשיבה עמוקה עובר בהדרגה מודל השפה לתפקד בתור רכיב חיצוני במערכת גדולה יותר. בCoT מודל השפה היה הרכיב היחיד בפתרון אך בזמן שהתפרסם מאמר הGoT כבר בבירור סוכם הקונצנזוס האקדמי שמודל השפה בתור חבילת תוכנה נפרדת המשולבת בפתרון עם ארכיטקטורה מורכבת מניב תוצאות איכותיות יותר.

נוסף על ההיפרדות הלוגית – יכולות חילול המחשבה, הסקת מסקנות (התמרת מחשבה), דירוג מחשבה והערכת איכות מתקבלות בולטות בתור בתחומי האחריות של מודל השפה.

וכעת הגענו לפסגת ביניים של הסמינר. דיברנו על מודל השפה הגדול ואופן פעולתו, חיברנו למודל כלים חיצוניים בהם מבצע שימוש לשיפור תוצאות והרחבת יכולות ולבסוף על יכולות המחשבה העמוקה שלו – שלושת התנאים ההכרחיים לבנייתו של **סוכן מודל שפה**.

עתה כשברשותנו אחיזה והבנה במונח הסוכן, נדון במערכות מרובות סוכנים ונשאל ונענה על שאלת התועלת בריבוי סוכנים בתוך מערכת ייעודית לפתרון בעיות מורכבות.

2.4 מערכת מרובת סוכנים (Multi-Agent System)

2.4.1 מוטיבציה

מאז שחר ההיסטוריה תועדו בני אדם ובעלי חיים אחרים שמשתפים פעולה למען השגת יעד גדול מכדי שמשותף יחיד, בין אם זו קבוצת לוחמים של שבט ציידים-לקטים בצייד אחר ממוטה או קבוצת מהנדסי תוכנה ומנהלי מוצר משחררים פיצ'ר חדש במוצר של החברה. הדבורים, המאוימות על ידי גוף זר להשמדת מתאחדות לנחיל ויוצאות למתקפה משותפת להגנת הקן. מאותן סיבות בדיוק קמה המוטיבציה לקבץ מספר סוכנים לכדי מערכת אחת ובכך למנוף את התפוקה המשותפת שלהם.

סוכן מודל שפה גדול בעל יכולות מגוונות ורבות, אך לעיתים תכונה רצויה יותר היא מקצועיות מרבית בתחום צר וספציפי. כל מודל שפה יכול לכתוב קוד, שירה ולפצח אתגר אריתמטי באותה הנחיה יחידה – אך לרוב אין בכך צורך. בעולם האמיתי, נעדיף שהסוכן אחד יהיה הכי טוב בכתיבת קוד, השני בארכיטקטורת מוצר והשלישי בחשיבה עסקית. שיתוף פעולה בין מומחים שכאלה למען מטרה משותפת אחת היא שם המשחק וזוהי המוטיבציה הראשונה מאחורי מערכות מרובות סוכנים – **מומחיות**.

מוטיבציה נוספת הנובעת כמעט מיידיית מהפרדת סוכני מומחיות הינה **מדרגיות (scalability)**, הרי כאשר קיים מומחה לתחום מסוים הוא יכול לספק שירות לכל "לקוח" שזקוק בו, בדומה ליועצים בעולם הקפיטליסטי המודרני.

גיוון מחשבתי הנובע משיתוף פעולה של מומחים הוא עוד מוטיבציה מתבקשת במערכות מרובות סוכנים.

2.4.2 ארכיטקטורות

ישנן שלוש גישות מרכזיות בתכנון ארכיטקטורות של מערכות סוכנים: שליטה מרכזית (centralized), שליטה מבוזרת (peer-to-peer) ומודל היברידי. להלן תיאור ויתרונות וחסרונות מרכזיים של כל גישה:

1. שליטה מרכזית – בקר ראשי (תוכנה, מודל שפה או אדם) שמנהל את תהליך המחשבה על ידי פירוק המשימה הראשית לתת משימות וחלוקן לסוכנים המתאימים במערכת. יתרון הוא תהליך העבודה ממוקד ומאורגן, סוכן ייעודי לניהול התהליך. החסרונות שסוכן הניהול גובה משאבים נוספים ועלול להגביל את מרחב הדיון בעקבות פיקוח הדוק.

2. שליטה מבוזרת – הסוכנים מתקשרים בינם לבין עצמם ללא בקר מרכזי. היתרון הוא שסגנון זה מדמה קבוצת מומחים המתדיינת בנושא באופן פתוח ומגוון ובכך מאפשר כיסוי רחב יותר של מרחב הבעיה. החיסרון הוא אתגר ניהול המשימות בעקבות העדר סוכן ניהולי.

3. היברידית – שילוב של שני הגישות לעיל, מכפרת על החסרונות של שניהם במחיר של מורכבות ארכיטקטונית.

עד כאן התדיינו לעומק במרכיבים של הסוכן – [מודל שפה גדול](#), [כלים חיצוניים](#) ו[חשיבה עמוקה](#). בנוסף ציינו את המוטיבציה ומספר ארכיטקטורות למערכות מרובות סוכנים וכעת ברשותנו ההבנה הנדרשת על מנת להמשיך לחלק המעשי בו נהנדס ונפעיל מערכת מרובת סוכנים לבניית תוכנה הסוחרת במטבעות קריפטוגרפים על ידי הפעלת אלגוריתמי סחר. נדון בתהליך יצירת הסוכנים והגדרתם, נפעיל אותם ונעריך את איכות התוצאה הסופית.

3. חלק מעשי: הדגמה – פיתוח מערכת מסחר אלגוריתמי במטבעות קריפטוגרפים באמצעות מערכת מרבית סוכנים

3.1 מבוא

מטרתו של הפרויקט הייתה התנסות בהפעלה של מערכת מרבית סוכנים לבניית מערכת תוכנה מורכבת – מערכת למסחר אלגוריתמי במטבעות קריפטוגרפים ואת הכלים החזותיים להצגת נתונים כמו שער מטבעות, סטטיסטיקות, התממשקות מול API צד שלישי ופיתוח אלגוריתם לקבל החלטות רכישה ומכירה. בפועל, ניסיונות רבים בהפעלת מערכות מרבית סוכנים (MetaGPT – open source multi-agent framework, MGX – commercial multi-agent system) נכשלו בבניית המערכת. לאחר התנסות במספר גישות וכלים שונים, נבנה רק חלק מתוך המערכת השלמה.

בחלק זה אציג את המערכת הבנויה ואת התהליך שהוביל לבנייתה, את הכלים השונים שהשתמשתי, את הגישות מאחוריהן, התוצאות והמסקנות.

3.2 המערכת AlgoTrader

מטרת המערכת היא לסמן לסוחרים נקודות רכישה ומכירה חזקות על מנת לייעל (או להחליף) את הסחר האנושי במטבעות. למעשה, בהינתן שינוי קטן בקוד והוספת גישה למשתמש binance, המערכת תבצע פעולות מסחר עם ערך מטבע אמיתי המוחזקים בארנק של המשתמש.

המערכת בנויה מהמרכיבים הבאים:

צד השרת (server side)

1. התממשקות עם Websocket API של פלטפורמת המסחר Binance.com ובכך מתאפשרת קבלת נתוני מסחר בזמן אמת ונתונים היסטוריים
2. מנוע קבלת החלטות לרכישה ומכירה, אסטרטגיה קלאסית EMA Crossover
3. התממשקות זרימת נתונים בזמן אמת (socket) עם צד הלקוח

צד לוקח (client side)

1. עמוד המציג את נתוני המסחר, גרפים של מחיר המטבע והממוצעים הנעים ואינדיקטורים על נפח מסחר, כמות עסקאות ועוד

2. התחברות לשרת באמצעות sockets



איור 16 – צילום מסך של ממשק מערכת המסחר. מוצגים גרפים של מטבעות BTC ו-ETH, עם נתונים המחיר בשוק, ממוצעים נעים באורך 21 ו-55 העומדים בבסיס ההחלטות של אסטרטגיית המסחר בשם EMA Crossover וחצים ירוקים ואדומים המסמלים הוראות קנייה ומכירה בהתאמה.

הקוד ותיעודו נכתבו במלואם על ידי סוכני מודל שפה גדולה בהינתן הנחיות והגדרות. הקוד זמין לצפייה הורדה והרצה באתר הגיט האב שלי – <https://github.com/boriboy/algotrader>.

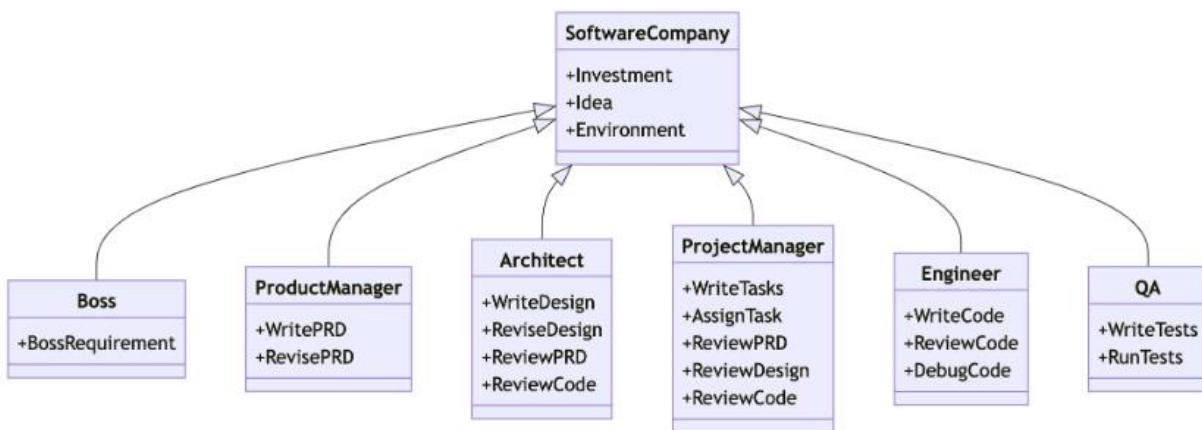
3.3 תהליך הפיתוח וגישות שונות למערכות סוכנים

כפי שצוין לעיל החזון למערכת המקורית התכווץ בעקבות מספר ניסיונות כושלים של הפעלתם של סביבות מרובות סוכנים, כאשר אותה מערכת הוגדרה להיות גדולה יותר בנפח ובמורכבות ההנדסית, ביניהם ריבוי מספר אסטרטגיות מסחר שונות ומנגנון הצבעה שביחד ממנפים את רווחיות המסחר, שליטת משתמש מוגברת על פעולות המערכת והתממשקות מלאה למשתמש סוחר ב-Binance על ידי ביצוע עסקאות מסחר.

למרות שלא התממש הרעיון המקורי, ההתנסות עם הכלים השונים לימדו אותי לא מעט על עבודה מול סוכני פיתוח תוכנה ואת הידע אסכם בסעיף זה.

ניסיון 1 – מערכת מרובת סוכנים קוד פתוח MetaGPT

בניסיון הראשון השתמשתי בכלי קוד-פתוח בשם [MetaGPT](#) שמפותח ומתוחזק על ידי מספר חוקרי מודל שפה ולמידה עמוקה, ביחד הם יצרו מערכת שבה ניתן הנחיה עם פירוט למערכת רצויה מייצרת אותה עם תוכנה מלאה, מסמכי אפיון, ארכיטקטורה ואומנות דיגיטלית. העבודה קוראת מאחורי הקלעים על ידי מספר סוכני מודל שפה גדול מוגדרים מראש לתפקידים שכיחים בתעשיית ההיי-טק: מנהל מוצר, ארכיטקט, מנהל טכנולוגיות, מהנדס תוכנה ומהנדס אבטחת איכות. למעשה, כל ישות מתפקדת בתור נותן שירות שבבסיסו מודל שפה, כל הישויות יחד מנוהלות על ידי אלגוריתם איטרטיבי של יצירת משימות והשלמתן בתורות.



איור 17 – סכמת חברת פיתוח תוכנה, לפי הפילוסופיה של MetaGPT (מתוך קוד המקור)

לאחר הגדרת מפתח API של OpenAI והגבלת תקציב נדיבה של \$12 דולר אמריקאי ההנחיה שנמסרה למערכת היא זו:

Create an algorithmic trading software that uses Binance API to trade crypto currencies. The algorithmic trader should detect good buying and selling points aiming to make profit, using advanced trading strategies. Implement a simple web dashboard providing live insights on market data (drawn from binance via socket) and the algorithm's decision making. Use the binance's testnet sandbox environment for committing market instructions. Provide profit/loss stats on the dashboard, displaying the accumulated losses and profits along with other useful information on the algorithm's performance.

לאחר מספר דקות של ריצה פלטה המערכת קובץ הגדרת מוצר וכבתה. כל הניסיונות להמשיך את הריצה או להתחילה כך שתסיים את עבודתה כשלו, ועם חוב של \$4 המשכתי לניסיון הבא.

ניסיון 2 – מערכת מרובת סוכנים מסחרית MGX

מערכת זו מבוססת על תשתית של הספרייה MetaGPT שאוזכרה לעיל ופועלת בענן בהפעלת הסוכנים דרך ממשק web ידידותי. בזמן הרצתה באותה הנחיה שצוטטה לעיל, עבדה המערכת מספר דקות עד שהופיעה הודעה על סיום מוחלט של בניית המערכת. כשעברתי על הקבצים התברר שחלק מן התיקיות ריקות ולא כל הקוד כתוב עד הסוף, מה שהפך את התוצר לבלתי ניתן להרצה.

בשלב הזה חלה ההבנה כי ייתכן אחד משני דברים: טכנולוגיות מרובות סוכנים לוקות בשיטות ניהול פנימי או ההנחיה שניתנה לה מורכבת מדי. לפשט את ההנחיה זה לא בא בחשבון מפני שמרכז ההתעניינות בסמינר זה הוא יכולת הביצוע של מערכת סוכנים שכזו, לכן הניסיון הבא בפיתוח המערכת הרצויה כלל התערבות אנושית יותר אינטנסיבית.

ניסיון 3 – מערכת היברידית סוכנים ובן אנוש מפקח

בארכיטקטורות של מערכות מרובות סוכנים צוינה אפשרות היברידית בה קיים שילוב של ישות מנחה (סוכן או בן אדם) קבוצת סוכנים לצד קבוצה עצמאית. בסעיף זה חלה התערבות אנושית יותר משמעותית מאשר כתיבת ההנחיה ולחיצה על מקש ה"אנטר", **נלקחה האוטונומיה** של קבוצת הסוכנים.

כעת כאשר בן אדם הוא מנהל העבודה של הסוכנים, הוסרה האפשרות לתקלות סביב יכולת ניהול המשימות העצמי של קבוצת הסוכנים.

לעזרתי יצרתי 2 סוכנים מומחים המייעצים בתחום המומחיות שלהם:

1. מומחה פיננסי – נוצר בעזרת המודל GPT-5 ותפקידו לבצע מחקר ממקורות אינטרנט בנוגע לשיטות (אסטרטגיות) מסחר במטבעות קריפטוגרפים ויצירת מסמך הגדרות שמיועד לקהל הטכני באופן שכל מהנדס תוכנה יוכל לממש את הפונקציונליות של האסטרטגיה מקריאה במסמך
2. מומחה להתממשקות תוכנה עם Binance – גם מבוסס על מודל GPT-5, הונחה לכתיבת קוד התממשקות (WebSocket, API) עם פלטפורמת המסחר על ידי שימוש במקבץ קבצי תיעוד מספרית התיעוד הרשמית של Binance שצורפו לו במאגר המידע

פיזור תחומי האחריות בין מספר סוכנים התגלתה כשיטה יעילה משמעותית יותר מהגדרת סוכן יחיד בעל תחום אחריות רחב, בכך התאפשר לכל מודל שפה לגלם תפקיד מוגדר היטב בתחום צר בעל עומק הבנה ואיכות ביצועית.

ניתן לצפות בתוצרים של הסוכנים המומחים בתיקיית המסמכים של הפרויקט [algotrader/docs](https://github.com/algotrader/docs).

תוצרי הקוד של הסוכן המומחה להתממשקות עם Binance הורצו בהצלחה מהניסיון הראשון והם יהוו מסמכי בסיס למערכת הגדולה יותר – אותה יפתח סוכן מודל שפה הידוע בהתמחותו בכתיבת קוד, המודל Claude Sonnet 4. לאחר מתן גישת קריאה וכתיבה לתיקיית הפרויקט המקומית וסדרה של הנחיות לפיתוח ותיקוני באגים, יצר המודל את המערכת לעיל תוך כשעתיים של עבודה משותפת.

להלן רשימה חלקית של הנחיות שניתנו למודל:

1. fix the dashboard graphs to be less jumpy and more similar to those appearing on trading platforms.
2. when app is initiated, collect previous 50 minutes of symbol data for the ema crossover strategy to begin executing right away.
3. add ema information on top of the existing graphs.

Great. More changes:

1. fix the candlestick section in the graphs, it's all messed up.
2. Whenever open_long() or close_position() are evoked, display the decision at the appropriate place on the line graph as an arrow above the price graph.

איור 18 – דוגמאות להנחיות למודל Claude Sonnet 4 בבניית AlgoTrader

4. סיכום ומסקנות

טכנולוגית מודל השפה הגדול, העומדת בבסיסו של סוכן, עברה התפתחויות עצומות בשני העשורים האחרונים וזאת כתוצאה ממחקר אקדמאי בתחום הNLP, התעצמות הכוח החישובי ושפע של מידע הזמין באינטרנט. מודל השפה מתממשק עם כלים חיצוניים ומשתמש בטכניקות חשיבה עמוקה ובכך משתפרת משמעותית יכולת פתרון בעיות ומתווספת מסוגלות לתפקוד אוטונומי – לתוצר זה קוראים סוכן. הסוכן כשלעצמו שימושי במגוון רחב של תחומים ביניהם התכתבות צ'אט (בידור, שירות לקוחות, עוזר אישי מנהל יומן ועוד), כתיבת קוד הרצתו ופענוח תקלות, תרגום בין שפות, כתיבה יצירתית, סיכום טקסט וחיפוש באינטרנט.

מערכות מרובות סוכנים מספקות הרחבה ליכולות הסוכן על ידי מינוף הפרדת תחומי אחריות והשרשת פרוטוקול תקשורת בין הסוכנים וניהול משימותיהם.

החלק המעשי היווה בחלקו בדיקה לרמת האוטונומיה הקיימת בכלים הפופולריים כיום בבניית מערכות תוכנה מורכבות. נכון לזמן כתיבת סמינר זה, רבים הפרסומים בפורומים ורשתות חברתיות באינטרנט המעידים על חשש אוכלוסיות העולם המתקדם מפני אבדן עבודתם לבינה מלאכותית. אמנם סמינר זה אינו עוסק בנושא החשש – חשוב לציין כי הפעלה "חסרת חשיבה" (prompt-to-complex software system) של סוכני מודל שפה והציפיה לקבל מוצר תוכנה מורכב אינה מציאותית. ייתכן כי מערכות פשוטות יותר אשר בנייתן תועדה מספר רב של פעמים במאגרי קוד פומבי תוך שימוש בספריות מוכרות ומתועדות היטב תיהנה קלות יותר להתמודדות עבור מערכות מרובות סוכנים, אך לייצר מוצר מורכב ויצירתי הינה משימה לא טריוויאלית.

סוכנים מתמודדים היטב עם משימות בהיקף קטן, כאשר יחד עם הכוונתו של האדם בהחלט ניתן להגיע לתוצר איכותי. שימוש בסוכנים חוסך לאדם את הזמן המושקע בפתרון בעיות טכניות מצומצמות כמו כתיבת פונקציות קוד ואפילו מודולים שלמים, ביצוע מחקר בנושא ממוקד באינטרנט וסיכומו ועוד. כלומר, ניתן לראות את סוכן מודל השפה בתור כלי שמרחיב את היכולת והתפוקה של האדם.

מקורות

1. Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).
2. Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. NeurIPS, 2024.
3. Gao, Luyu, et al. "Pal: Program-aided language models." *International Conference on Machine Learning*. PMLR, 2023.
4. Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. NeurIPS, 2022.
5. Yao, Shunyu, et al. "Tree of thoughts: Deliberate problem solving with large language models." *Advances in neural information processing systems* 36 (2023): 11809-11822.
6. Besta, Maciej, et al. "Graph of thoughts: Solving elaborate problems with large language models." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 38. No. 16. 2024.