המכללה האקדמית להנדסה ע"ש סמי שמעון

SW reuse with LLM'S

פרויקט הנדסי

דו"ח מסכם

הוכן לשם השלמת הדרישות לקבלת
תואר ראשון בהנדסה B.Sc.

מאת

שם הסטודנט: ליאור גופמן מספר זהות: 208212191

שם הסטודנט: ניר קנימח מספר זהות: 209375906

בהנחיית ד"ר מרינה ליטבק

הוגש למחלקה להנדסת תוכנה
המכללה האקדמית להנדסה סמי שמעון
באר שבע

תאריך עברי
כ"ד בסיון תשפ"ה

תאריך לועזי
20.6.25

הבעת תודה

ברצוננו להביע את תודתנו העמוקה לד"ר מרינה ליבטק, על ההנחיה הצמודה, הליווי המקצועי, וההשקעה הרבה לאורך כל שלבי הפרויקט. ההכוונה שלך, התמיכה הרציפה והנכונות ללוות אותנו גם ברגעים של אתגר וחוסר ודאות הפכו את התהליך כולו לחוויה לימודית משמעותית עבורנו.

הגישה הסבלנית, הדייקנות, והיכולת שלך לראות את התמונה הרחבה לצד הפרטים הקטנים, אפשרו לנו להתקדם בביטחון ולשאוף למצוינות אמיתית. בזכותך הצלחנו לא רק לעמוד ביעדי הפרויקט, אלא גם להעמיק את ההבנה שלנו בתחום, ולחדד את הכלים המחקריים והמעשיים שלנו.

אנו רואים בך מקור השראה כאשת מקצוע וכמנחה ואין לנו ספק שתרומתך לפרויקט ניכרת בכל היבט שלו. אנו מלאי הערכה על ההזדמנות שניתנה לנו ללמוד ולהתפתח בהנחייתך.

תוכן עניינים

רשימת איורים

תקציר

פרויקט זה בוחן שיטה חדשנית לאחזור קוד, במטרה לתמוך בשימוש חוזר בתוכנה ולשפר את חוויית הפיתוח. בניגוד לגישות מסורתיות אשר מבצעות התאמה ישירה בין שאילתות בשפה טבעית לבין קוד המקור או תיעוד טכני, אנו מציעים גישה עקיפה המבוססת על סיכומים טקסטואליים. במסגרת הגישה שלנו, נעשה שימוש במודלים שפתיים גדולים (LLMs) לצורך יצירת סיכומים בשפה טבעית עבור קטעי קוד, כאשר האחזור מתבצע באמצעות השוואה בין הסיכומים לשאילתות במקום לקוד עצמו. את הניסוי ביצענו בהתבסס על מאגר הקוד CodeSearchNet, ובו בחנו מספר מודלים מתקדמים לסיכום קוד, בהם LLaMA, GPT, CodeT5 ו־DeepSeek. כל אחד מהמודלים ייצר מאגר נתונים חדש הכולל דוגמאות קוד, שאילתות תואמות וסיכומים טקסטואליים. לצורך הערכת איכות האחזור, נעשה שימוש במודל מהתחרות (Code Reuse Challenge) COIR , אשר נחשב לקו בסיס חזק באחזור קוד במסגרת אתגר NeurIPS 2022 . לבסוף, ביצענו השוואה בין המודלים ונמצא כי מודל DeepSeek הפיק את הסיכומים האיכותיים ביותר, וכאשר שולב עם מודלים מובילים לאחזור מתוך התחרות התקבלו ביצועים גבוהים במיוחד.

מילות מפתח: סיכום קוד, אחזור קוד, מודלים שפתיים גדולים CodeSearchNet, CoIR, , DeepSeek, CodeT5, LLaMA, GPT, איכות אחזור, השוואת מודלים, שימוש חוזר בקוד, סיכום בשפה טבעית, הערכת ביצועים, ייצוגים וקטוריים.

קישור לגיטהאב: https://github.com/Liorgof/SW-Reuse

פרק 1- מבוא

שימוש חוזר בקוד הוא עקרון מרכזי בפיתוח תוכנה, שמטרתו להקטין את זמן הפיתוח, להפחית עלויות ולשפר את תחזוקת התוכנה. גישה זו מאפשרת למפתחים לעשות שימוש חוזר ברכיבים קיימים במקום לכתוב אותם מחדש, דבר שמייעל את העבודה ומונע כפילויות. עם זאת, קיימת בעיה משמעותית בתהליך זה : איתור קטעי קוד רלוונטיים יכול להיות משימה מורכבת בשל הבדלים בסגנון כתיבת הקוד, חוסר תיעוד מספק, ומגוון רחב של שפות תכנות וסביבות עבודה.

בעשור האחרון, עם התקדמות הטכנולוגיה בתחום הבינה המלאכותית, במיוחד בפיתוח מודלים מתקדמים לעיבוד שפה, החלו להופיע פתרונות שמאפשרים חיפוש קטעי קוד לפי הבנה טקסטואלית ולא רק לפי התאמה מילולית. עם זאת, הפתרונות הקיימים עדיין מתקשים לספק תוצאות מדויקות במקרים בהם מבנה הקוד שונה מהצורה בה המשתמש מחפש אותו.

הפרויקט שלנו מציע גישה חדשנית להתמודדות עם בעיה זו. הרעיון המרכזי הוא להשתמש בטכנולוגיה לעיבוד שפה כדי ליצור סיכומים תמציתיים של קטעי קוד, ולאחר מכן לבצע חיפוש בהתבסס על הסיכומים במקום על הקוד עצמו. כך ניתן לגשר על הפער בין הדרך שבה בני אדם מתארים קוד לבין הדרך שבה הוא כתוב בפועל. שיטה זו מאפשרת למפתחים לבצע חיפוש בשפה טבעית, ולקבל קטעי קוד המתאימים לצרכים שלהם גם אם אינם מכירים את התחביר או את המבנה הספציפי של הקוד.

יתרון משמעותי של גישה זו הוא בכך שהיא משפרת את הנגישות של קטעי קוד קיימים, ומקלה על תהליך ההבנה של פרויקטים גדולים. בעזרת שילוב של כלים מתקדמים, ניתן לבצע סינון חכם יותר של תוצאות ולהפחית את הזמן הדרוש למציאת פתרון מתאים. הפרויקט ייבחן על מאגרי קוד שונים כדי להוכיח את יעילותו בהשוואה לשיטות המסורתיות לאיתור קוד חוזר.

פרק 2- סקירת ספרות

הקדמה והצגת הבעיה

שימוש חוזר בקוד (Code Reuse) הוא גישה בפיתוח תוכנה שמטרתה להקטין את עלויות הפיתוח, לשפר את איכות הקוד ולהאיץ את תהליך הפיתוח. המהות היא להטמיע רכיבים קיימים בפרויקטים חדשים במקום לכתוב אותם מאפס. תהליך זה, למרות יתרונותיו הברורים, דורש מיומנות גבוהה, גישה לרכיבים תואמים, ותיעוד איכותי.

למרות היתרונות, ישנם מספר חסמים בולטים, רכיבים רבים אינם מתועדים כהלכה, מה שמקשה על הבנתם והתאמתם לשימושים שונים. לעיתים קרובות, התאמת רכיבים למערכות שונות מחייבת ביצוע שינויים מבניים משמעותיים, תהליך שמוסיף למורכבות. בנוסף, איתור רכיבים מתאימים מתוך מאגרי מידע גדולים עשוי להיות משימה עתירת משאבים ודורשת השקעה רבה של זמן ומאמץ.

בשנים האחרונות, מודלים גדולים לשפה (LLMs) כמו GPT ו ChatGPT-חוללו מהפכה ביכולת להבין שפה טבעית ולקדם את השימוש החוזר בקוד באמצעות כלי אוטומציה חכמים. מודלים אלו תומכים בזיהוי רכיבים רלוונטיים, הצעת שיפורים לקוד ,(Refactoring) ומתן תובנות מיידיות למפתחים.

2.1 LLMs (Large Language Models) בהקשר של שימוש חוזר בקוד

הגדרה ותכונות מרכזיות
מודלים לשוניים גדולים (LLMs) הם מערכות בינה מלאכותית מתקדמות המבוססות על ארכיטקטורות רשתות עצביות, ובעיקר טרנספורמרים. הם מאומנים על מאגרי נתונים רחבים הכוללים קוד מקור בשפות תכנות שונות, יחד עם דוקומנטציה טכנית ודיונים טקסטואליים . המודלים הללו נועדו לסייע בשימוש חוזר בקוד על ידי הבנת מבנים של קוד קיים, זיהוי רכיבים רלוונטיים והצעת פתרונות מותאמים.

2.1.1 יכולות

- הבנת טקסט ויצירתו : LLMs מפגינים יכולות הבנה עמוקה של הקשר טקסטואלי ויצירת טקסטים עקביים ורלוונטיים. הם עוזרים למפתחים להבין את מטרת הקוד ולשלבו בפרויקטים, מספקים תיעוד טקסטואלי לקוד קיים, דבר המאפשר הבנה מהירה ושימוש חוזר בקוד [3].

- התאמה למשימות ספציפיות : באמצעות ,Fine-Tuning ניתן להתאים את המודלים למשימות ייחודיות כגון יצירת קוד מתוך מאגרי מידע קיימים.
לדוגמא שימוש ב- Fine-Tuning על מאגרי קוד פתוח משפר את יכולת המודלים לזהות פונקציות מוכנות לשימוש חוזר [1].
מאמר [5] מתאר את תרומתם של LLMs בפרויקטים קהילתיים בקוד פתוח, במיוחד בזיהוי ושימוש חוזר במודולים קיימים.

- תמיכה במודולריזציה ואופטימיזציה : LLMs מסייעים בהפיכת קוד למודולרי ונוח לשימוש חוזר. לדוגמא, מטרת מערכת A3-CodGen היא לפשט את

התהליך על ידי שימוש במידע מקומי, גלובלי וספריות צד שלישי ליצירת רכיבי קוד מותאמים . [9]

- שיפור תהליכי תיעוד והסבר : מודלים לשוניים יוצרים תיעוד אוטומטי לקוד, המספק למפתחים הסברים מפורטים ומקלים על שילוב רכיבים קיימים.

2.1.2 כלים קיימים

- עזרי קוד : כלים כמו GitHub Copilot, המבוססים על מודלים לשוניים גדולים (LLMs)כגון OpenAI Codex, הפכו לכלים מהפכניים עבור מפתחים. עזרי קוד אלו מסייעים למפתחים בשלושה תחומים עיקריים :
  1. יצירת קוד :המערכת מציעה הצעות קוד מותאמות בהקשר לקוד הנכתב, ומסייעת בכתיבת פונקציות שלמות בהתבסס על כוונת המשתמש.
  2. הסברת קוד :הכלי מסוגל להסביר קוד קיים בשפה פשוטה וברורה, מה שעוזר למפתחים להבין רכיבים מורכבים בתוכנה.
  3. תיקון שגיאות :זיהוי ותיקון של באגים באופן אוטומטי או באמצעות המלצות מותאמות.

  היכולות האלו לא רק משפרות את מהירות הפיתוח, אלא גם את איכות הקוד. לדוגמה, מפתחים יכולים לזהות שגיאות מוקדם יותר בתהליך העבודה, מה שחוסך זמן ומשאבים. [4]

- מערכות המלצה : מודלים לשוניים גדולים (LLMs) משמשים גם כבסיס לפיתוח מערכות המלצה מתקדמות, המסוגלות :
  1. ללמוד דפוסים ממאגרי נתונים גדולים :המודלים משתמשים בטכניקות למידת מכונה כדי לזהות העדפות משתמשים מתוך נתונים היסטוריים.
  2. להציע המלצות מותאמות אישית :בין אם מדובר בהמלצה על סרטים, מוצרים או שירותים, מערכות ההמלצה פועלות בזמן אמת כדי לספק למשתמשים תוכן שמתאים לצרכים שלהם.

  דוגמה לכך ניתן לראות במערכות כמו Netflix או Amazon, שבהן המלצות מותאמות אישית תורמות לשיפור חוויית המשתמש והגדלת המעורבות.
  מערכות אלו מבוססות על שימוש חכם ב LLMs-שמאפשר ניתוח מדויק של נתוני משתמשים ודפוסי צריכה. [1]

- סוכני AI : סוכני AI הם דוגמה מתקדמת ליכולות של LLMs, המסוגלים לשלב בין עיבוד נתונים לבין אינטראקציה עם כלים חיצוניים ו-APIs . יכולותיהם כוללות :
  1. פתרון משימות מורכבות ורב-שלביות : לדוגמה, ניהול לוגיסטיקה, ביצוע תהליכים אוטומטיים מורכבים, או שילוב בין מקורות מידע שונים.
  2. למידה והשתפרות עצמאית :הסוכנים יכולים ללמוד מדפוסים קיימים ולשפר את ביצועיהם לאורך זמן.
  3. פוטנציאל להתקדמות ל) AGI-בינה מלאכותית כללית :(סוכני AI מסוגלים להציג תהליכי חשיבה דמויי אנוש, תוך שהם משלבים בין משימות שונות ביעילות יוצאת דופן.

  שימושים מתקדמים אלו כוללים פיתוח עוזרים אישיים וירטואליים, כמו Amazon Alexa או Google Assistant, המספקים שירותים מותאמים אישית בזמן אמת. [2]

2.1.3 יתרונות

- ידע עולם עשיר: LLMs מאומנים על מאגרי נתונים עצומים, ומסוגלים להציע מידע מדויק ורלוונטי להקשר ובכך מודלים מסוגלים לנסח תשובות המתאימות לסיטואציות מגוונות, כולל פתרון בעיות רב-שלביות ושימוש חוזר בקוד [2].

- ייעול אוטומציה: המודלים מפחיתים עומס עבודה קוגניטיבי על ידי ביצוע משימות באופן אוטומטי, מה שמאפשר לבני האדם להתמקד במשימות יצירתיות או בעלות ערך מוסף גבוה יותר [3].

2.1.4 שאלות פתוחות וחסרונות בתחום

- עלויות חישוב גבוהות: אימון וכיוונון של LLMs דורשים משאבים חישוביים אדירים. תהליך האימון הראשוני כולל עיבוד של כמויות עצומות של נתונים, מה שמצריך שימוש בכוח מחשוב מתקדם כמו GPU או TPU. הכיוונון (fine-tuning) דורש משאבים נוספים להתאמה למשימות ייעודיות. העלויות הגבוהות עלולות להפוך את השימוש במודלים אלו ללא משתלם עבור יישומים מסחריים או מחקריים עם תקציב מוגבל. התלות במערכות מחשוב חזקות מגבילה את זמינות הטכנולוגיה, במיוחד עבור קהילות מדעיות או מפתחים במדינות מתפתחות [1].

- שגיאות אפשריות: על אף היכולות המתקדמות של LLMs, הם עלולים להפיק תוצאות שנראות נכונות אך בפועל אינן מדויקות. לדוגמה, המודלים עשויים לנסח תשובות שהן תחבירית תקינות אך מכילות טעויות לוגיות או עובדתיות. במערכות קריטיות, כמו רפואה או משפטים, תוצאה שגויה יכולה לגרום לנזק משמעותי או להשלכות משפטיות. בנוסף, המשתמשים עלולים לפתח תלות במודלים ולהאמין ביכולת שלהם באופן מוחלט, מה שמוביל לבעיות של אמינות ואמון [3].

- שיקולים אתיים: הטיה בנתוני האימון, מאגרי הנתונים שבהם מאומנים LLMs כוללים לעיתים תכנים מוטים או לא מאוזנים, מה שיכול להשפיע על התוצאות שהמודל מפיק. משתמשים עשויים להישען על המודלים באופן מוגזם, מה שמוביל לצמצום המיומנויות שלהם ולתפיסה שגויה של יכולות הבינה המלאכותית. לכן יש צורך במחקר ופיקוח נוספים ובכך להבטיח שהמודלים פועלים בצורה הוגנת ואמינה.

2.1.5 פתרונות עתידיים ועתיד התחום של LLMs בהקשר לשימוש חוזר בקוד

- ייעול חישובי:
  1. שימוש בטכניקות כמו Fine-tuning יעיל, המאפשר אימון מודלים על תת-דגימות ממוקדות, ובכך מפחית את הצורך במשאבים חישוביים גדולים [1].
  2. פיתוח מודלים היברידיים, המשלבים בין LLMs כלליים למודלים קטנים יותר המותאמים למשימות ספציפיות [2].

- מזעור הטיה ואמינות:
  1. פיתוח שיטות לזיהוי ונטרול הטיות בנתוני האימון כדי להפוך את המודלים לשקופים והוגנים יותר [4].
  2. שימוש במנגנוני ביקורת אנושיים ובינה מלאכותית משלימה, המסוגלת לבדוק את פלט המודל לפני השימוש בו במערכות קריטיות [3].

- גישה מבוססת הקשר:
  1. פיתוח מערכות שמסוגלות לשלב בין נתונים ממקורות שונים בזמן אמת כדי לייצר תשובות מדויקות ורלוונטיות יותר [2]
  2. שילוב מודלים עם כלי חישוב נוספים, כמו APIs, ליצירת פתרונות אינטגרטיביים [2].

LLMs מציגים פוטנציאל לשינוי מהותי בתחומים מגוונים, כגון שיפור בתהליכי פיתוח תוכנה, הבנת קוד מורכב, שימוש חוזר בקוד מתוך מאגרי נתונים שונים ותכנון פתרונות למשימות רב-שלביות. [3].

2.2 שימוש חוזר בקוד (Code Reuse)

שימוש חוזר בקוד (Code Reuse) הוא פרקטיקה מרכזית בפיתוח תוכנה, המאפשרת למפתחים להשתמש במקטעי קוד, פונקציות או מודולים קיימים במקום לכתוב קוד חדש. גישה זו מפחיתה עלויות, משפרת את איכות הקוד ומאיצה את תהליך הפיתוח. בעוד שבעבר יושמה בעיקר בארגונים, כיום היא נפוצה גם בפרויקטים בקוד פתוח, יישומים חינוכיים ובשילוב עם מודלים מתקדמים של בינה מלאכותית .(LLMs) יחד עם זאת, התחום מתמודד עם אתגרים טכנולוגיים, ארגוניים ואתיים, הדורשים פתרונות חדשניים.

2.2.1 יכולות

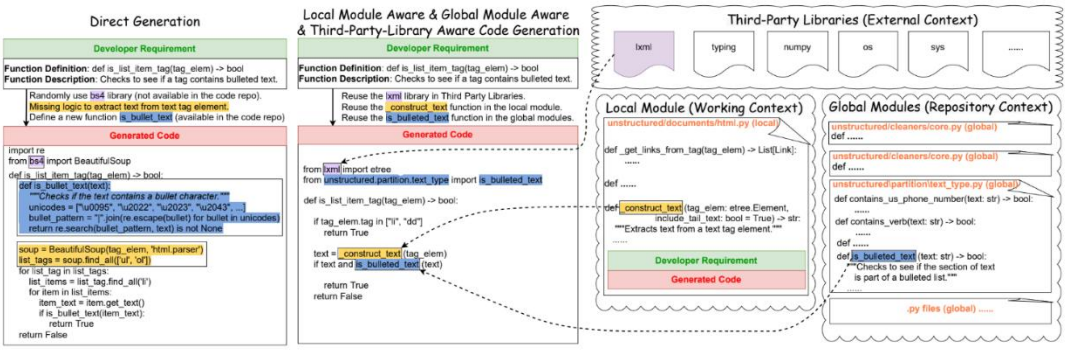
- שימוש במאגרי מידע קיימים : מפתחים נוטים להשתמש בקוד שכבר נכתב בפרויקטים אחרים כדי לחסוך זמן ומשאבים. עם זאת, לפני שהם משתמשים ברכיבי קוד אלו, הם מעריכים את איכותם בכמה שלבים :
  1. בדיקת איכות הקוד בכך שהם בודקים שהקוד עומד בסטנדרטים מקצועיים כמו שימוש בסגנון כתיבה אחיד או הימנעות משגיאות תכנותיות.
  2. האם הקוד נכתב בצורה "מודולרית", כלומר, האם הוא מחולק למקטעים קטנים וברורים שקל לשלבם בפרויקט אחר.
  3. האם הקוד מתוחזק באופן פעיל, למשל, האם המפתחים המקוריים ממשיכים לעדכן אותו כדי לתקן בעיות או לשפר אותו.

2.2.2 כלים קיימים

GitHub Copilot ו-AI Code Whisperer עוזרים למפתחים ליצור ולשפר קוד באמצעות הצעות מבוססות AI [8].

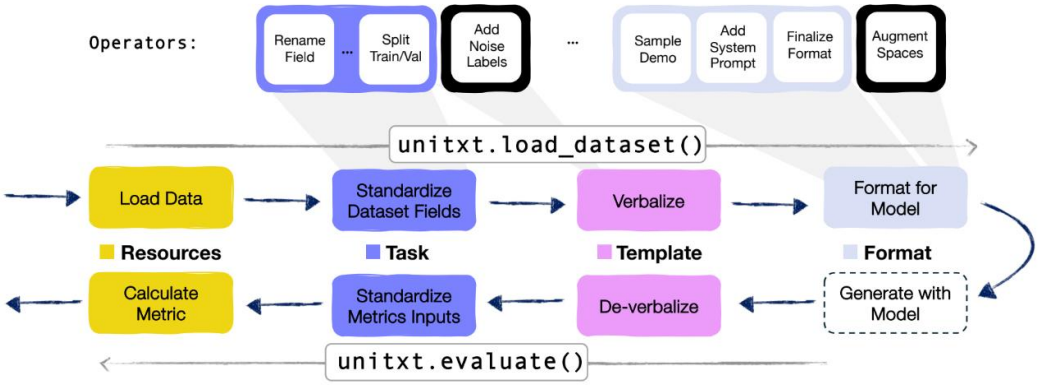
2.2.3 כלים מחקריים

- A3-CodGen - מסגרת מחקרית המשתמשת במידע מקומי, גלובלי וספריות צד שלישי לצורך יצירת קוד מותאם ומדויק [9].



איור 1- יצירת קוד ללא שימוש חוזר בקוד לעומת יצירת קוד עם שימוש חוזר בקוד

- Unitxt- ספרייה מודולרית ליצירה, עיבוד ושיתוף קוד לשימוש חוזר, המשפרת את הגמישות והיעילות של תהליכי פיתוח [7].



איור 2- החלק בעליון מתאר את השלבים של הכנת הנתונים, הכולל טעינת מערכי נתונים, סטנדרטיזציה בהתאם למשימה, עיבוד מילולי באמצעות תבניות, ויישום של עיצוב פורמלי.
החלק התחתון מתאר את שלבי ההערכה, הכולל פעולות הסרת עיבוד מילולי וסטנדרטיזציה של התוצאות לפני הערכת הביצועים באמצעות מדדים שהוגדרו עבור המשימה.

2.2.4 יתרונות

- חסכון בזמן ומשאבים
שימוש חוזר בקוד מפחית את הזמן הנדרש לפיתוח רכיבי תוכנה חדשים ומשפר את יעילות הפיתוח, מכיוון שמפתחים יכולים להסתמך על קוד קיים שכבר פותח, נבדק ומיושם בפרויקטים אחרים [5].

- שיפור האיכות
קוד שנעשה בו שימוש חוזר נוטה להיות איכותי יותר מכיוון שהוא כבר נבדק בפרויקטים אחרים ועבר תהליכי תיקון ושיפור [7].

- שיתוף פעולה והערכה פשוטים
כלים כמו Unitxt מאפשרים למפתחים לשתף בקלות קטעי קוד, מודולים וספריות עם מפתחים אחרים, תוך שמירה על עקביות ושקיפות. השיתוף נעשה באמצעות מתכונים מודולריים (Recipes), שמספקים למפתחים הוראות ברורות לשימוש ברכיבי הקוד ושילובם בתוכנה אחרת [7].

2.2.5 שאלות פתוחות וחסרונות בתחום

שאלות פתוחות:

1. כיצד ניתן להגדיר ולהטמיע סטנדרטים אוניברסליים לשימוש חוזר בקוד?
מאמר [7] על Unitxt מציין את הצורך במתודולוגיה מודולרית ושימוש במתכונים (Recipes) לשיתוף קוד בצורה מסודרת ומובנית. אך לא קיים עדיין סטנדרט אחיד בתחום.

2. איך להתמודד עם בעיות רישוי בקוד פתוח?
מאמר [5] מדגיש את חשיבות השמירה על רישיונות בפרויקטים בקוד פתוח ומעלה את האתגר של תאימות רישוי כאשר קוד נעשה בו שימוש חוזר.

3. כיצד ניתן להבטיח תיעוד מספק לקוד המיועד לשימוש חוזר?
מאמר [3] מציג את תרומת LLMs בכתיבת תיעוד אוטומטי שמסביר את מטרת הקוד ואופן השימוש בו, אך התיעוד אינו תמיד שלם או מותאם באופן מלא.

4. איך לשלב קוד ממאגרים שונים תוך שמירה על עקביות?
מאמר [9] מציג את המערכת A3-CodGen אשר מנסה להתמודד עם אתגר זה על ידי שילוב מידע מקומי, גלובלי וספריות צד שלישי, אך עדיין מדובר בתהליך מורכב.

5. כיצד להניע מפתחים לשתף קוד בפרויקטים בקוד פתוח?
מאמר [5] מעלה את בעיית חוסר התמריצים למפתחים לשתף קוד בצורה שמתאימה לשימוש חוזר, במיוחד כשמדובר בהתנדבות בקהילות קוד פתוח.

חסרונות הקיימים היום בתחום

1. חוסר סטנדרניזציה- בתחום של שימוש חוזר בקוד חסרים סטנדרטים אחידים, במיוחד בפרויקטים גדולים ומורכבים שבהם המודולריות אינה תמיד ברורה [7].

2. בעיות תחזוקה- שמירה על תאימות הקוד לאורך זמן, במיוחד כאשר קוד משולב מספריות צד שלישי שאינן מתעדכנות [9].

3. סיכוני אבטחה- שימוש חוזר בקוד עשוי לחשוף נתונים רגישים במאגרי קוד משותפים, במיוחד כאשר משולבים מאגרי מידע מסביבות שונות. [10].

4. חוסר תיעוד- אף שה LLMs יכולים לכתוב תיעוד אוטומטי, תיעוד זה אינו תמיד מספק ומותיר פערים עבור מפתחים אחרים [3].

5. היעדר תמריצים- ללא תמריצים כלכליים או חברתיים, מפתחים בקהילות קוד פתוח אינם ששים לשתף קוד שמתאים לשימוש חוזר [5].

6. מורכבות באינטגרציה- מתאר את הקושי בשילוב קוד ממקורות שונים בסביבה טכנולוגית משתנה, במיוחד כאשר גרסאות אינן תואמות [9].

7. פגיעה ביצירתיות- שימוש חוזר בקוד עלול לגרום למפתחים להסתמך יתר על המידה על פתרונות קיימים במקום לפתח גישות חדשניות [6].

השאלות והחסרונות שהוזכרו ממחישים את האתגרים בתחום השימוש החוזר בקוד. בעוד מציעים ומנסים לפתח פתרונות מגוונים, עדיין נדרש מחקר נוסף וכלים מתקדמים כדי להתגבר על האתגרים הללו ולהפוך את התהליך ליעיל יותר.

2.2.6 פתרונות עתידיים

- סטנדרטיזציה ושיתוף פעולה
  יצירת תקנים אוניברסליים לשימוש חוזר בקוד יכולה לשפר את העקביות והאיכות של הקוד המשותף בין פרויקטים ומפתחים. סטנדרטיזציה יכולה לכלול כללים ברורים למבנה הקוד, תיעוד אחיד, ודרכים להערכת איכות הקוד [7].

  <u>פתרון אפשרי</u>: יצירת פלטפורמות שיתופיות המשלבות מתכונים (Recipes) מודולריים כדי לתעד ולהסביר שימוש חוזר בקוד בצורה נוחה וברורה.

- שימוש בבינה מלאכותית
  מודלים של AI, כמו *A3-CodGen*, יכולים לנתח באופן אוטומטי מאגרי קוד, לזהות רכיבים שמתאימים לשימוש חוזר, ולשלב אותם בתוכנה חדשה. יתר על כן, מודלים אלו יכולים לייצר קוד מותאם אישית בהתאם לצרכים הספציפיים של הפרויקט. A3-CodGen משתמש בגישה "מודעת-הקשר", המשלבת מידע מקומי (מהקובץ הנוכחי), גלובלי (ממאגרי קוד אחרים בפרויקט), וספריות צד שלישי, מה שמבטיח שהקוד המשולב יהיה מדויק ותואם לתקנים הנוכחיים [9].

  <u>פתרון אפשרי</u>: פיתוח כלים המשלבים LLMs ליצירת תיעוד אוטומטי, אופטימיזציה ומודולריזציה של קוד.

- פתרון בעיות רישוי
  כלים חכמים לזיהוי סוגי רישוי בקוד ולווידוא תאימות לפני שימוש חוזר. בנוסף, פיתוח מנגנונים להכללת מידע רישוי בצורה אוטומטית בקוד שנעשה בו שימוש חוזר [5].

  <u>פתרון אפשרי</u>: שילוב כלים של בינה מלאכותית לבדיקת רישיונות ולמניעת הפרות רישוי.

- שיפור אבטחת קוד
  פיתוח מערכות לזיהוי בעיות אבטחה בקוד שנעשה בו שימוש חוזר בקוד והצעת תיקונים בזמן אמת [9]

- שיפור תיעוד קוד
  פיתוח מנועי AI לכתיבת תיעוד מלא ואטומטי לכל רכיב קוד [3].

- התאמת קוד לסביבות משתנות
  נצטרך לזהות התאמות נדרשות וליצור גרסאות מותאמות של הקוד לסביבות עבודה שונות [9]. לדוגמא: אם פרויקט פותח עבור Python 3.7 המערכת תוכל לשדרג את הקוד כך שיתאים ל Python 3.10 תוך שמירה על תאימות.

<u>פתרונות אפשרי</u>:

1. שימוש ב LLMs-לניתוח הדרישות של כל סביבה (לדוגמה: סוג מערכת הפעלה, גרסאות ספריות, תלויות).
2. זיהוי ההבדלים בין סביבות העבודה והצעת פתרונות מותאמים.
3. הפקת גרסה חדשה של הקוד שתתאים לכל סביבה. לדוגמא: שינויים בקוד כך שיתאים לסביבת פיתוח בענן או לסביבת Desktop .

שימוש חוזר בקוד (Code Reuse) מציע יתרונות משמעותיים מבחינת יעילות, איכות וחיסכון בזמן, אך מתמודד עם אתגרים בתחום התקנים, התמריצים והתחזוקה. הפתרונות העתידיים המתוארים מבוססים על מחקרים מתוך המאמרים. הם מדגישים את הצורך בשיפור כלים מבוססי AI , יצירת תמריצים למפתחים, פתרון אתגרי רישוי, ושיפור התיעוד והאבטחה. תחום השימוש החוזר בקוד נמצא בפני התפתחות משמעותית, עם פוטנציאל להפוך את תהליכי הפיתוח ליעילים, בטוחים ומבוססי שיתוף פעולה רחב יותר.

פרק 3- דרישות

3.1 דרישות פונקציונליות

1. יכולת סיכום קוד –המערכת תספק מנגנון לסיכום אוטומטי של מקטעי קוד, תוך שמירה על הדיוק והמשמעות של כל קטע.
2. אחזור קוד מבוסס סיכומים –המשתמש יוכל לחפש קטעי קוד רלוונטיים על בסיס שאילתות טקסטואליות, אשר יושוו לתקצירי הקוד ולא ישירות לקוד עצמו.
3. שיפור איכות הסיכום –שימוש במודל מתקדם שיוכל לשפר את איכות התקצירים ולספק סיכומים ברורים, תמציתיים ורלוונטיים.
4. תמיכה במספר שפות תכנות –המערכת תוכל להתמודד עם מגוון שפות תכנות מתוך מאגרי מידע שונים.
5. השוואת ביצועים –המערכת תאפשר מדידה של דיוק החיפושים בהשוואה למערכות אחזור מידע קיימות.

3.2 דרישות לא-פונקציונליות

1. דיוק ואמינות –המערכת תמדוד את מידת ההתאמה של הסיכומים לחיפושים ותחזיר תוצאות מדויקות ככל האפשר.
2. יכולת הרחבה –התמיכה במאגרים גדולים של קוד תאפשר למערכת להתמודד עם כמויות נתונים משמעותיות.
3. אינטגרציה עם כלים קיימים –המערכת תאפשר שילוב עם פלטפורמות קיימות כמו סביבת פיתוח משולבת, מאגרי קוד מקוונים וכלי ניתוח קוד.
4. שמירה על פרטיות ואבטחת מידע –המערכת תכלול מנגנונים למניעת אחזור קוד הכולל מידע רגיש, ותבטיח הגנה על מאגרי הקוד מהם נאספים הנתונים.

פרק 4- השיטה

4.1 מטרת המחקר

המטרה המרכזית של המחקר היא לבחון את היכולת של מודלים לשוניים גדולים (LLMs) לשפר את תהליך השימוש החוזר בקוד באמצעות סיכום אוטומטי של פונקציות. בפרט, אנו בודקים האם סיכומים טקסטואליים בשפה טבעית של קטעי קוד יכולים להוות גשר בין אופן החיפוש של המשתמש (שנעשה לרוב בשפה טבעית) לבין המבנה הפורמלי והטכני של קוד המקור. המחקר מציע לשלב טכנולוגיות עיבוד שפה טבעית במערכות אחזור קוד, באופן שיאפשר למפתחים לחפש ולהבין רכיבי קוד בצורה נגישה, מהירה ויעילה יותר גם כשאינם מכירים את התחביר או את מבנה הפונקציה שהם מחפשים. בכך, המחקר שואף להנגיש קוד קיים לצרכים חדשים ולייעל את תהליך הפיתוח בתחומים מגוונים.

4.2 שאלת המחקר

- האם סיכומים בשפה טבעית של קוד יכולים לשפר את הביצועים של אחזור קוד לעומת גישות מסורתיות המתבססות על הקוד הגולמי?

  שאלה זו בוחנת האם ניתן להגיע לדיוק גבוה יותר באיתור פונקציות רלוונטיות כאשר תהליך החיפוש מבוסס על סיכום מילולי של מטרת הקוד, במקום להשוות את השאילתה ישירות מול טקסט הקוד עצמו.

- איזה מודל סיכום מספק את הסיכומים האיכותיים ביותר מבחינת בהירות, דיוק ורלוונטיות לצורכי אחזור קוד?

  כאן אנו בודקים מספר מודלים מובילים (כגון CodeT5, CodeBERT, DeepSeek ועוד) כדי לזהות את המודל שמספק את הסיכום המדויק והיעיל ביותר בהקשר של שימוש חוזר בקוד.

4.3 חשיבות המחקר

המחקר שלנו עוסק בסוגיה מרכזית בעולמות התכנות המודרני: היכולת לעשות שימוש חוזר בקוד באופן חכם, מהיר ונגיש. בעידן שבו מאגרי קוד עצומים זמינים לכל מפתח, האתגר האמיתי הוא לא רק ליצור קוד אלא לאתר ולהשתמש בקוד שכבר קיים. עם זאת, האיתור של פונקציות רלוונטיות מתוך מאגרים אלו הוא לרוב משימה לא פשוטה, במיוחד כאשר הקוד אינו מתועד היטב או נכתב בסגנון שונה מהציפיות של המשתמש.

שילוב של מודלים לשוניים המסוגלים לנסח סיכום קצר וברור של כל פונקציה יכול לשנות את כללי המשחק. סיכום איכותי מפשט את ההבנה של קטעי קוד, מזרז את תהליך הפיתוח, מפחית טעויות, ומאפשר תחזוקה נוחה יותר של מערכות גדולות.

מעבר להיבט הטכנולוגי, המחקר גם בוחן את הפוטנציאל של בינה מלאכותית לשפר את חוויית המפתח, להעצים את השימוש בכלים חכמים, ולהנגיש ידע טכנולוגי גם למפתחים מתחילים או לאנשי תוכן שאינם מתכנתים מקצועיים.

לכן, למחקר חשיבות אקדמית, תעשייתית וחינוכית והוא יכול להוות בסיס לפיתוח כלים עתידיים שישפרו את סביבת הפיתוח ויגבירו את השימוש האפקטיבי במאגרים קיימים של קוד.

4.4 השערות המחקר

- אחת מהבעיות המרכזיות בתהליכי אחזור קוד ממאגרים גדולים היא הפער שבין אופן החיפוש של המשתמש שמתבצע לרוב בשפה טבעית (כגון "sort a list by length") לבין הייצוג של הפונקציות בשפת תכנות טכנית, שלרוב אינה מתועדת או לא מנוסחת באופן שמתכתב ישירות עם כוונת המשתמש. כלומר, יש פער בין איך אנחנו חושבים ומנסחים צורך, לבין איך הקוד עצמו כתוב .השערה זו טוענת כי אם נוכל לייצר סיכום טקסטואלי ברור וקצר לכל פונקציה תיאור מילולי שמסביר מה הפונקציה עושה אז תהליך האחזור ישתפר משמעותית, כי נוכל להשוות את השאילתה של המשתמש ישירות אל הסיכומים, במקום לקוד עצמו .השוואת טקסטים בשפה טבעית היא משימה בה מודלים לשוניים מתקדמים מצטיינים, ולכן שיטה זו צפויה להניב תוצאות מדויקות יותר. הסיכומים פועלים למעשה כמתווך בין השפה האנושית לקוד הטכני וככל שהסיכום מדויק, כך ניתן לאתר את הפונקציה הנכונה בקלות רבה יותר.

- CodeT5 הוא מודל שתוכנן מראש להתמודדות עם משימות של קוד תכנותי, ובמיוחד לסיכום פונקציות, תרגום בין שפות קוד, והשלמת קוד חסר. המודל מבוסס על ארכיטקטורת T5 , ועבר אימון על מאגרי קוד רחבים, מה שהופך אותו לכלי ייעודי ויעיל מאוד במשימות של סיכום קוד. ביצוע Fine-Tuning (כוונון נוסף) של CodeT5 על מאגרי קוד ממוקדים מאפשר למודל להתמקד טוב יותר במשימות הספציפיות של המחקר, כגון יצירת סיכומים קצרים, מדויקים ורלוונטיים .עם זאת, בשנים האחרונות הופיעו מודלים חזקים יותר בהיבט של גודל, הבנה הקשרית, ויכולת כללית להתמודד עם מידע מורכב, דוגמת LLaMA ו־ DeepSeek. מודלים אלו אמנם לא ייעודיים לקוד בלבד, אך בזכות ההיקף האדיר של נתוני האימון שלהם, המורכבות הארכיטקטונית שלהם, והיכולת להבין הקשרים מורכבים, הם מציגים פעמים רבות ביצועים העולים גם על מודלים מתמחים, במיוחד לאחר Fine-Tuning מותאם-משימה .במילים אחרות, למרות ש־ CodeT5 מותאם לקוד, המודלים החדשים יותר, כשהם מותאמים כראוי, עשויים להפיק סיכומים עמוקים, עשירים וברורים יותר ולשפר אף יותר את תהליך אחזור הקוד. הסיבה לכך טמונה לא רק בהתמקדות על קוד, אלא גם ביכולת להבין ולהביע את המשמעות של הקוד בצורה בהירה ומשכנעת, כפי שנדרש בסיכום איכותי .לכן, ההשערה היא שהיכולת המובנית של המודל בלבד אינה מספיקה אלא יש לבחון גם את העוצמה של המודל, את גודלו, ואת יכולתו הכללית להתמודד עם טקסטים מורכבים. לפיכך, מודלים עוצמתיים כמו DeepSeek או LLaMA עשויים להניב תוצאות טובות אף יותר, למרות ש־ CodeT5 עבר התאמה ייעודית לתחום.

4.5 תוצרים צפויים של המחקר

בסיום המחקר אנו מצפים לגבש תובנות מבוססות לגבי ההשפעה של סיכום קוד בשפה טבעית על תהליך אחזור קוד. אף על פי שלא ניתן יהיה לקבוע באופן חד־משמעי כי סיכום תמיד משפר את תהליך האחזור, נוכל להצביע על מגמות ברורות והקשרים מסוימים שבהם הסיכום תורם באופן מובהק לדיוק ואפקטיביות החיפוש. כמו כן, נצבור הבנה מעמיקה לגבי הבדלים בין מודלים שונים (כגון CodeT5, DeepSeek, LLaMA) ביכולת שלהם להפיק סיכומים איכותיים, ונוכל להמליץ אילו מודלים מתאימים יותר למשימות של אחזור קוד מבוסס סיכום. בנוסף, תתקבל טבלת השוואה מפורטת בין גישות שונות, כלים קיימים, ומדדים של איכות האחזור אשר תוכל לשמש כבסיס למחקרים עתידיים או ליישומים מעשיים בתעשייה.

4.6 מהלך המחקר

<u>שלב 1</u> : איסוף והכנת הנתונים

- איתור מאגרי קוד המכילים פונקציות יחד עם תיעוד או סיכום טקסטואלי, בדגש על מאגר CodeSearchNet .
- ארגון הנתונים כך שיכללו קוד מקור, סיכום טקסטואלי ושאילתת חיפוש מתאימה.
- טיוב הנתונים : ניקוי, תקנון פורמט, סידור לפי קטגוריות/שפות תכנות.

<u>שלב 2</u> : בחירת מודלים לסיכום קוד

- בחירה ראשונית של מודלים מתאימים, בהם :
- CodeT5- מודל ייעודי לסיכום קוד
- DeepSeek : מודל כללי ועוצמתי עם ביצועים גבוהים.
- LLaMA- מודל לשפה כללית עם יכולות הקשר מתקדמות.
- ביצוע Fine-Tuning על מודלים נבחרים בהתאם למבנה הנתונים שנאסף.

<u>שלב 3</u> : הפקת סיכומים והשוואה בין המודלים

- הפעלת המודלים על קטעי הקוד והפקת סיכומים טקסטואליים.
- השוואת איכות הסיכומים לפי מדדים של בהירות, דיוק ורלוונטיות.
- בחירה במודל המצטיין לסיכומים כהמשך לניסויי אחזור קוד.

<u>שלב 4</u> : בניית מנגנון אחזור קוד מבוסס סיכומים

- פיתוח מנגנון חיפוש המתבסס על השוואת השאילתה מול הסיכומים שנוצרו, ולא מול הקוד הגולמי.
- שימוש בטכניקות חישוב דמיון (כגון : Euclidean Distance, Cosine Similarity ו- DOT).
- יצירת טבלת השוואה הכוללת קוד, שאילתה, וסיכומים משלושה מודלים לצורך ניתוח ביצועים.

שלב 5 : ניסויים והשוואת ביצועים מול שיטות קיימות

- הרצת ניסויי אחזור על אלפי דוגמאות מתוך הטבלה שהופקה.
- השוואת התוצאות אל מול ביצועי מודלים שהשתתפו בתחרות CoIR , המתמקדת באחזור קוד.
- ניתוח ביצועים לפי מדדים : MRR, Recall@K, דיוק מיקום, וזמן אחזור.

שלב 6 : ניתוח תוצאות והסקת מסקנות

- זיהוי מגמות : באילו מקרים הסיכומים שיפרו אחזור ובאילו לא.
- בחינה של חוזקות וחולשות של כל מודל.
- הסקת מסקנות בנוגע ליעילות הכללית של השיטה המוצעת.
- ניסוח המלצות לעבודה עתידית ולשילוב גישות אלו בכלי פיתוח מודרניים.

4.7 אתגרים

במהלך ביצוע המחקר והשלבים השונים של פיתוח והערכת המודלים, נתקלנו במספר אתגרים טכניים ומתודולוגיים, אשר דרשו פתרונות מותאמים והובילו לעדכון מהלכי העבודה שלנו:

1. מגבלות שימוש ב-API:
בעת שליחת בקשות סיכום למודלים כמו DeepSeek ו־LLaMA דרך ממשקי API, נתקלנו בהגבלות על מספר הבקשות שניתן לשלוח בפרק זמן מסוים (Rate Limit). כל מפתח API הוגבל לכמות פניות יומית או חודשית. כדי להתמודד עם מגבלה זו, בנינו מנגנון סבב (Round-Robin) בין מספר מפתחות, שכלל גם מעקב אחר שימושים לכל מפתח. כך הצלחנו להבטיח שימוש אופטימלי, רציף ובטוח.

2. ההבדלים במבנה הפלט בין המודלים:
כל מודל מחזיר פלט במבנה שונה לעיתים כמחרוזת פשוטה, ולעיתים כ־JSON מורכב. נדרשנו לבנות פונקציות עיבוד שונות עבור כל מודל, כדי לאחזר את הסיכום עצמו בצורה עקבית ואחידה. השוואת הסיכומים דרשה אחידות במבנה ובסגנון הפלט, כדי לאפשר ניתוח אמיתי.

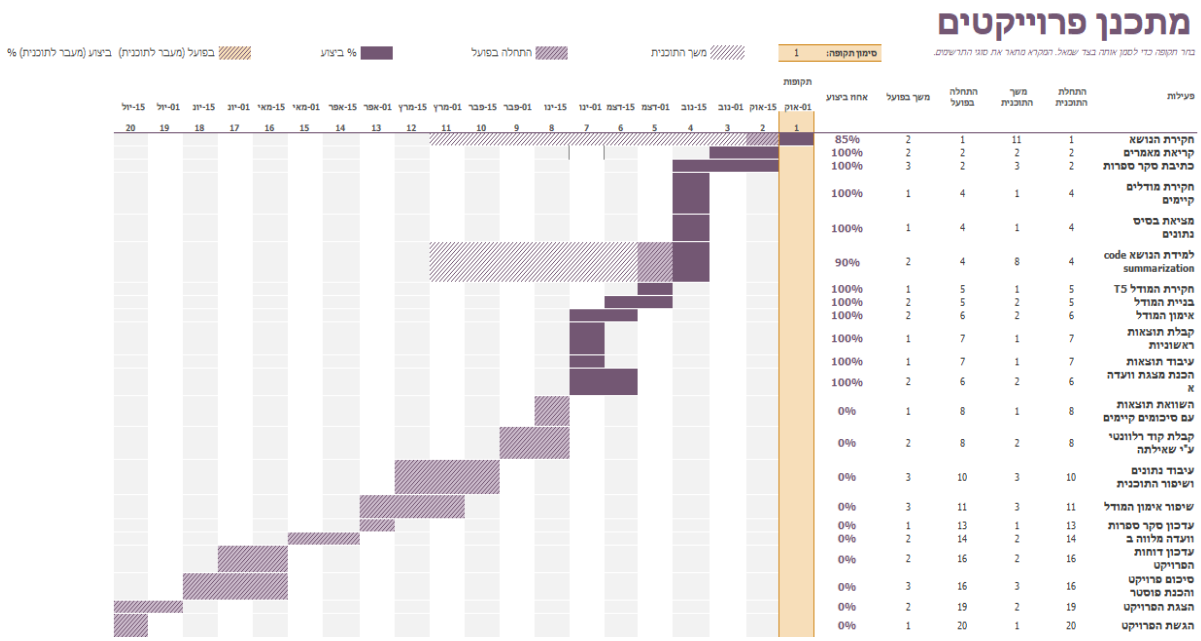
3. סיכומים חלקיים או כלליים מדי:
במקרים רבים (בעיקר מצד LLaMA), המודלים החזירו סיכומים כלליים שלא שיקפו בצורה מדויקת את פעולת הפונקציה. זה דרש מאיתנו לסנן תוצאות לא תקינות, להתמודד עם תקלות באחזור, ואף לחזור ולהפעיל את הבקשה מחדש.

4. שמירה על פורמט עקבי בטבלה הסופית:
שילוב סיכומים שונים לצד קוד המקור דרש מאיתנו להבטיח אחידות בטבלת הנתונים הסופית (DataFrame), כדי לאפשר ניתוח ישיר הן איכותני והן כמותי של תוצאות הסיכום וההשפעה על שלב אחזור הקוד.

5. זמן ריצה ממושך:
מאחר וכל סיכום דרש פנייה למודל חיצוני והמתנה לתגובה, תהליך הסיכום עבור מאות פונקציות היה איטי יחסית. לצורך כך שולב sleep יזום בין בקשות והוגדרו בדיקות תקינות כדי לאתר מקרים של תקלה בתשובה מ־API.

פרק 5- תוכנית עבודה

5.1 תרשים Gantt



איור 3- תרשים Gantt

5.2 סיכונים

זיהוי הסיכון	הגדרת פעילות התמודדות	סיווג חומרה	סיכויים
סיכומים אינם מדויקים	מציאת מודל היודע לסכם קוד בצורה איכותית ולאמן אותו על קוד עם סיכומים איכותיים כך שזה יפחית את הסיכוי לסיכום שאינו מדויק	**גבוה**	**גבוה**
אי עמידה בלוחות זמנים	הגדרת לוחות זמנים בצורה מדויקת והקפדה על שיתוף פעולה על מנת שנתקדם בקצב הכי מהיר שאפשר	**בינוני**	**נמוך**
תלות בסיכומים קיימים	יצירת בסיס נתונים אשר יכיל מגוון רחב של סיכומים מדויקים וכך המודל יוכל ללמוד על מגוון רחב ויצליח להתמודד עם מגוון רחב של קודים.	**גבוה**	**גבוה**
מאגר נתונים מוגבל	הגדלת מאגר הנתונים על ידי חיפוש קוד ויצירה ידנית של סיכום איכותי	**גבוה**	**בינוני**
משאבים בסביבת העבודה	בקשת שרת חזק אשר יצליח להתמודד עם לאמן את המודל על בסיס נתונים רחב	**בינוני**	**גבוה**
הערכת התוצאות	מציאת כלי שיוכל להעריך טקסט על מנת שיעריך את הסיכום אשר המודל מסכם ובנוסף לעבור ידנית על חלק ולבדוק האם הסיכום איכותי	**בינוני**	**גבוה**

5.3 תהליך איסוף הנתונים, הכנת הטבלה

על-מנת להעריך את המודלים ולאפשר ניסויי אחזור מבוקרים, היינו זקוקים למאגר גדול של פונקציות קוד יחד עם תיאור טקסטואלי תואם לכל פונקציה. לאחר בחינת מספר אפשרויות, החלטנו להשתמש במאגר CodeSearchNet כבסיס הנתונים המרכזי. מאגר CodeSearchNet הוא קורפוס ידוע למחקרי אחזור קוד, המכיל אלפי דוגמאות של פונקציות במספר שפות תכנות בצירוף תיעוד או תיאור קצר לכל פונקציה (לרוב תיעוד כפי שנכתב במקור – למשל docstring בפייתון Javadoc ,בג'אווה וכו'). המאגר כולל נתונים בשפות Python, Java, JavaScript, : Ruby, Go ו-PHP, כאשר עבור כל פונקציה או שיטת מחלקה מסופק מלל שמתאר את תפקודה.

5.3.1 התמקדות בשפת Java

במסגרת הפרויקט בחרנו להתמקד בשפת Java לצורך קביעת תחום אחיד לניסויים.
הבחירה בג'אווה נבעה ממספר שיקולים :

- בקוד Java , פונקציות לרוב כוללות *תיעוד מובנה (Javadoc)* כך שיש תיאור יחסית פורמלי של מטרת הפונקציה, פרמטרים, ומה היא מחזירה. תיעוד זה יכול לשמש לנו כשאילתת חיפוש (כלומר, כ"טקסט שמתאר מה הפונקציה עושה").
- ג'אווה שפה סטטית ומבנית, כך שהפונקציות מבודדות יחסית (קל לחלץ פונקציה שלמה עם התיעוד שלה).
- מבחינת נפח CodeSearchNet-Java ,הוא בין התת-מאגרים הגדולים, ולכן *מספק די נתונים לניסוי נרחב*.

5.3.2 עיבוד נתוני CodeSearchNet ויצירת טבלת הקלט

לאחר הורדת נתוני CodeSearchNet עבור Java , בוצע עיבוד לנתונים : סיננו כפילויות, שמרנו רק פונקציות בעלות תיעוד לא ריק, והסרנו תיעוד טריוויאלי מדי (למשל, אם התיאור היה "constructor"בלבד מקרים כאלה לא מועילים). כמו כן, פינינו סימני עיצוב מהתיעוד(דוגמת תגיות HTML או פסאודו-קוד) כדי לקבל טקסט תמציתי ונקי. בסופו של שלב העיבוד יצרנו *טבלת נתונים מרכזית* שבה כל שורה מכילה :

- Code : טקסט הקוד של הפונקציה (בד"כ מספר שורות קוד).
- Documentation : תיאור הפונקציה המקורי (מתוך ה Javadoc, או תגובות המתארות את הפונקציה).
- Summary : עמודה המיועדת לתקציר (סיכום) שניצור באופן אוטומטי עבור אותה פונקציה. בתחילה עמודה זו הייתה ריקה, ומולאה בהמשך על-ידי מודל סיכום.

5.4 עבודה עם מודל CodeT5

מודל Codet5 פותח על בסיס הארכיטקטורה של T5 (Text-to-Text Transfer Transformer), והוא מותאם במיוחד למשימות בתחום הקוד התכנותי. המודל עבר אימון על כמויות עצומות של קוד ממגוון שפות תכנות, תוך שילוב של משימות כגון השלמת קוד, תרגום בין שפות תכנות, הפקת תיעוד אוטומטי, ובעיקר – סיכום קוד.
אחת מהחוזקות המרכזיות של Codet5 היא היכולת להמיר קוד טכני לפסקאות ברורות בשפה טבעית, תכונה שהופכת אותו לכלי אפקטיבי במיוחד בהבנה מהירה של פונקציות וקטעי קוד מורכבים.
לכן, כאשר המטרה היא ניתוח קוד והפקת סיכום נהיר של פעולתו Codet5 מהווה בחירה מצוינת.

5.4.1 איסוף והכנת הנתונים

- שימוש במאגר קוד המכיל פונקציות עם סיכומים איכותיים.
- ארגון וסידור הנתונים כך שיתאימו לשימוש במודל הסיכום ולתהליך אחזור הקוד.

	code	summary
0	code	Summary
1	package apoCommando.entity;\n\nimport java.awt...	The ApoMarioCannon class is a subclass of ApoM...
2	package apoCommando.entity;\n\nimport java.awt...	The ApoMarioCannonCannon class represents a ca...
3	package apoCommando.entity;\n\nimport java.awt...	The ApoMarioCoin class is a subclass of ApoMar...
4	package apoCommando.entity;\n\nimport java.awt...	The ApoMarioCoinParticle class is a subclass o...
5	package apoCommando.entity;\n\nimport java.awt...	The ApoMarioDestructableWall class is a subcla...
6	package apoCommando.entity;\n\nimport java.awt...	The ApoMarioEnd class is a subclass of ApoMari...
7	package apoCommando.entity;\n\nimport java.awt...	The ApoMarioEnemy class represents a enemy in ...
8	package apoCommando.entity;\n\nimport java.awt...	The ApoMarioEntity class represents a single e...
9	package apoCommando.entity;\n\nimport java.awt...	The ApoMarioFireball class represents a fireba...

איור 4- מאגר קוד עם סיכומים איכותיים

5.4.2 כוונון המודל(Fine-Tuning)

- בחירת מודל מתאים לסיכום קוד, לדוגמה CodeT5.
- טעינת בסיס הנתונים לסביבת הפיתוח והגדרת תהליכי הלמידה.
- כוונון המודל על בסיס הנתונים שנאספו כדי לשפר את דיוק הסיכומים.
- ביצוע אימונים חוזרים והערכת ביצועים לשיפור איכות התקצירים.

```
[1] !pip install transformers torch

Show hidden output

from transformers import pipeline, RobertaTokenizer, T5ForConditionalGeneration, TrainingArguments, Trainer
from sentence_transformers import SentenceTransformer, util
import torch
import pandas as pd
import numpy as np
from collections import Counter
from math import sqrt
from google.colab import files
import shutil
```

איור 5- הורדת ספריות מתאימות

```
[ ] train_df = pd.read_csv("Data_Base.csv", header=None)
    train_df.columns = ["code","Summ"]
    train_df
```

איור 6- יצירת בסיס נתונים לאימון

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    tokenizer=tokenizer
)

# Train the model
trainer.train()
```

איור 7- אימון המודל

Epoch	Training Loss	Validation Loss
1	1.951200	1.791750
2	1.322300	1.580822
3	1.092500	1.535374
4	0.880900	1.509531
5	0.886800	1.521667
6	0.754100	1.514355
7	0.560700	1.516414
8	0.599800	1.520550

איור 8- תוצאות האימון

5.4.3 בדיקת איכות הסיכומים

- בדיקה של סיכומי קוד לפני ואחרי כוונון המודל.
- השוואת הסיכומים שנוצרו למודלים קיימים כמו CodeBert ו- T5.
- בדיקה אם הסיכומים אכן משקפים את מטרת ותפקוד הקוד.
- מדידת איכות התקצירים במדדים כגון דיוק, בהירות ורלוונטיות.

index	Code	summary
0	package apoCommando.entity; import java.awt.Graphics2D; import java.awt.Point; import java.awt.geom.Rectangle2D; import java.awt.image.BufferedImage; import java.util.ArrayList; import apoCommando.ApoMarioConstants; import apoCommando.level.ApoMarioLevel; public class ApoMarioCannon extends ApoMarioWall { private BufferedImage iCanon; private int shootTime; private int startTime; private int curShootTime; private ArrayList<ApoMarioCannonCannon> cannons; public ApoMarioCannon(BufferedImage animation, BufferedImage iCanon, float x, float y, int shootTime, int startTime, int id) { super(animation, x, y, animation.getWidth(), animation.getHeight(), 1, 1000000000, id); this.iCanon = iCanon; this.shootTime = shootTime; this.startTime = startTime; this.curShootTime = startTime; } public void init() { super.init(); this.curShootTime = this.startTime; } public ArrayList<ApoMarioCannonCannon> getCannons() { return this.cannons; } public void think(int delta, ApoMarioLevel level) { super.think(delta); this.curShootTime -= delta; if (this.curShootTime <= 0) { this.curShootTime += this.shootTime; this.makeCannon(level); } } public void thinkCanon(int delta, ApoMarioLevel level) { if (this.cannons != null) { for (int i = this.cannons.size() - 1; i >= 0; i--) { this.cannons.get(i).think(delta, level); if (level.getPlayer().isBVisible()) { Point p = this.getMinMax(level, ApoMarioConstants.GAME_WIDTH*2); if ((this.cannons.get(i).getX() < p.x - ApoMarioConstants.GAME_WIDTH*1/2) || (this.cannons.get(i).getX() > p.y + ApoMarioConstants.GAME_WIDTH*1/2) || (!this.cannons.get(i).isBVisible())) { this.cannons.remove(i); break; } } } } } private void makeCannon(ApoMarioLevel level) { Rectangle2D.Float rec = level.getPlayer().getRec(); if ((rec.getX() + rec.getWidth() > this.getX() - ApoMarioConstants.TILE_SIZE) && (rec.getX() < this.getX() + 2 * ApoMarioConstants.TILE_SIZE)) { return; } if (this.cannons == null) { this.cannons = new ArrayList<ApoMarioCannonCannon>(); } BufferedImage cannon = new BufferedImage(this.iCanon.getWidth(), this.iCanon.getHeight(), BufferedImage.TYPE_INT_ARGB); Graphics2D g = cannon.createGraphics(); float vecX = ApoMarioConstants.CANNON_VEC_X; if (level.getPlayer().getX() < this.getX()) { vecX = -vecX; g.drawImage(this.iCanon, 0, 0, this.iCanon.getWidth(), this.iCanon.getHeight(), this.iCanon.getWidth(), 0, 0, this.iCanon.getHeight(), null); } else { g.drawImage(this.iCanon, 0, 0, null); } g.dispose(); this.cannons.add(new ApoMarioCannonCannon(cannon, this.getX(), this.getY(), ++ApoMarioLevel.ID)); this.cannons.get(this.cannons.size() - 1).setVelocityX(vecX); } public void render(Graphics2D g, int changeX, int changeY) { try { super.render(g, changeX, changeY); } catch (Exception ex) { } } }	apomariocannonnonnon (int), or apocomariolevel level, is a tool that can be used by any user. if (this.. cannons) get ((or) above the level, it.
1	package apoCommando.entity; import java.awt.Color; import java.awt.Graphics2D; import java.awt.geom.Rectangle2D; import java.awt.image.BufferedImage; import apoCommando.ApoMarioConstants; import apoCommando.level.ApoMarioLevel; public class ApoMarioCannonCannon extends ApoMarioEnemy { public ApoMarioCannonCannon(BufferedImage animation, float x, float y, int id) { super(animation, x, y, animation.getWidth(), animation.getHeight(), 1, 99999999, ApoMarioConstants.POINTS_ENEMY_CANNON, id); } public void init() { super.init(); } /** * gibt das aktuelle Rechteck der Entity zur&#65533;ck * @return gibt das aktuelle Rechteck der Entity zur&#65533;ck */ public Rectangle2D.Float getRec() { return new Rectangle2D.Float(this.getX(), this.getY() + 4 * ApoMarioConstants.SIZE, this.getWidth(), this.getHeight() - 8 * ApoMarioConstants.SIZE); } /** * gibt das Rechteck f&#65533;r den n&#65533;chsten Schritt der Entity zur&#65533;ck * @return gibt das Rechteck f&#65533;r den n&#65533;chsten Schritt der Entity zur&#65533;ck */ public Rectangle2D.Float getNextRec(int delta) { return new Rectangle2D.Float(this.getX() + delta * this.getVelocityX(), this.getY() + delta * this.getVelocityY() + 4 * ApoMarioConstants.SIZE, this.getWidth(), this.getHeight() - 8 * ApoMarioConstants.SIZE); } public void think(int delta, ApoMarioLevel level) { super.setLevel(level); if (!this.isBVisible()) { return; } if (this.isBDie()) { super.think(delta, level); } else { this.setX(this.getX() + this.getVelocityX() * delta); this.thinkJump(delta, level); } } public void render(Graphics2D g, int changeX, int changeY) { super.render(g, changeX, (int)(changeY - this.getChangeY())); if (ApoMarioConstants.DEBUG) { g.setColor(Color.red); Rectangle2D.Float rec = this.getRec(); g.drawRect((int)(rec.x - changeX), (int)(rec.y - changeY + this.getChangeY()), (int)rec.width, (int)rec.height); } } }	apomariocannoncannonnoncancannon can't be used for any reason. if this is true, it's not the first time that this... isbdie (or..), it ' s

איור 9- סיכומים של CodeBert

index	code	Translated to text	Summary
0	package apoDefence.game; import java.awt.geom.Rectangle2D; import java.awt.image.BufferedImage; import apoDefence.ApoDefenceConstants; import apoDefence.ApoDefencePoint; public class ApoDefenceCastle extends ApoDefenceTower { public ApoDefenceCastle(BufferedImage iTower, int x, int y) { super(-1, iTower, x, y, "Castle"); this.setRec(new Rectangle2D.Double(x, y, iTower.getWidth(), iTower.getHeight())); this.setValues(0, 0, 100, 1, 1, 0, new ApoDefencePoint(0, 0)); this.setHealth(this.getMaxHealth()); this.setMaxLevel(ApoDefenceConstants.TOWER_CASTLE_MAX_LEVEL); this.setUpgradePrice(ApoDefenceConstants.TOWER_CASTLE_PRICE_UPGRADE); } public float getUpgradePrice() { return ApoDefenceConstants.TOWER_CASTLE_PRICE_UPGRADE; } public void upgrade() { if (this.getLevel() < this.getMaxLevel()) { this.setMaxHealth(this.getHealth() + ApoDefenceConstants.TOWER_CASTLE_HEALTH_UPGRADE); this.setLevel(this.getLevel() + 1); this.setBuildUp(true); } } }		
1	package apoDefence.game; import java.awt.Color; import java.awt.Graphics; import java.awt.Graphics2D; import java.awt.Image; import java.awt.Point; import java.awt.geom.Rectangle2D; import java.awt.image.BufferedImage; import java.util.ArrayList; import apoDefence.ApoDefenceConstants; import apoDefence.ApoDefenceEntity; import apoDefence.ApoDefencePoint; public class ApoDefenceEnemy extends ApoDefenceEntity { private final int UP = 0; private final int RIGHT = 1; private final int DOWN = 2; private final int LEFT = 3; private final int RANGE = 200; private BufferedImage iMiniEnemy; private float speedX, speedY; private float maxHealth, healthUpgrade, speed, speedUpgrade, armor, armorUpgrade; private ApoDefencePoint damage; private boolean bShoot; private boolean bHealthVisible; private float shootTime; private int level; private float money; private ArrayList<Point> enemyWay; private ArrayList<ApoDefenceShoot> shoot; private int currentWay, lookWay, countLookWay; private float countWay; private boolean bVisible, bSelect; private boolean bIce, bFire, bLight, bUndead; private float extraTime, extraFire, extraAfterTime; private Rectangle2D Double rec; private Rectangle2D Double rangeRec; private String name; private int enemy; private long startTime; public ApoDefenceEnemy(int enemy, BufferedImage iEnemy, ArrayList<Point> enemyWay) { super(iEnemy, -1, -1, iEnemy.getWidth()/3, iEnemy.getHeight()/4); this.init(); this.enemy = enemy; this.enemyWay = enemyWay; float minX = 0; float minY = 0; if (this.getWidth() > this.getHeight()) { minX = 89; minY = (float)this.getHeight() / (float)this.getWidth() * 89; } else { minY = 89; minX = (float)this.getWidth() / (float)this.getHeight() * 89; } this.iMiniEnemy = new BufferedImage(89, 89, BufferedImage.TYPE_INT_RGB); Graphics g = this.iMiniEnemy.getGraphics(); BufferedImage sublimage = iEnemy.getSubimage(1 * this.getWidth(), 2 * this.getHeight(), this.getWidth(), this.getHeight()); Image miniMap = sublimage.getScaledInstance((int)minX, (int)minY, Image.SCALE_SMOOTH); g.drawImage(miniMap, (int)(0 + (89 - minX)/2), (int)(0 + (89 - minY)/2), null); g.dispose(); } public void init() { this.speedX = 0.0f; this.speedY = 0.0f; this.currentWay = 1; this.countLookWay = (int)(Math.random() * 20); this.lookWay = this.UP; this.bVisible = true; this.bIce = false; this.bFire = false; this.bLight = false; this.bUndead = false; this.bShoot = true; this.bHealthVisible = false; this.shootTime = -1; this.extraTime = -1; this.extraAfterTime = -1; this.extraFire = 0; this.startTime = System.nanoTime(); this.rec = new Rectangle2D.Double(1, 1, 1, 1); this.rangeRec = new Rectangle2D.Double(this.ge. ().- 100, this.getY() - 100, 200, 200); this.shoot = new ArrayList<ApoDefenceShoot>(); } public void setValues(String name, int level, float health, float healthUpgrade, float speed, float speedUpgrade, float armor, float armorUpgrade, ApoDefencePoint damage, float damageUpgrade, float money) { this.name = name; this.maxHealth = health; this.healthUpgrade = healthUpgrade; this.speed = speed; this.speedUpgrade = speedUpgrade; this.level = level; this.armor = armor; this.armorUpgrade = armorUpgrade; this.damage = damage; this.money = money; float damageX = this.damage.x; float damageY = this.damage.y; for (int i = 1; i < this.level; i++) { this.maxHealth = this.maxHealth + this.healthUpgrade; this.speed = this.speed + this.speed * this.speedUpgrade; this.armor = this.armor + this.armorUpgrade; damageX = damageX + damageX * damageUpgrade; damageY = damageY + damageY * damageUpgrade; } this.maxHealth = this.maxHealth + this.level * 2.5f; float difficulty = 1.0f; if (ApoDefenceConstants.DIFFICULTY == ApoDefenceConstants.DIFFICULTY_EASY) { difficulty = ApoDefenceConstants.DIFFICULTY_EASY_CONST; } else if (ApoDefenceConstants.DIFFICULTY == ApoDefenceConstants.DIFFICULTY_HARD) { difficulty = ApoDefenceConstants.DIFFICULTY_HARD_CONST; } damageX = damageX * difficulty; damageY = damageY * difficulty; this.armor = this.armor * difficulty; this.speed = this.speed * difficulty; this.maxHealth = this.maxHealth * difficulty; this.money = this.money * (3.0f - difficulty); this.damage = new ApoDefencePoint(damageX, damageY); this.health = this.maxHealth; this.setEnemyWay(this.enemyWay); } public ApoDefencePoint getShootPoint() { float x = (float)this.getX(); float y = (float)this.getY(); for (int i = 0; i < 10; i++) { x += this.getSpeedX(); y += this.getSpeedY(); } return new ApoDefencePoint(x, y); } public ArrayList<Point> getEnemyWay() { return this.enemyWay; } public void setEnemyWay(ArrayList<Point> enemyWay) { if ((enemyWay == null) || (enemyWay.size() < 2)) { return; this.enemyWay = enemyWay; this.bVisible = true; int x = this.enemyWay.get(0).x; int y = this.enemyWay.get(0).y; this.setX(x); this.setY(y); this.rec = new Rectangle2D.Double(this.getX(), this.getY(), this.getWidth(), this.getHeight()); this.currentWay = 0; this.getNewWay(this.enemyWay.get(this.currentWay).x, this.enemyWay.get(this.currentWay).y); } protected void getNewWay(int x, int y) { if (this.currentWay + 1 >= this.enemyWay.size()) { this.bVisible = false; return; } this.currentWay += 1; int xDiff = x - this.enemyWay.get(this.currentWay).x; int yDiff = y - this.enemyWay.get(this.currentWay).y; int num = Math.abs(xDiff) + Math.abs(yDiff); float maxSpeed = (float)(this.speed / 100.0f); if (Math.abs(xDiff) > Math.abs(yDiff)) { if (xDiff < 0) { this.lookWay = this.RIGHT; this.speedX = maxSpeed * (Math.abs((float)xDiff / (float)num)); } else { this.lookWay = this.LEFT; this.speedX = -maxSpeed * (Math.abs((float)xDiff / (float)num)); } this.countWay = (int)((float)(Math.abs(xDiff) / maxSpeed) + 1); yDiff = -yDiff; this.speedY = maxSpeed * (Math.abs((float)yDiff / (float)num)); if (yDiff < 0) { this.speedY = -Math.abs(this.speedY); else this.speedY = Math.abs(this.speedY); } else { if (yDiff < 0) { this.lookWay = this.DOWN; this.speedY = maxSpeed * (Math.abs((float)yDiff / (float)num)); //System.out.println("speedY = "+speedY+" = ("+maxSpeed+" num = "+num); } else { this.lookWay = this.UP; this.speedY = -maxSpeed * (Math.abs((float)yDiff / (float)num)); } xDiff = -xDiff; this.countWay = (int)(Math.abs(yDiff) / maxSpeed) + 1; this.speedX = maxSpeed * (Math.abs((float)xDiff / (float)num)); if (xDiff < 0) { this.speedX = -Math.abs(this.speedX); else this.speedX = Math.abs(this.speedX); } //System.out.println("x = "+x+" y = "+y+" newX = "+this.enemyWay.get(this.currentWay).x+" newY = "+this.enemyWay.get(this.currentWay).y); //System.out.println("xDiff = "+xDiff+" yDiff = "+yDiff+"maxSpeed = "+maxSpeed+" speedX = "+this.speedX+" speedY = "+this.speedY); } public boolean isNextStep() { if (this.speedX > 0) { if (this.getX() < this.enemyWay.get(this.currentWay).x) { return false; } else { this.setX(this.enemyWay.get(this.currentWay).x); } } else if (this.speedX < 0) { if (this.getX() > this.enemyWay.get(this.currentWay).x) { return false; } else { this.setX(this.enemyWay.get(this.currentWay).x); } } if (this.speedY > 0) { if (this.getY() < this.enemyWay.get(this.currentWay).y) { return false; } else { this.setY(this.enemyWay.get(this.currentWay).y); } } else if (this.speedY < 0) { if (this.getY() > this.enemyWay.get(this.currentWay).y) { return false; } else { this.setY(this.enemyWay.get(this.currentWay).y); } } return true; } public void think(ApoDefenceGame game) { if (this.enemyWay == null) return; this.rangeRec = new Rectangle2D.Double(this.getX() - this.RANGE/2, this.getY() - this.RANGE/2, this.RANGE, this.RANGE); if (this.extraTime != -1) { this.extraTime -= 5; if (this.extraTime <= 0) { this.extraAfterTime = 200; this.extraTime = -1; this.extraFire = 0; this.bFire = false; this.bIce = false; this.bLight = false; } } if (this.extraAfterTime > 0) this.extraAfterTime -= 5; this.countLookWay = this.countLookWay + 1; if (this.countLookWay >= 20) this.countLookWay = 0; if (this.getHealth() < 1) { this.setHealth(0); this.setBVisible(false); game.setMoney((int)(game.getMoney() + this.getMoney())); } if (this.shoot.size() != 0) { for (int i = this.shoot.size() - 1; i >= 0; i--) { this.shoot.get(i).think(game); if ((this.shoot.get(i).isBVisible()) || (this.getRangeRec().intersects(this.shoot.get(i).getRec()))) { this.shoot.remove(i); } } } } public float getCountWay() { return this.countWay; } public void setCountWay(float countWay) { this.countWay = countWay; } public int getCurrentWay() { return this.currentWay; } public void setCurrentWay(int currentWay) { this.currentWay = currentWay; } public float getArmor() { return this.armor; } public ApoDefencePoint getDamage() { return this.damage; } public float getHealth() { return this.health; } public void setHealth(float health) { this.health = health; } public int getLevel() { return this.level; } public float getSpeed() { return this.speed; } public boolean isBVisible() { return this.bVisible; } public void setBVisible(boolean bVisible) { this.bVisible = bVisible; } protected Rectangle2D.Double getRec() { return this.rec; } public void setRec(Rectangle2D.Double rec) { this.rec = rec; } protected boolean intersects(Rectangle2D.Double rec) { return this.rec.intersects(rec); } public boolean isBSelect() { return this.bSelect; } public void setBSelect(boolean bSelect) { this.bSelect = bSelect; } public String getName() { return this.name; } public BufferedImage getMiniEnemy() { return this.iMiniEnemy; } public float getMaxHealth() { return this.maxHealth; } public float getMoney() { return this.money; } public void setMoney(float money) { this.money = money; } public boolean isBFire() { return this.bFire; } public void setBFire(boolean bFire) { this.bFire = bFire; } public boolean isBIce() { return this.bIce; } public void setBIce(boolean bIce) { this.bIce = bIce; } public boolean isBLight() { return this.bLight; } public void setBLight(boolean bLight) { this.bLight = bLight; } public boolean isBUndead() { return this.bUndead; } public void setBUndead(boolean bUndead) { this.bUndead = bUndead; } public boolean isExtra() { if (((this.bIce) || (this.bFire) || (this.bLight)) && (this.extraAfterTime < 1)) return true; return false; } public float getExtraFire() { return this.extraFire; } public void setExtraFire(float extraFire) { this.extraFire = extraFire; } public float getExtraTime() { return this.extraTime; } public void setExtraTime(float extraTime) { this.extraTime = extraTime; } public boolean isBHealthVisible() { return this.bHealthVisible; } public void setBHealthVisible(boolean bHealthVisible) { this.bHealthVisible = bHealthVisible; } public float getSpeedX() { return this.speedX; } public void setSpeedX(float speedX) { this.speedX = speedX; } public float getSpeedY() { return this.speedY; } public void setSpeedY(float speedY) { this.speedY = speedY; } public ArrayList<ApoDefenceShoot> getShoot() { return this.shoot; } public void setShoot(ArrayList<ApoDefenceShoot> shoot) { this.shoot = shoot; } public Rectangle2D.Double getRangeRec() { return this.rangeRec; } public boolean isBShoot() { return this.bShoot; } public void setBShoot(boolean bShoot) { this.bShoot = bShoot; } public float getShootTime() { return this.shootTime; } public void setShootTime(float shootTime) { this.shootTime = shootTime; } public int getEnemy() { return this.enemy; } public boolean isIn(double x, double y, double xm, double ym) { double dx = (double)(x-xm); double dy = (double)(y-ym); return ((dx/this.RANGE)*(dx/this.RANGE)+(dy/this.RANGE)*(dy/this.RANGE)) <= 0.25; } public long getStartTime() { return this.startTime; } public void setStartTime(long startTime) { this.startTime = startTime; } /** * malt das Objekt * @param g */ public void render(Graphics g) { this.render((Graphics2D)g); } /** * malt das Objekt * @param g */ public void render(Graphics2D g, int x, int y) { if (this.isBVisible()) { if ((this.isBSelect()) || ((this.isBHealthVisible()))) { g.setColor(Color.GREEN); if (this.isBSelect()) g.drawOval((int)(this.getX() + x - this.getWidth()/2), (int)(this.getY() + y - this.getHeight()/2), this.getWidth(), this.getHeight()); double value = ((double)this.health / (double)this.maxHealth); value = value * 50; g.fillRect((int)(this.getX() + x - 25), (int)(this.getY() - this.getHeight()/2 + y - 13), (int)(value, 10); g.setColor(Color.RED); g.fillRect((int)(value - 1 + this.getX() + x - 25), (int)(this.getY() - this.getHeight()/2 + y - 13), 50 - (int)(value, 10); g.setColor(Color.BLACK); g.drawRect((int)(this.getX() + x - 25), (int)(this.getY() - this.getHeight()/2 + y - 13), 49, 10); } /*g.setColor(Color.BLACK); g.drawRect((int)(this.rec.getX() + x), (int)(this.rec.getY() + y), this.getWidth(), this.getHeight());*/ int countLookWay = this.countLookWay / 5; if (countLookWay >= 3) countLookWay = 1; g.drawImage(this.getBackground(), (int)(this.getX() + x - this.getWidth()/2), (int)(this.getY() + y - this.getHeight()/2), (int)(this.getX() + x + this.getWidth()/2), (int)(this.getY() + y + this.getHeight()/2), countLookWay * this.getWidth(), this.lookWay * this.getHeight(), (countLookWay + 1) * this.getWidth(), (this.lookWay + 1) * this.getHeight(), null); for (int i = 0; i < this.shoot.size(); i++) { this.shoot.get(i).render(g, x, y); } /*g.setColor(Color.BLACK); /*g.fillRect((int)this.getX() + x - 1, (int)this.getY() + y - 1, 3, 3);*/ } } }	ApoDefenceEntity - Creating a new instance of an ApoDefenceEntity WITHOUT creating new instances in Physics3D - Terrain3D - Game 3D - Terrain4D - Game 2D - Terrain3D - Game 4D - Terrain3D - Game 4D - Terrain3D - Terrain3D - Game 4D - Terrain3D - Game Game	Physics3D - Terrain3D - Game 2D - Terrain4D - Game 3D - Game 4D - Game 5D - Game 6

איור 10- סיכומים של מודל T5

Analyse a Java game class and return a list of ApoMarioCannon objects that can be used to visualize the game .

איור 11- סיכום של מודל CodeT5 לפני הכוונון

The ApoMarioCannon class is a subclass of ApoMarioWall that represents a cannon in the game. It initializes the cannon with attributes like shootTime, startTime, and id. It is designed to be a subclass of ApoMarioWall, and is designed to be used as a base of the ApoMarioWall class. It inherits from the ApoMarioWall class, and is designed to provide functionality for interacting with cannons on the game

איור 12- סיכום של מודל CodeT5 לאחר כוונון

5.4.4 בניית מנגנון אחזור הקוד

- פיתוח מנגנון חיפוש המתבסס על סיכומי הקוד ולא על הקוד עצמו.
- בדיקת רמת ההתאמה בין השאילתות לסיכומים.
- השוואה בין תוצאות חיפוש מבוססות סיכומים לעומת חיפוש ישיר בקוד.
- מדידה של דיוק אחזור, אחוז התאמות רלוונטיות וזמן חיפוש.

```python
# Function to get embeddings
def get_embedding(text):
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True)
    with torch.no_grad():
        outputs = model(**inputs)
    return outputs.last_hidden_state[:, 0, :].squeeze().numpy()  # Extract CLS token
```

איור 13- הפונקציה הזאת מקבלת טקסט ומחזירה וקטור שמייצג את הטקסט בצורה קומפקטית

```python
# Generate embeddings for Summaries (this is what we compare against)
summary_embeddings = np.array([get_embedding(summary) for summary in df["Summary"].tolist()])
```

איור 14- וקטורים המייצגים את הסיכומים

```python
# Generate embeddings for Queries (docstrings from CodeSearchNet)
query_embeddings = np.array([get_embedding(query) for query in df["Query"].tolist()])
```

איור 15- וקטורים המייצגים את השאילתות

```python
# Function to find top matches for each query
def find_top_matches(query_embedding, summary_embeddings, df, top_n=5):
    distances = [euclidean(query_embedding, emb) for emb in summary_embeddings]  # Compute distances
    top_indices = np.argsort(distances)[:top_n]  # Get indices of top matches

    # Return the top matching code snippets
    return [df.iloc[i]["Code"] for i in top_indices]

# Initialize lists to store evaluation results
mrr_scores = []
recall_at_5 = []
```

איור 16- פונקציה המחזירה את 5 הפונקציות הכי קרובות לשאילתה

5.4.5 תוצאות אחזור של מודל CodeT5

- הכנת וקטורים מהסיכומים ומהתיעוד של הקוד.
- שימוש בתיעודים של הקוד כשאילתות.
- השוואת הוקטורים של השאילתות אל מול הוקטורים של הסיכומים.
- החזרת הקוד המתאים לשאילתה.

Summaries	func_documentation_string	func_code_string
/*	Makes sure the fast-path emits in order.	protected final void
Assemble a concise and accurate summary of the Java code provided by the sources.	Mirrors the one ObservableSource in an Iterable of several ObservableSources that first	@CheckReturnValue
A variant of ambArray that takes an observable sequence of sources and produces a concise and accurate summary of the	Mirrors the one ObservableSource in an array of several ObservableSources that first	@SuppressWarnings("unchecke
This method is designed to generate a concise and accurate summary of the provided Java code. You can pass in multiple Ob	Concatenates elements of each ObservableSource provided via an Iterable sequence into	@SuppressWarnings({
This method performs a concise and accurate summary of the provided Java code. You can pass in multiple sources as an obs	Returns an Observable that emits the items emitted by each of the ObservableSources	@SuppressWarnings({
This method takes an observable sequence of sources and concatenates them into a single Observable. A single ObservableS	Concatenates a variable number of ObservableSource sources.	@SuppressWarnings({
This method takes an observable sequence of sources and concatenates them into one observable, delaying the first error to	Concatenates a variable number of ObservableSource sources and delays errors from any	@SuppressWarnings({
This method takes an observable sequence of sources and concatenates them into a single observable sequence, which is	Concatenates an array of ObservableSources eagerly into a single stream of values.	@CheckReturnValue
Concatenate an observable sequence of Objects or Arrays into a single Observable. A concise summary is	Concatenates an array of ObservableSources eagerly into a single stream of values.	@SuppressWarnings({
This method can be used to generate a concise and accurate summary of the provided Java code. For example,	Concatenates an array of {@link ObservableSource}s eagerly into a single stream of values	@SuppressWarnings({
This method takes an ObservableSource and concatenates multiple observable sequences into a single single observable, del	Concatenates the ObservableSource sequence of ObservableSources into a single	@CheckReturnValue
This method can be used to generate a concise and accurate summary of the provided Java code. You can provide a couple	Concatenates the ObservableSource sequence of ObservableSources into a single	@SuppressWarnings({
Concatenate several Java code to a single Observable. A concise summary is	Concatenates an ObservableSource sequence of ObservableSources eagerly into a single	@SuppressWarnings({
Concatenate several Java code to a single Observable, eagerly prefetching the results. For performance reasons,	Concatenates a sequence of ObservableSources eagerly into a single stream of values.	@SuppressWarnings({
You are a summarization assistant specialized in analyzing Java code. You can use this method to generate a concise and acc	Returns an Observable that emits no items to the {@link Observer} and immediately	@CheckReturnValue
Analyzes Java code and generates a concise and accurate summary of the Java code. For example, if you want to print out a c	Returns an Observable that invokes an {@link Observer}'s {@link Observer#onError	@CheckReturnValue
Asks Java for a concise and accurate summary of the provided Java code. A simple example is to generate a concise summar	Converts an Array into an ObservableSource that emits the items in the Array.	@CheckReturnValue
This method is a specialized method that takes a Java Iterable and returns an Observable that emits	Converts an {@link Iterable} sequence into an ObservableSource that emits the items in the	@CheckReturnValue
This method takes a Java publisher and returns an Observable that emits	Converts an arbitrary Reactive-Streams Publisher into an Observable.	@BackpressureSupport(Backpre
You are a summarization assistant specialized in analyzing Java code. You can use this method to generate a concise and	Returns a cold, synchronous and stateless generator of values.	@CheckReturnValue

MRR	Recall	K
0.164	0.22	5
0.1686	0.255	10
0.1712	0.2917	20
0.1736	0.3717	50

איור 17- תוצאות לדוגמא

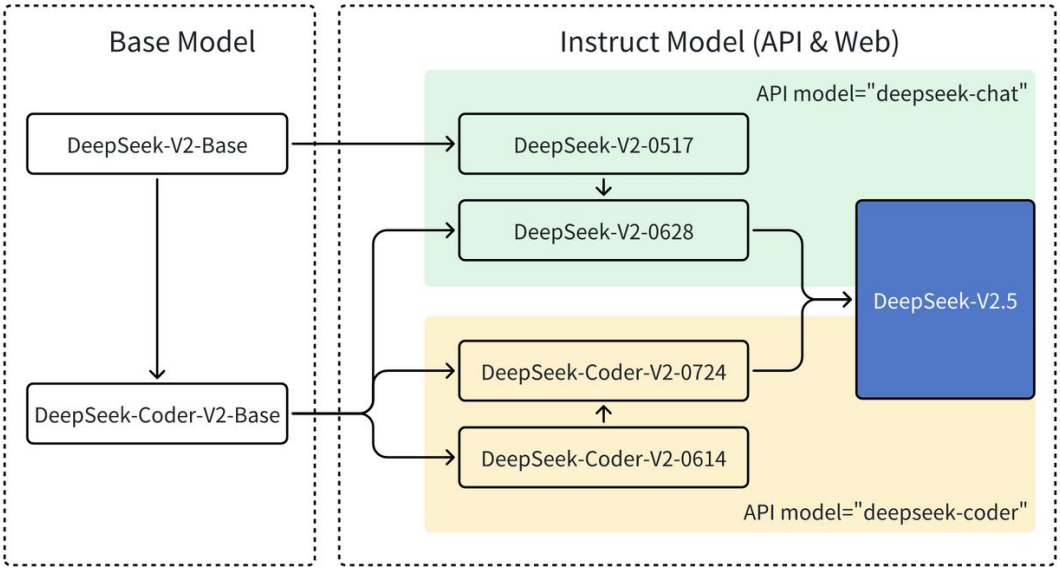
במהלך ההערכה של המודל השתמשנו בשלושה מדדים מרכזיים Recall@K, K ו־ MRR ערך K מציין את מספר הסיכומים המובילים שנבדקו למשל, כאשר K=10, נבדק האם הסיכום הנכון נמצא בין עשרת הסיכומים הראשונים. Recall@K מודד את שיעור ההצלחות של המודל בזיהוי סיכום נכון בתוך אותם K ניסיונות. המדד MRR (Mean Reciprocal Rank) בוחן את המיקום של הסיכום הנכון ברשימת הסיכומים, ומעניק ציון גבוה יותר ככל שהסיכום הנכון מופיע מוקדם יותר.

5.4.6 ניתוח תוצאות ושיפור המודל

- איסוף משוב על איכות הסיכומים והתאמת תוצאות החיפוש.
- ניתוח שגיאות וסטיות כדי לשפר את תהליך כוונון המודל.
- שיפור מערכת האחזור בהתאם לתובנות שהופקו מהבדיקות.
- הסקת מסקנות לגבי היתרונות והחסרונות של הגישה החדשה בהשוואה לשיטות המסורתיות.

5.5 עבודה עם מודל DeepSeek

DeepSeek הוא מודל שפה גדול מתקדם, אשר פותח לאחרונה ונועד להתמודד עם משימות מגוונות בשפה טבעית, כולל קוד תכנותי. בניגוד למודלים ותיקים שהתמקדו בעיקר בטקסט רגיל או בקוד בלבד DeepSeek ,שואף לשלב בין השניים והוא אומן על כמויות עצומות של טקסט וקוד, תוך התאמה למשימות כמו סיכום, תרגום, שאילתות, השלמת קוד ועוד .היתרון המרכזי של DeepSeek נעוץ בגודלו ובארכיטקטורה המתקדמת שלו, המאפשרת לו להבין הקשרים רחבים ולהפיק תוצרים איכותיים גם כאשר המידע מורכב. למרות שהוא אינו מודל ייעודי לקוד בלבד (כמו CodeT5), הוא מציג ביצועים מרשימים גם בסיכום קוד בזכות עושר הנתונים עליהם אומן. במחקר שלנו נעשה שימוש בגרסה של DeepSeek שעברה Fine-Tuning מותאם, ונבדקה יכולתו לייצר סיכומים טקסטואליים מדויקים וברורים לפונקציות קוד. היתרון של DeepSeek מתבטא במיוחד ביכולת שלו לשלב בין הבנה לשונית עמוקה לבין ניתוח מבנה הקוד, ולכן הוא נבחן אצלנו כחלופה פוטנציאלית למודלים הייעודיים.



איור 18- ארכיטקטורת מודל DeepSeek

5.5.1 מימוש טכני ויישום אלגוריתמי

```
# Define multiple API keys (replace with actual keys)
API_KEYS = [
    "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",
    "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",
    "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",
    "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",
    "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",
    "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"
]

# API details
API_URL = "https://openrouter.ai/api/v1/chat/completions"

# Set up API key tracking
api_key_usage = {key: 0 for key in API_KEYS}  # Track requests per key
active_keys = API_KEYS.copy()  # Keeps track of available keys
```

איור 19- שימוש בAPI של DeepSeek

```
# Function to get the next available API key with remaining quota
def get_next_api_key():
    global active_keys
    for key in active_keys:
        if api_key_usage[key] < 150:
            return key
    return None  # If all keys are exhausted, return None
```

איור 20- פונקציה הניגשת למפתח הפנוי הבא

במהלך העבודה עם ממשק ה־APIשל DeepSeek, נתקלנו במגבלה של מספר פניות שניתן לבצע עם כל מפתח (API Key). כל מפתח מוגבל ל־150 פניות בלבד, ולאחר מכן נחסם זמנית לשימוש. כדי להתמודד עם מגבלה זו, בנינו מנגנון ניהול חכם של מפתחות API .

בראשית הקוד הגדרנו רשימת מפתחות (API_KEYS) שבהם נעשה שימוש, כאשר כל מפתח מיוצג כטקסט מוצפן. לאחר מכן, יצרנו מילון (api_key_usage) שמטרתו לעקוב אחרי מספר הפעמים שכל מפתח כבר שימש לשליחת בקשה. במקביל, נבנתה רשימה פעילה (active_keys) המכילה רק את המפתחות שעודם בתוקף ולא מיצו את מגבלת השימוש.

הפונקציה get_next_api_key() אחראית לבחור את המפתח הבא מתוך הרשימה הפעילה. היא בודקת, אחד־אחד, האם יש מפתח שעדיין לא עבר את סף 150 הבקשות. אם נמצא מפתח כזה הוא מוחזר לשימוש. אם כל המפתחות מוצו הפונקציה מחזירה None , כלומר : אין כרגע אפשרות להמשיך את התהליך . גישה זו אפשרה לנו לבצע אלפי פניות ל־ API של DeepSeek באופן רציף, מבלי לעבור על מגבלות השימוש של כל מפתח ומבלי להיתקל בשגיאות מערכתיות עקב חריגות.

```python
def summarize_single_code(code, api_key):
    data = {
        "model": "deepseek/deepseek-r1-distill-llama-70b:free",
        "messages": [
            {
                "role": "user",
                "content": (
                    "Analyze the following Java function and provide an accurate, standalone summary in exactly three sentences. "
                    "Ensure the summary precisely describes this function's purpose, parameters, and key functionality "
                    "without including any unrelated information. The summary should be concise but detailed enough to reflect "
                    "the function's core operations.\n\n"
                    f"Function:\n```java\n{code}\n```"
                )
            }
        ]
    }

    headers = {"Authorization": f"Bearer {api_key}", "Content-Type": "application/json"}
```

איור 21- פרומפט למודל DeepSeek

```python
    try:
        response = requests.post(API_URL, headers=headers, json=data)
        response.raise_for_status()
        summary_text = response.json().get('choices', [{}])[0].get('message', {}).get('content', "").strip()

        # Check if the response is generic
        if not summary_text or "Java code" in summary_text or len(summary_text.split()) < 10:
            print("Warning: Possible generic response, needs manual review.")
            return None

        return summary_text

    except requests.exceptions.HTTPError as http_err:
        print(f"HTTP error occurred: {http_err}")
        return None
    except Exception as e:
        print(f"An error occurred: {e}")
        return None
```

איור 22- יצירת סיכומים 1

פונקציה זו משמשת לשליחת קטע קוד בשפת Java לממשק ה־API של מודל השפה DeepSeek , מתוך מטרה להפיק עבורו סיכום מילולי תמציתי וברור. במהלך הפעולה, הפונקציה יוצרת בקשה בפורמט JSON שבה מצוין המודל הרצוי (deepseek/deepseek-r1-distill-llama-70b) ונוסח הנחיה (prompt) מותאם. ההנחיה מנוסחת באופן חד־משמעי, ומנחה את המודל להחזיר סיכום עצמאי, ממוקד ואינפורמטיבי, בדיוק בשלוש שורות.

הבקשה נשלחת באמצעות פרוטוקול HTTP בשיטה של POST, עם מפתח גישה (API Key) ייחודי שמועבר בכותרות הבקשה. עם קבלת התשובה, הפונקציה מפענחת את תגובת ה־JSON ומחלצת מתוכה את הסיכום שנוצר במידה והוא אכן קיים. הסיכום מוחזר כמחרוזת טקסט רגילה.

בנוסף, קיימת התמודדות עם מצבים שבהם ייתכנו שגיאות כמו חיבור לקוי לשרת, תגובה ריקה או סיכום כללי מדי. במקרים כאלה הפונקציה מחזירה None , כדי לא לפגוע בהמשך התהליך, ולאפשר טיפול נפרד בשורות קוד בעייתיות.

גישה זו מאפשרת הפעלה רחבה של המודל על קטעי קוד רבים, תוך שמירה על יציבות וניטור איכות התוצאות.

```python
def summarize_code_df(df, delay=1):
    df = df.copy()
    summaries = []

    for index, row in tqdm(df.iterrows(), total=len(df), desc="Processing Functions"):
        api_key = get_next_api_key()  # Get an API key with quota left

        if api_key is None:
            print("All API keys exhausted. Stopping execution.")
            break  # Stop processing if all keys are used up

        summary = summarize_single_code(str(row['func_code_string']), api_key)
        summaries.append(summary)
        api_key_usage[api_key] += 1  # Track usage

        # If key reaches 150 requests, remove it from active keys
        if api_key_usage[api_key] >= 150:
            active_keys.remove(api_key)

        time.sleep(delay)  # Avoid hitting rate limits

    df['Summaries'] = summaries
    return df
```

איור 23- יצירת סיכומים 2

הפונקציה נועדה לבצע סיכום אוטומטי לכל שורה במאגר הפונקציות שהכנו מראש, באופן סדרתי ויעיל. מאחר והגישה לממשק ה־ API של DeepSeek מוגבלת במספר הבקשות שניתן לשלוח לכל מפתח (API Key) בפרק זמן מסוים, נבנה מנגנון סבב (Round-Robin) בין מספר מפתחות. בכל איטרציה של הלולאה נבחר מפתח פעיל שטרם מיצה את מגבלת השימוש שלו.

לאחר בחירת המפתח, קטע הקוד הרלוונטי נשלח לפונקציית הסיכום, והתוצאה שנחזירה כלומר, הסיכום המילולי מתווספת לרשימה. במקביל, נספרים מקרי השימוש בכל מפתח, ובמקרה שמפתח מגיע לתקרת השימוש (למשל, 150 בקשות), הוא מוצא מהרשימה לצורך מניעה של קריסת הבקשה או חסימתה מצד השרת.

בנוסף, לצורך שמירה על קצב הגיוני והימנעות מחריגות במדיניות השימוש, הוכנס sleep של שנייה אחת בין קריאה לקריאה. תכנון זה מבטיח שהריצה על מאות שורות הקוד תתבצע בצורה רציפה, מבלי שהממשק החיצוני יסרב לבקשות או יחסום את הגישה.

בסיום, הפונקציה מחזירה טבלה (DataFrame) מעודכנת, שבה לצד כל פונקציה מופיע הסיכום החדש שהופק. טבלה זו משמשת בסיס לניתוח ההשפעה של הסיכומים על יכולת אחזור הקוד בהמשך המחקר.

5.5.2 תוצאות אחזור של מודל

- הכנת וקטורים מהסיכומים ומהתיעוד של הקוד.
- שימוש בתיעודים של הקוד כשאילתות.
- השוואת הוקטורים של השאילתות אל מול הוקטורים של הסיכומים.
- החזרת הקוד המתאים לשאילתה.

Summaries	func_documentation_string	func_code_string
The `fastPathOrderedEmit` method handles the asynchronous emission of a value by checking if the curr	Makes sure the fast-path emits in order.	protected final void fastPathOrderedEmit(U value,
This Java function, `amb`, creates an Observable that uses the ambiguity operator to select the first Obse	Mirrors the one ObservableSource in an Iterable of several	@CheckReturnValue
The `ambArray` function is a static method that takes a varargs of `ObservableSource` instances, returnir	Mirrors the one ObservableSource in an array of several	@SuppressWarnings("unchecked")
This Java function, `concat`, generates an `Observable<T>` that sequentially emits and concatenates iten	Concatenates elements of each ObservableSource provided via	@SuppressWarnings({ "unchecked", "rawtypes" })
This Java function, `concat`, takes an `ObservableSource` emitting `ObservableSource` instances of type	Returns an Observable that emits the items emitted by each of the	@SuppressWarnings({ "unchecked", "rawtypes" })
The `concatArray` function is a utility method that concatenates multiple `ObservableSource` instances se	Concatenates a variable number of ObservableSource sources.	@SuppressWarnings({ "unchecked", "rawtypes" })
The function `concatArrayDelayError` creates an Observable that sequentially concatenates emissions fr	Concatenates a variable number of ObservableSource sources	@SuppressWarnings({ "unchecked" })
This function, `concatArrayEager`, creates an Observable that concatenates emissions from multiple Obs	Concatenates an array of ObservableSources eagerly into a	@CheckReturnValue
The function `concatArrayEager` creates an Observable that emits items from an array of Observable so	Concatenates an array of ObservableSources eagerly into a	@SuppressWarnings({ "rawtypes", "unchecked" })
This function creates an Observable that concatenates and processes multiple ObservableSource instan	Concatenates an array of {@link ObservableSource}s eagerly into	@SuppressWarnings({ "rawtypes", "unchecked" })
The function `concatDelayError` creates an Observable that concatenates and processes emissions from	Concatenates the ObservableSource sequence of	@CheckReturnValue
The Java function `concatDelayError` is a generic method that creates an Observable to sequentially con	Concatenates the ObservableSource sequence of	@SuppressWarnings({ "rawtypes", "unchecked" })
This Java function `concatEager` creates an `Observable` that processes a sequence of nested `Observ	Concatenates an ObservableSource sequence of	@SuppressWarnings({ "unchecked", "rawtypes" })
The `concatEager` function is a generic method that processes an iterable of `ObservableSource` object	Concatenates a sequence of ObservableSources eagerly into a	@SuppressWarnings({ "unchecked", "rawtypes" })
The `empty()` function is a static method that returns an empty Observable of type `T`, which does not emi	Returns an Observable that emits no items to the {@link	@CheckReturnValue
This Java function creates an Observable that emits an error obtained from a provided errorSupplier, a C	Returns an Observable that invokes an {@link Observer}'s {@link	@CheckReturnValue
The `fromArray` function is a static method that generates an `Observable<T>` from a variable-length arra	Converts an Array into an ObservableSource that emits the items	@CheckReturnValue
The `fromIterable` function is a static method that converts an `Iterable` into an `Observable` which emits	Converts an {@link Iterable} sequence into an ObservableSource	@CheckReturnValue
The function `fromPublisher` creates an `Observable` by wrapping a provided `Publisher`, ensuring comp	Converts an arbitrary Reactive-Streams Publisher into an	@BackpressureSupport(BackpressureKind.UNBOUN
The function `generate` is a static method that creates and returns an `Observable<T>` designed to prod	Returns a cold, synchronous and stateless generator of values.	@CheckReturnValue

איור 24- הטבלה עם הסיכומים שיצרנו עם מודל דיפסיק

MRR	Recall	K
0.7246	0.8713	5
0.7299	0.9097	10
0.7315	0.9333	20
0.7321	0.9513	50

במהלך ההערכה של המודל השתמשנו בשלושה מדדים מרכזיים Recall@K ,K ו־ MRR ערך K מציין את מספר הסיכומים המובילים שנבדקו למשל, כאשר K=10, נבדק האם הסיכום הנכון נמצא בין עשרת הסיכומים הראשונים Recall@K. מודד את שיעור ההצלחות של המודל בזיהוי סיכום נכון בתוך אותם K ניסיונות. המדד MRR (Mean Reciprocal Rank) בוחן את המיקום של הסיכום הנכון ברשימת הסיכומים, ומעניק ציון גבוה יותר ככל שהסיכום הנכון מופיע מוקדם יותר.

5.6 עבודה עם מודל LLaMA

מודל LLaMA ראשי תיבות של Large Language Model Meta AI , הוא משפחת מודלים שפותחה על ידי Meta לשעבר Facebook ונחשבת כיום לאחת ממערכות הבינה המלאכותית המובילות בתחום השפה הטבעית. מדובר במודל רב־עוצמה שתוכנן להיות פתוח, קל יחסית לפריסה (בהשוואה למודלים כמו GPT-3), וגמיש להתאמה אישית עבור משימות שונות.

למרות ש־ LLaMA לא תוכנן מלכתחילה עבור קוד תכנותי, גרסאות מתקדמות שלו, כמו LLaMA 2 או גרסאות שעברו Fine-Tuning על קוד (כגון CodeLLaMA) , הוכיחו ביצועים מרשימים גם במשימות כמו השלמת קוד, תרגום פונקציות וסיכום בשפה טבעית .

במחקר שלנו נעשה שימוש במודל LLaMA לאחר התאמה לצורך הפקת סיכומים לפונקציות קוד. המודל שולב בתהליך בדיוק כמו שאר המודלים, תוך שימוש באותו API ובאותו מנגנון בקשת סיכום.

היתרון של LLaMA טמון ביכולת ההקשר הרחבה שלו, ובעובדה שהוא אומן על כמויות עצומות של מידע. לכן, גם אם אינו ייעודי לקוד, הוא מצליח להבין את משמעות הפונקציה ברמה גבוהה, ולהפיק סיכום בהיר וכוללני. עם זאת, בניגוד למודלים כמו DeepSeek או CodeT5 הוא פחות רגיש לדקויות תחביריות של קוד, ולכן נצפו גם מקרים שבהם הסיכום היה כללי או פחות מדויק מבחינה פונקציונלית.

5.6.1 תכנון ויישום

היישום של מודל LLaMA בפרויקט התבצע באותו מבנה כמו היישום של מודל DeepSeek גם כאן, נעשה שימוש בממשק API, על בסיס קובץ קוד זהה, כאשר ההבדל היחיד היה החלפת שם המודל שנשלח בבקשה ל־ API . תהליך ההכנה, ניהול מפתחות ה־ API , סבב הבקשות, ושמירת הסיכומים נותרו זהים לחלוטין .

לפיכך, לא הייתה דרישה לכתיבת קוד חדש או לשינויים במבנה העבודה והמודל שולב כחלק ממערך ההשוואה שביצענו, כך שניתן יהיה לנתח את ביצועיו מול מודלים נוספים באותם תנאים בדיוק.

5.6.2 תוצאות אחזור של מודל

- הכנת וקטורים מהסיכומים ומהתיעוד של הקוד.
- שימוש בתיעודים של הקוד כשאילתות.
- השוואת הוקטורים של השאילתות אל מול הוקטורים של הסיכומים.
- החזרת הקוד המתאים לשאילתה.

Summaries	func_documentation_string	func_code_string
The `fastPathOrderedEmit` function is designed to handle the emission of a value of type `U` to an observer in a thread-safe m	Makes sure the fast-path emits in order.	protected final void
This Java function, `amb`, is a static method that takes an `Iterable` of `ObservableSource` objects as a parameter, where eac	Mirrors the one ObservableSource in an Iterable of several	@CheckReturnValue
The `ambArray` function is a static method that creates an Observable which emits the items emitted by the first Observable in	Mirrors the one ObservableSource in an array of several	@SuppressWarnings("uncheck
The provided Java function, `concat`, is a static method that takes an `Iterable` of `ObservableSource` objects as input, where	Concatenates elements of each ObservableSource provided	@SuppressWarnings({
This Java function, `concat`, is a static method that takes two parameters: `sources`, an `ObservableSource` of `ObservableSo	Returns an Observable that emits the items emitted by each of	@SuppressWarnings({
The `concatArray` function is a Java method that combines multiple `ObservableSource` objects into a single `Observable`, er	Concatenates a variable number of ObservableSource	@SuppressWarnings({
This Java function, `concatArrayDelayError`, is a static method that takes a variable number of `ObservableSource` objects as	Concatenates a variable number of ObservableSource	@SuppressWarnings({
The `concatArrayEager` function in Java is a static method that creates an `Observable` by concatenating the values from mult	Concatenates an array of ObservableSources eagerly into a	@CheckReturnValue
The `concatArrayEager` function is a Java method that takes in an array of `ObservableSource` objects and returns an `Obser	Concatenates an array of ObservableSources eagerly into a	@SuppressWarnings({
The `concatArrayEagerDelayError` function is a Java method that returns an `Observable` of type `T` by concatenating multiple	Concatenates an array of {@link ObservableSource}s eagerly	@SuppressWarnings({
The Java function `concatDelayError` is a static method that takes an `ObservableSource` of `ObservableSource` objects as a	Concatenates the ObservableSource sequence of	@CheckReturnValue
This Java function, `concatDelayError`, is designed to concatenate a sequence of `ObservableSource` instances, allowing for	Concatenates the ObservableSource sequence of	@SuppressWarnings({
This Java function, `concatEager`, creates an `Observable` that concatenates the `ObservableSource` instances emitted by th	Concatenates an ObservableSource sequence of	@SuppressWarnings({
The `concatEager` function is a static method that creates an Observable from an iterable of ObservableSources, allowing for	Concatenates a sequence of ObservableSources eagerly into	@SuppressWarnings({
This Java function, named `empty`, returns an Observable that completes without emitting any items, serving as a useful tool fo	Returns an Observable that emits no items to the {@link	@CheckReturnValue
This Java function, named `error`, creates an Observable that immediately emits an error when subscribed to, allowing for a cu	Returns an Observable that invokes an {@link Observer}'s	@CheckReturnValue
The `fromArray` function is a Java method that creates an `Observable` from an array of items, where the type of items is repre	Converts an Array into an ObservableSource that emits the	@CheckReturnValue
The Java function `fromIterable` creates an `Observable` sequence from a given `Iterable` source, allowing for the asynchronou	Converts an {@link Iterable} sequence into an	@CheckReturnValue
The provided Java function, `fromPublisher`, is designed to create an `Observable` instance from a given `Publisher` object, a	Converts an arbitrary Reactive-Streams Publisher into an	@BackpressureSupport(Backpr
This Java function generates an Observable that produces a sequence of values based on a provided generator, which is a Co	Returns a cold, synchronous and stateless generator of values.	@CheckReturnValue

איור 25- הטבלה עם הסיכומים שיצרנו עם מודל לאמה

MRR	Recall	K
0.5331	0.7233	5
0.5444	0.8067	10
0.5477	0.8517	20
0.5491	0.895	50

במהלך ההערכה של המודל השתמשנו בשלושה מדדים מרכזיים Recall@K ,K ו־ MRR ערך K מציין את מספר הסיכומים המובילים שנבדקו למשל, כאשר K=10, נבדק האם הסיכום הנכון נמצא בין עשרת הסיכומים הראשונים Recall@K. מודד את שיעור ההצלחות של המודל בזיהוי סיכום נכון בתוך אותם K ניסיונות. המדד MRR (Mean Reciprocal Rank) בוחן את המיקום של הסיכום הנכון ברשימת הסיכומים, ומעניק ציון גבוה יותר ככל שהסיכום הנכון מופיע מוקדם יותר.

פרק 6- תוצאות ראשוניות

בשלב הראשוני של הניסוי, בטרם נבחנו מדדי אחזור הקוד בפועל, ביצענו השוואה איכותית בין הסיכומים שהופקו על ידי שלושת המודלים המרכזיים: CodeT5, DeepSeek ו־LLaMA. ההשוואה נעשתה על בסיס תוצרי סיכום עבור אותה קבוצת פונקציות, תוך בחינה של בהירות, רלוונטיות, ועצמאות הסיכום (כלומר האם הסיכום עומד בפני עצמו ומכיל את המידע הנדרש להבנת תפקיד הפונקציה).

לצורך ההשוואה נבנה קובץ טבלה מסודר שבו לצד כל פונקציית קוד הופיעו שלושת הסיכומים, מה שאיפשר ניתוח חזותי והשוואה ישירה. התרשמות ראשונית העלתה כי מודל DeepSeek מציג סיכומים מפורטים ומדויקים יותר לעומת LLaMA ולעיתים אף עולה על CodeT5 במיוחד כשמדובר בפונקציות מורכבות יותר.

ממצאים ראשוניים אלה שימשו אותנו לקביעת סדר העדיפויות להמשך המחקר: העדפנו להתמקד במודלים שהראו ביצועים טובים יותר בשלב הסיכום, מתוך הנחה שזה יתרום גם לשלב אחזור הקוד.

6.1 תוצאות המודלים

CodeT5:

Summaries	func_documentation_string	func_code_string
/*	Makes sure the fast-path emits in order.	protected final void
Assemble a concise and accurate summary of the Java code provided by the sources.	Mirrors the one ObservableSource in an Iterable of several ObservableSources that first	@CheckReturnValue
A variant of ambArray that takes an observable sequence of sources and produces a concise and accurate summary of the	Mirrors the one ObservableSource in an array of several ObservableSources that first	@SuppressWarnings("unchecke
This method is designed to generate a concise and accurate summary of the provided Java code. You can pass in multiple Ob	Concatenates elements of each ObservableSource provided via an Iterable sequence into	@SuppressWarnings({
This method performs a concise and accurate summary of the provided Java code. You can pass in multiple sources as an obs	Returns an Observable that emits the items emitted by each of the ObservableSources	@SuppressWarnings({
This method takes an observable sequence of sources and concatenates them into a single Observable. A single ObservableS	Concatenates a variable number of ObservableSource sources.	@SuppressWarnings({
This method takes an observable sequence of sources and concatenates them into one observable, delaying the first error to	Concatenates a variable number of ObservableSource sources and delays errors from any	@SuppressWarnings({
This method takes an observable sequence of sources and concatenates them into a single observable sequence, which is	Concatenates an array of ObservableSources eagerly into a single stream of values.	@CheckReturnValue
Concatenate an observable sequence of Objects or Arrays into a single Observable. A concise summary is	Concatenates an array of ObservableSources eagerly into a single stream of values.	@SuppressWarnings({
This method can be used to generate a concise and accurate summary of the provided Java code. For example,	Concatenates an array of {@link ObservableSource}s eagerly into a single stream of values	@SuppressWarnings({
This method takes an ObservableSource and concatenates multiple observable sequences into a single single observable, del	Concatenates the ObservableSource sequence of ObservableSources into a single	@CheckReturnValue
This method can be used to generate a concise and accurate summary of the provided Java code. You can provide a couple	Concatenates the ObservableSource sequence of ObservableSources into a single	@SuppressWarnings({
Concatenate several Java code to a single Observable. A concise summary is	Concatenates an ObservableSource sequence of ObservableSources eagerly into a single	@SuppressWarnings({
Concatenate several Java code to a single Observable, eagerly prefetching the results. For performance reasons,	Concatenates a sequence of ObservableSources eagerly into a single stream of values.	@SuppressWarnings({
You are a summarization assistant specialized in analyzing Java code. You can use this method to generate a concise and acc	Returns an Observable that emits no items to the {@link Observer} and immediately	@CheckReturnValue
Analyzes Java code and generates a concise and accurate summary of the Java code. For example, if you want to print out a c	Returns an Observable that invokes an {@link Observer}'s {@link ObserverforOnError	@CheckReturnValue
Asks Java for a concise and accurate summary of the provided Java code. A simple example is to generate a concise summa	Converts an Array into an ObservableSource that emits the items in the Array.	@CheckReturnValue
This method is a specialized method that takes a Java Iterable and returns an Observable that emits	Converts an {@link Iterable} sequence into an ObservableSource that emits the items in the	@CheckReturnValue
This method takes a Java publisher and returns an Observable that emits	Converts an arbitrary Reactive-Streams Publisher into an Observable.	@BackpressureSupport(Backpre
You are a summarization assistant specialized in analyzing Java code. You can use this method to generate a concise and	Returns a cold, synchronous and stateless generator of values.	@CheckReturnValue

MRR	Recall	K
0.164	0.22	5
0.1686	0.255	10
0.1712	0.2917	20
0.1736	0.3717	50

LLaMA:

Summaries	func_documentation_string	func_code_string
The `fastPathOrderedEmit` function is designed to handle the emission of a value of type `U` to an observer in a thread-safe m	Makes sure the fast-path emits in order.	protected final void
This Java function, `amb`, is a static method that takes an `Iterable` of `ObservableSource` objects as a parameter, where eac	Mirrors the one ObservableSource in an Iterable of several	@CheckReturnValue
The `ambArray` function is a static method that creates an Observable which emits the items emitted by the first Observable in	Mirrors the one ObservableSource in an array of several	@SuppressWarnings("unchecke
The provided Java function, `concat`, is a static method that takes an `Iterable` of `ObservableSource` objects as input, where	Concatenates elements of each ObservableSource provided	@SuppressWarnings({
This Java function, `concat`, is a static method that takes two parameters: `sources`, an `ObservableSource` of `ObservableSo	Returns an Observable that emits the items emitted by each of	@SuppressWarnings({
The `concatArray` function is a Java method that combines multiple `ObservableSource` objects into a single `Observable`, er	Concatenates a variable number of ObservableSource	@SuppressWarnings({
This Java function, `concatArrayDelayError`, is a static method that takes a variable number of `ObservableSource` objects as	Concatenates a variable number of ObservableSource	@SuppressWarnings({
The `concatArrayEager` function in Java is a static method that creates an `Observable` by concatenating the values from mult	Concatenates an array of ObservableSources eagerly into a	@CheckReturnValue
The `concatArrayEager` function is a Java method that takes in an array of `ObservableSource` objects and returns an `Obser	Concatenates an array of ObservableSources eagerly into a	@SuppressWarnings({
The `concatArrayEagerDelayError` function is a Java method that returns an `Observable` of type `T` by concatenating multiple	Concatenates an array of {@link ObservableSource}s eagerly	@SuppressWarnings({
The Java function `concatDelayError` is a static method that takes an `ObservableSource` of `ObservableSource` objects as a	Concatenates the ObservableSource sequence of	@CheckReturnValue
This Java function, `concatDelayError`, is designed to concatenate a sequence of `ObservableSource` instances, allowing for	Concatenates the ObservableSource sequence of	@SuppressWarnings({
This Java function, `concatEager`, creates an `Observable` that concatenates the `ObservableSource` instances emitted by th	Concatenates an ObservableSource sequence of	@SuppressWarnings({
The `concatEager` function is a static method that creates an Observable from an iterable of ObservableSources, allowing for	Concatenates a sequence of ObservableSources eagerly into	@SuppressWarnings({
This Java function, named `empty`, returns an Observable that completes without emitting any items, serving as a useful tool fo	Returns an Observable that emits no items to the {@link	@CheckReturnValue
This Java function, named `error`, creates an Observable that immediately emits an error when subscribed to, allowing for a cu	Returns an Observable that invokes an {@link Observer}'s	@CheckReturnValue
The `fromArray` function is a Java method that creates an `Observable` from an array of items, where the type of items is repre	Converts an Array into an ObservableSource that emits the	@CheckReturnValue
The Java function `fromIterable` creates an `Observable` sequence from a given `Iterable` source, allowing for the asynchronou	Converts an {@link Iterable} sequence into an	@CheckReturnValue
The provided Java function, `fromPublisher`, is designed to create an `Observable` instance from a given `Publisher` object, a	Converts an arbitrary Reactive-Streams Publisher into an	@BackpressureSupport(Backpr
This Java function generates an Observable that produces a sequence of values based on a provided generator, which is a G	Returns a cold, synchronous and stateless generator of values.	@CheckReturnValue

MRR	Recall	K
0.5331	0.7233	5
0.5444	0.8067	10
0.5477	0.8517	20
0.5491	0.895	50

DeepSeek:

Summaries	func_documentation_string	func_code_string
The `fastPathOrderedEmit` method handles the asynchronous emission of a value by checking if the curr	Makes sure the fast-path emits in order.	protected final void fastPathOrderedEmit(U value,
This Java function, `amb`, creates an Observable that uses the ambiguity operator to select the first Obse	Mirrors the one ObservableSource in an Iterable of several	@CheckReturnValue
The `ambArray` function is a static method that takes a varargs of `ObservableSource` instances, returnir	Mirrors the one ObservableSource in an array of several	@SuppressWarnings("unchecked")
This Java function, `concat`, generates an `Observable<T>` that sequentially emits and concatenates iten	Concatenates elements of each ObservableSource provided via	@SuppressWarnings({ "unchecked", "rawtypes" })
This Java function, `concat`, takes an `ObservableSource` emitting `ObservableSource` instances of type	Returns an Observable that emits the items emitted by each of the	@SuppressWarnings({ "unchecked", "rawtypes" })
The `concatArray` function is a utility method that concatenates multiple `ObservableSource` instances se	Concatenates a variable number of ObservableSource sources.	@SuppressWarnings({ "unchecked", "rawtypes" })
The function `concatArrayDelayError` creates an Observable that sequentially concatenates emissions fr	Concatenates a variable number of ObservableSource sources	@SuppressWarnings({ "unchecked" })
This function, `concatArrayEager`, creates an Observable that concatenates emissions from multiple Obs	Concatenates an array of ObservableSources eagerly into a	@CheckReturnValue
The function `concatArrayEager` creates an Observable that emits items from an array of Observable so	Concatenates an array of ObservableSources eagerly into a	@SuppressWarnings({ "rawtypes", "unchecked" })
This function creates an Observable that concatenates and processes multiple ObservableSource instan	Concatenates an array of {@link ObservableSource}s eagerly into	@SuppressWarnings({ "rawtypes", "unchecked" })
The function `concatDelayError` creates an Observable that concatenates and processes emissions from	Concatenates the ObservableSource sequence of	@CheckReturnValue
The Java function `concatDelayError` is a generic method that creates an Observable to sequentially con	Concatenates the ObservableSource sequence of	@SuppressWarnings({ "rawtypes", "unchecked" })
This Java function `concatEager` creates an `Observable` that processes a sequence of nested `Observa	Concatenates an ObservableSource sequence of	@SuppressWarnings({ "unchecked", "rawtypes" })
The `concatEager` function is a generic method that processes an iterable of `ObservableSource` object	Concatenates a sequence of ObservableSources eagerly into a	@SuppressWarnings({ "unchecked", "rawtypes" })
The `empty()` function is a static method that returns an empty Observable of type `T`, which does not emi	Returns an Observable that emits no items to the {@link	@CheckReturnValue
This Java function creates an Observable that emits an error obtained from a provided errorSupplier, a C	Returns an Observable that invokes an {@link Observer}'s {@link	@CheckReturnValue
The `fromArray` function is a static method that generates an `Observable<T>` from a variable-length arra	Converts an Array into an ObservableSource that emits the items	@CheckReturnValue
The `fromIterable` function is a static method that converts an `Iterable` into an `Observable` which emits	Converts an {@link Iterable} sequence into an ObservableSource	@CheckReturnValue
The function `fromPublisher` creates an `Observable` by wrapping a provided `Publisher`, ensuring comp	Converts an arbitrary Reactive-Streams Publisher into an	@BackpressureSupport(BackpressureKind.UNBOUN
The function `generate` is a static method that creates and returns an `Observable<T>` designed to prod	Returns a cold, synchronous and stateless generator of values.	@CheckReturnValue

MRR	Recall	K
0.7246	0.8713	5
0.7299	0.9097	10
0.7315	0.9333	20
0.7321	0.9513	50

פרק 7- מסקנות ראשוניות

לאחר שלב ההשוואה האיכותית בין שלושת המודלים המרכזיים CodeT5, LLaMA ו־ DeepSeek התגבשה מסקנה ברורה באשר להמשך כיוון המחקר. לצורך קבלת החלטה מושכלת, בוצעה הפקה של סיכומים עבור אותה קבוצת פונקציות בכל אחד מהמודלים, תוך שמירה על אחידות במבנה ובתנאי ההרצה. לאחר מכן, הושוו הסיכומים זה לצד זה בטבלה חזותית, הן מבחינה איכותית והן תוך מדידה ראשונית של השפעת הסיכומים על ביצועי אחזור הקוד בפועל.

ניתוח ראשוני של תוצאות האחזור הראה כי סיכומים שהופקו על ידי DeepSeek הניבו את דיוק האחזור הגבוה ביותר הן במדד Recall@k והן במדד MRR בהשוואה למודלים האחרים.
הסיכומים היו עקביים, אינפורמטיביים, וניסחו היטב את כוונת הפונקציה, מה שאִפשר למנוע אחזורי שווא ולשפר את ההתאמה לשאילתות בשפה טבעית. לעומת זאת, הסיכומים שהופקו על ידי LLaMA נטו להיות כלליים מדי, ואילו אלו של CodeT5 אמנם טובים אך פחות יציבים עם שונות בין פונקציות פשוטות למורכבות.

מסקנה זו הובילה אותנו להחלטה קריטית בשלב זה: להתמקד בהמשך המחקר במודל DeepSeek בלבד, הן לצורך הפקת טבלת סיכומים נרחבת (הכוללת מאות ואלפי פונקציות), והן לצורך בדיקת השפעת הסיכומים הללו על מנוע אחזור הקוד שבנינו. בחירה זו לא רק מבוססת על תצפיות אמפיריות, אלא גם עולה בקנה אחד עם המטרה המרכזית של המחקר שיפור השימוש החוזר בקוד באמצעות סיכום איכותי.

הממצאים עד כה מחזקים את ההשערה כי סיכום איכותי מהווה רכיב מהותי בהצלחת אחזור הקוד, וכי למודל המדויק והמתאים יש השפעה מכרעת על הצלחת התהליך כולו.

פרק 8- המשך תוכנית עבודה

לאחר שבשלב הראשוני זיהינו את DeepSeek כמודל המוביל מבחינת איכות הסיכומים ותרומתו לאחזור קוד, עברנו לשלב מתקדם שבו הפקנו סיכומים עבור מאגר רחב ומגוון יותר של פונקציות. סיכומים אלו שימשו אותנו לבניית טבלה מסיבית הכוללת אלפי דוגמאות, מתוך כוונה לבחון באופן מעמיק את השפעת הסיכום על ביצועי אחזור הקוד.

Summary	Query	Code
…	…	…

בשלב זה, בנינו מנגנון אחזור שמתבסס על השוואת וקטורי טקסט הן עבור הסיכומים והן עבור השאילתות בשפה טבעית. ביצענו השוואות באמצעות מספר שיטות להשוואת וקטורים (כגון קוסינוס, מרחק אוקלידי ואחרות), ובדקנו אילו מהן מניבות את תוצאות האחזור הטובות ביותר. התוצאות שנאספו הושוו ישירות לתוצאות הבסיסיות של תחרות COIR, שנעשו ללא שימוש בסיכומים.

```python
# Function to get embeddings
def get_embedding(text):
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True)
    with torch.no_grad():
        outputs = model(**inputs)
    return outputs.last_hidden_state[:, 0, :].squeeze().numpy()  # Extract CLS token
```

איור 26- פונקציה שמקבלת טקסט ומחזירה וקטורים

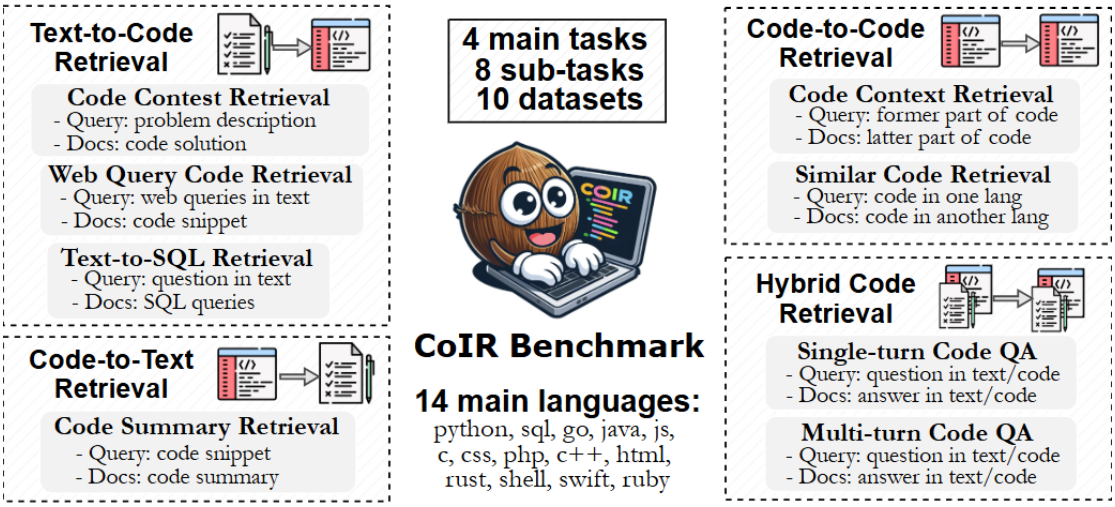
באמצעות הניסוי המורחב הצלחנו להדגים כיצד תוספת של סיכום איכותי משנה את אופי האחזור, משפרת את הדיוק של התוצאות המוחזרות, ולעיתים אף מעלה את הביצועים באופן מובהק. ניתוח זה שימש כבסיס למסקנות המחקריות המרכזיות בפרויקט.

```python
k_values = [1, 3, 5, 10, 100, 1000]

summary_results = []

# Loop over each K and calculate the metrics
for k in k_values:
    print(f"\nEvaluating for Top-{k}...")

    # Initialize lists to store evaluation results
    mrr_scores = []
    recall_at_k = []

    # Evaluate each query
    for i, (query_text, query_emb) in enumerate(zip(df["Query"], query_embeddings)):
        expected_code = df.iloc[i]["Code"]
        top_matches = cosine_top_matches(query_emb, summary_embeddings, df, top_n=k)

        rank = next((i + 1 for i, code in enumerate(top_matches) if code == expected_code), 0)
        mrr_score = 1 / rank if rank > 0 else 0
        mrr_scores.append(mrr_score)

        recall_at_k.append(1 if expected_code in top_matches else 0)

    # Calculate final metrics for this K
    final_mrr = round(np.mean(mrr_scores), 4)
    final_recall = round(np.mean(recall_at_k), 4)

    # Append to summary results
    summary_results.append({
        "K": k,
        "Recall": final_recall,
        "MRR": final_mrr
    })

    print(f"K={k}: Recall={final_recall}, MRR={final_mrr}")

# Convert the summary to a dataframe
summary_df = pd.DataFrame(summary_results)

# Save the summary table to CSV
summary_df.to_csv("cosine_summary_results_by_k.csv", index=False)

# Display the summary
from IPython.display import display
display(summary_df)

print("\nSummary completed! Results saved to 'summary_results_by_k.csv'.")
```

איור 27- פונקציה המחזירה את תוצאות האחזור

CoIR 8.1

תחרות (Code Reuse Challenge) COIR היא תחרות בין־אקדמית שמטרתה לפתח פתרונות חכמים לאחזור קוד מתוך מאגרי פונקציות גדולים, באמצעות שאילתות בשפה טבעית. התחרות מספקת מספר בסיסי נתונים עשירים של פונקציות קוד (לרוב בשפת Java), ומזמינה את המשתתפים לבנות מנועי אחזור מתקדמים, המצליחים להתאים בין שאילתת טקסט לבין פונקציית הקוד המתאימה ביותר מתוך המאגר.



איור 28- תיאור תחרות CoIR

בתחרות משתתפים מגוון רחב של מודלים מתקדמים חלקם מבוססים על שיטות למידת מכונה כלליות, ואחרים מותאמים במיוחד למשימות הקשורות להבנה וייצוג של קוד. כל מודל מנסה למפות בצורה מיטבית את השאלה והקוד לייצוגים מספריים (וקטוריים), ולמדוד ביניהם דמיון שיאפשר התאמה מדויקת. הביצועים נבחנים לפי מדדים מקובלים כמו Recall@k ו־MRR, המודדים את יכולת המודל להחזיר את הפונקציה הנכונה בתוך רשימת תוצאות.

Rank	Model	Model Size (Million Parameters)	Apps	CosQA	Synthetic Text2sql	CodeSearchNet	CodeSearchNet-CCR	CodeTrans-Contest	CodeTrans-DL	StackOverFlow QA	CodeFeedBack-ST	CodeFeedBack-MT	Avg
1	Salesforce/SFR-Embedding-Code-2B_R	2000	74.99	36.31	59.0	73.5	85.77	86.63	33.17	90.54	81.15	53.08	67.41
2	CodeSage-large-v2	1300	50.45	32.73	59.78	94.26	78.09	85.27	33.29	79.41	71.32	57.16	64.18
3	Salesforce/SFR-Embedding-Code-400M_R	400	48.57	34.05	58.96	72.53	80.15	75.67	34.85	89.51	78.87	45.75	61.89
4	CodeSage-large	1300	34.16	28.59	57.9	90.58	84.36	80.1	33.45	79.46	66.06	55.72	61.04
5	Voyage-Code-002	-	26.52	29.79	69.26	81.79	73.45	72.77	27.28	87.68	65.35	28.74	56.26
6	E5-Mistral	7000	21.33	31.27	65.98	54.25	65.27	82.55	33.24	91.54	72.71	33.65	55.18
7	E5-Base-v2	110	11.52	32.59	52.31	67.99	56.87	62.5	21.87	86.86	74.52	41.99	50.9
8	OpenAI-Ada-002	-	8.7	28.88	58.32	74.21	69.13	53.34	26.04	72.4	47.12	17.74	45.59
9	BGE-Base-en-v1.5	110	4.05	32.76	45.59	69.6	45.56	38.5	21.71	73.55	64.99	31.42	42.77
10	BGE-M3	567	7.37	22.73	48.76	43.23	47.55	47.86	31.16	61.04	49.94	33.46	39.31
11	UniXcoder	123	1.36	25.14	50.45	60.2	58.36	41.82	31.03	44.67	36.02	24.21	37.33
12	GTE-Base-en-v1.5	110	3.24	30.24	46.19	43.35	35.5	33.81	28.8	62.71	55.19	28.48	36.75
13	Contriever	110	5.14	14.21	45.46	34.72	35.74	44.16	24.21	66.05	55.11	39.23	36.4

איור 29- לוח תוצאות תחרות CoIR

במסגרת המחקר שלנו, בחרנו מספר מודלים שהשיגו תוצאות גבוהות בתחרות COIR והשתמשנו בהם כבסיס להשוואה. במקום לבחון את ביצועיהם רק עם הקוד המקורי, שילבנו סיכומים טקסטואליים אוטומטיים שהופקו באמצעות מודלים ייעודיים לסיכום קוד (כגון CodeT5, DeepSeek ו־LLaMA), ובחנו האם תוספת זו תורמת לשיפור איכות האחזור. כך נוצר שילוב בין מנועי אחזור קיימים חזקים לבין גישה חדשנית של העשרת הקוד בתיאור שפה טבעית לטובת שיפור ההתאמה לשאילתות המשתמש.

```
import coir
from coir.data_loader import get_tasks
from coir.evaluation import COIR
from coir.models import YourCustomDEModel

model_name = "codesage/codesage-large"

# Load the model
model = YourCustomDEModel(model_name=model_name)

# Get tasks for Java code only
# The 'codesearchnet-ccr-java' dataset filters only Java-related code
tasks = get_tasks(tasks=["codesearchnet-ccr-java"])


# Initialize evaluation
evaluation = COIR(tasks=tasks, batch_size=128)

# Run evaluation
results = evaluation.run(model, output_folder=f"{model_name}")
print(results)
```

איור 30- שימוש ב-API של תחרות CoIR

8.2 מודלים הנבחרו להשוואת אחזור קוד

8.2.1 Salesforce/SFR-Embedding-Code-400M_R

מודל ייעודי לאחזור קוד, שפותח על ידי חברת Salesforce. הוא מבוסס על מנגנון של הטמעת קוד (code embedding) ומאומן על מאגר קוד גדול במיוחד בהיקף של 400 מיליון פרמטרים. המודל נבנה כך שימפה קטעי קוד ושאילתות טקסטואליות למרחב וקטורי משותף, ובכך יאפשר לבצע אחזור על בסיס קרבה בין הווקטורים. בזכות אימון רחב וממוקד על נתוני קוד, הוא מצטיין באחזור מדויק של פונקציות לפי משמעותן, ומדורג בין המובילים בתחרות COIR במדדים כמו Recall@10.

```
"Recall": {
    "Recall@1": 0.41899,
    "Recall@3": 0.56467,
    "Recall@5": 0.61561,
    "Recall@10": 0.67029,
    "Recall@100": 0.8115,
    "Recall@1000": 0.90379
},
```

איור 31- תוצאות מודל Salesforce בתחרות

8.2.2 BAAI/bge-base-en-v1.5

מודל כללי לעיבוד שפה טבעית שפותח על ידי BAAI (Beijing Academy of Artificial Intelligence), ומתמחה ביצירת הטמעות טקסט כלליות (text embeddings). הוא אינו מתמקד בקוד, אך בזכות איכות ההטמעה שלו, הוא מסוגל לייצג שאילתות בשפה טבעית בצורה מדויקת, מה שיכול לסייע כאשר הן מושוות מול ייצוגים טקסטואליים של פונקציות כמו סיכומים. המודל שימושי במיוחד לצורך אחזור מבוסס טקסט, ולכן משמש כבסיס השוואה מעניין גם בפרויקטים שאינם עוסקים בקוד בלבד.

```
"Recall": {
    "Recall@1": 0.08889,
    "Recall@3": 0.12778,
    "Recall@5": 0.21111,
    "Recall@10": 0.45,
    "Recall@100": 0.88889,
    "Recall@1000": 1.0
},
```

איור 32- תוצאות מודל BAAI בתחרות

8.2.3 intfloat/e5-base-v2

מודל נוסף מהמשפחה של E5 שפותח על ידי הקבוצה intfloat, ומתמחה ביצירת ייצוגים אוניברסליים לטקסטים במגוון תחומים (text embeddings for retrieval).

הגרסה base-v2 היא גרסה מאוזנת בגודל ודיוק, ומתאימה למשימות אחזור טקסט בטווח רחב של הקשרים כולל אחזור קוד כשמשווים מול סיכומים.

הוא אינו מתמחה בקוד כשלעצמו, אך נחשב מודל כללי חזק ובעל ביצועים טובים במגוון משימות אחזור טקסטואלי.

```
"Recall": {
    "Recall@1": 0.48152,
    "Recall@3": 0.64272,
    "Recall@5": 0.69037,
    "Recall@10": 0.74496,
    "Recall@100": 0.87522,
    "Recall@1000": 0.94587
},
```

איור 33- תוצאות מודל Intfloat בתחרות

8.2.4 codesage/codesage-large-v2

מודל ייעודי מתקדם בקנה מידה גדול (Large Model), אשר תוכנן במיוחד למשימות הקשורות לקוד כולל אחזור, סיכום, וניתוח פונקציות . הוא כולל כ־1.3 מיליארד פרמטרים ואומן מראש על כמויות עצומות של קוד ונתוני עזר. בזכות כך הוא מסוגל להבין מבנה קוד, שמות משתנים, והקשרים לוגיים בין שורות קוד Codesage-large-v2 . מדורג בעקביות בין המובילים בתחרויות כמו COIR , ומציג ביצועים גבוהים במיוחד במדדים כמו Recall ו־MRR.

```
"Recall": {
    "Recall@1": 0.45899,
    "Recall@3": 0.59467,
    "Recall@5": 0.66561,
    "Recall@10": 0.72029,
    "Recall@100": 0.8515,
    "Recall@1000": 0.93379
},
```

איור 34- תוצאות מודל Codesage בתחרות

פרק 9- תוצאות

בשלב זה של המחקר, בחנו את השפעת סיכומי הקוד על ביצועי אחזור הקוד בפועל. מטרת הניסוי הייתה לבדוק האם תוספת של סיכום טקסטואלי איכותי משפרת את יכולת ההתאמה בין שאילתה בשפה טבעית לבין פונקציית קוד רלוונטית. לצורך כך, נבנתה טבלה רחבה שכללה פונקציות קוד, הסיכומים שלהן שנוצרו באמצעות מודל (DeepSeek) ושאילתות תואמות, כאשר ההשוואה בוצעה מול מספר מנועי אחזור קוד מתקדמים שנבחרו מתוך תחרות COIR.

את תהליך האחזור ביצענו תוך שימוש בשיטות השוואה בין ייצוגים וקטוריים של הטקסטים כלומר, גם השאילתה וגם הסיכום הומרו לווקטורים, וההתאמה ביניהם נמדדה באמצעות מדדים נפוצים כמו דמיון קוסינוס (cosine similarity) או מכפלת נקודה (dot product).

השוואות אלה אפשרו לנו לדרג את תוצאות האחזור ולנתח את תרומת הסיכום לביצועים בפועל. ממצאים אלה מהווים בסיס לפרק ההשוואות המעמיק שמוצג בהמשך.

9.1 תוצאות המודלים

- מודל SalesforceSFR-Embedding-Code-400M_R

תוצאות המודל שלנו עם DOT			תוצאות המודל שלנו עם Euclidian Distance			תוצאות המודל שלנו עם Cosine Similarity		
K	Recall	MRR	K	Recall	MRR	K	Recall	MRR
1	0.626	0.626	1	0.6247	0.6247	1	0.6235	0.6235
3	0.8185	0.713	3	0.8217	0.7137	3	0.822	0.7132
5	0.8707	0.725	5	0.8712	0.7251	5	0.8713	0.7246
10	0.9092	0.7304	10	0.9097	0.7305	10	0.9097	0.7299
100	0.9619	0.7328	100	0.9627	0.7329	100	0.9624	0.7323
1000	0.9830	0.7328	1000	0.9830	0.7330	1000	0.9827	0.7324

איור 35- תוצאות מודל Salesforse

הניתוח שבוצע על מודל Salesforce SFR-Embedding-Code-400M_R מצביע בבירור על שיפור משמעותי בביצועי אחזור הקוד לאחר שילוב סיכומי הקוד שהופקו במודל שלנו. בתמונה מוצגות תוצאות השוואת ביצועים בין התוצאות המקוריות של המודל כפי שהופיעו בתחרות COIR, לבין שלוש שיטות השוואת וקטורים Cosine Similarity, Euclidean Distance ו־ DOT שבוצעו על הסיכומים שהפקנו.

ניכר כי כל שלוש השיטות הפיקו תוצאות טובות יותר כמעט בכל ערך של K. לדוגמה, ב־ Recall@10 עמד הביצוע המקורי של המודל בתחרות על 0.6702, בעוד שהתוצאה עם הסיכומים שלנו עלתה ל־0.9097 (Cosine) ו־0.9092 (DOT) . גם במדדים אחרים, כמו Recall@1000 ו־MRR, נרשמו שיפורים מובהקים.

שיפורים אלו מדגישים את התרומה האפשרית של סיכום קוד בשפה טבעית לתהליך אחזור הקוד בכך שהסיכומים יוצרים גשר סמנטי מדויק יותר בין השאילתה לבין המשמעות הפונקציונלית של הקוד, ומאפשרים למנוע האחזור לבצע התאמה טובה יותר . ממצא זה מהווה חיזוק להשערת המחקר שלנו, לפיה סיכום איכותי עשוי לשפר משמעותית את ביצועי האחזור, גם כשמשתמשים במודל מהשורה הראשונה בתחרות.

- מודל intfloate5-base-v2

תוצאות המודל שלנו עם DOT			תוצאות המודל שלנו עם Euclidian Distance			תוצאות המודל שלנו עם Cosine Similarity		
K	Recall	MRR	K	Recall	MRR	K	Recall	MRR
1	0.2472	0.2472	1	0.2541	0.2541	1	0.3288	0.3288
3	0.3936	0.3113	3	0.4336	0.3333	3	0.5181	0.412
5	0.4644	0.3274	5	0.5108	0.351	5	0.5965	0.4299
10	0.5465	0.3386	10	0.6048	0.3636	10	0.6837	0.4415
100	0.7904	0.3489	100	0.8363	0.3734	100	0.8771	0.4498
1000	0.9375	0.3496	1000	0.952	0.3739	1000	0.9622	0.4501

איור 36- תוצאות מודל Intfloat

הערכת ביצועי מודל intfloat/e5-base-v2 מראה גם היא שיפור ניכר כאשר נעשה שימוש בסיכומי קוד בשפה טבעית במקום בקוד המקורי בלבד. בטבלה מוצגות תוצאות המודל המקורי כפי שדווחו בתחרות COIR , מול ביצועי אותו מודל כשהוזנו לו סיכומים שהופקו במודל שלנו, תוך השוואה בשלוש שיטות מדידה וקטוריות:
Cosine Similarity, Euclidean Distance, Dot

ניתן לראות בבירור שבכל שלוש השיטות חל שיפור עקבי כמעט בכל ערך של K. למשל, ב־Recall@10, הביצועים בתחרות עמדו על 0.7449, בעוד שבשימוש בסיכומים עלה הערך ל־0.6837 (Euclidean), 0.6048 (Cosine) ו־0.5465 (DOT). השיפור משמעותי אף יותר ב־Recall@1000 שם התוצאה המקורית הייתה 0.9458, ואילו עם סיכומים התקבלו ערכים של עד 0.9622 (Cosine) ו־0.952 (Euclidean), מה שמעיד על שימור ואף שיפור של ביצועי אחזור בקנה מידה רחב.

למרות שמודל זה אינו מתמחה בקוד, אלא מיועד למשימות אחזור טקסט כללי, ניתן לראות שהוא מפיק תוצאות טובות יותר כאשר נעשה שימוש בסיכום בשפה טבעית. ממצא זה מחזק את הרעיון שסיכום מדויק מצליח לייצר התאמה סמנטית עמוקה יותר לשאילתה, גם כאשר מנוע האחזור אינו ייעודי לקוד. זהו חיזוק נוסף לחשיבותו של סיכום איכותי ככלי משמעותי לשיפור אחזור קוד חוצה מודלים.

- מודל codesagecodesage-large

תוצאות המודל שלנו עם DOT			תוצאות המודל שלנו עם Euclidian Distance			תוצאות המודל שלנו עם Cosine Similarity		
K	Recall	MRR	K	Recall	MRR	K	Recall	MRR
1	0.6137	0.6137	1	0.5279	0.5279	1	0.6137	0.6137
3	0.8067	0.7012	3	0.7757	0.6401	3	0.8058	0.7007
5	0.8572	0.7128	5	0.8384	0.6546	5	0.8573	0.7127
10	0.8841	0.7179	10	0.8863	0.6612	10	0.8935	0.7176
100	0.9443	0.7202	100	0.9425	0.6638	100	0.9443	0.72
1000	0.9672	0.7203	1000	0.9664	0.6639	1000	0.9672	0.7201

איור 37- תוצאות מודל Codesage

המודל codesage/codesage-large הוא אחד המובילים בתחרות COIR , והציג ביצועים מרשימים כבר בגרסתו המקורית, עם Recall@10 של 0.72029 ו־ Recall@1000 של 0.93379. עם זאת, לאחר שילוב סיכומי קוד טקסטואליים שנוצרו באמצעות המודל שלנו, התקבל שיפור נוסף ומשמעותי בביצועי האחזור.

בתמונה מוצגות תוצאות השוואה בין ביצועי המודל בתחרות לבין שלוש שיטות מדידה שבהן נעשה שימוש לאחר הוספת הסיכומים : Cosine Similarity, Euclidean Distance ו־ DOT. בכל אחת מהשיטות, נרשמה עלייה עקבית בערכי ה־ Recall וה־ MRR לאורך כל ערכי K.

לדוגמא, Recall@10 שופר מ־0.72029 בתחרות ל־0.8935, (Cosine) ל־0.8863 (Euclidean) ואף ל־0.9841 (DOT) שיפור מרשים במיוחד. גם Recall@1000, שעמד בתחרות על 0.93379, עלה ל־0.9672 (Euclidean) 0.9664, (Cosine) ו־0.9672.(DOT) .

נתונים אלו מצביעים על כך שגם מודל חזק כמו codesage-large, שנחשב למודל מוביל בתחום, מצליח להרוויח תוספת ביצועים משמעותית כאשר הוא פועל מול סיכומים ברמה גבוהה, המאפשרים התאמה סמנטית מדויקת יותר לשאלות המשתמש. ממצאים אלו מחזקים את המסקנה שסיכום טקסטואלי איכותי מהווה תוספת ערכית ממשית, גם במערכות אחזור מתקדמות במיוחד.

- מודל BAAI/bge-base-en-v1.5

תוצאות המודל שלנו עם DOT			תוצאות המודל שלנו עם Euclidian Distance			תוצאות המודל שלנו עם Cosine Similarity		
K	Recall	MRR	K	Recall	MRR	K	Recall	MRR
1	0.2472	0.2472	1	0.2541	0.2541	1	0.3288	0.3288
3	0.3936	0.3113	3	0.4336	0.3333	3	0.5181	0.412
5	0.4644	0.3274	5	0.5108	0.351	5	0.5965	0.4299
10	0.5465	0.3386	10	0.6048	0.3636	10	0.6837	0.4415
100	0.7904	0.3489	100	0.8363	0.3734	100	0.8771	0.4498
1000	0.9375	0.3496	1000	0.952	0.3739	1000	0.9622	0.4501

איור 38- תוצאות מודל BAAI

מודל BAAI/bge-base-en-v1.5 הוא מודל כללי לייצוג טקסטים (text embeddings) שאינו ייעודי לקוד, אך נבחן במחקר שלנו כדי לבדוק האם סיכומים בשפה טבעית יכולים לגשר על הפער בין שאילתות טקסטואליות לבין קוד תכנותי גם כשהמודל עצמו לא אומן על קוד. בבחינה ראשונית, ביצועי המודל כפי שהוצגו בתחרות COIR היו נמוכים יחסית, עם Recall@1 של 0.08889 ו־ Recall@10 של 0.45 בלבד דבר הממחיש את מגבלות המודל באחזור ישיר של קוד.

עם זאת, לאחר שהוזנו לו סיכומים טקסטואליים שנוצרו על ידי המודל שלנו, נרשמה עלייה ברורה בביצועי האחזור בכל ערכי K, ובשלוש שיטות מדידה: Cosine Similarity Euclidean Distance ו־ DOT. לדוגמה Recall@10, עלה מ־0.45 ל־0.6837 (Cosine), ל־ 0.6048 (Euclidean) ול־0.5465 (DOT).

התוצאות מדגימות כי גם מודל כללי שלא נועד למשימת אחזור קוד מסוגל להפיק תועלת ברורה משימוש בסיכומים איכותיים. המשמעות היא שסיכום קוד מדויק לא רק משפר את ביצועי המודלים הייעודיים, אלא אף מאפשר למודלים כלליים להפוך לכלי שימושי בתהליך אחזור קוד במיוחד במקרים בהם אין זמינות של מודלים מתמחים.

פרק 10- מסקנות

- סיכום טקסטואלי תורם לאחזור קוד מדויק יותר- נמצא כי תוספת של סיכומים בשפה טבעית לקוד תכנותי שיפרה את איכות האחזור בכל אחד מהמודלים שנבדקו. העלייה ניכרה במדדי Recall ו־ MRR, והייתה עקבית גם במודלים חזקים כמו CodeSage וגם במודלים כלליים כמו E5 או BGE . הסיכום שימש כ"שכבת תיווך סמנטית" שהביאה להתאמה מדויקת יותר בין שאילתות בשפה טבעית לפונקציות קוד.
- המודל DeeSeek הפיק את הסיכומים האיכותיים ביותר- לאורך כלל הניסויים, מודל DeepSeekהוביל בתוצאות האחזור לאחר הפקת סיכומים. הוא הפיק תיאורים מדויקים, בהירים, ורלוונטיים מה שסייע למנועי האחזור לבצע התאמה טובה יותר בין הסיכום לשאילתה. הבחירה להתמקד בו בניסוי המורחב התבררה כמוצדקת והביאה לתוצאות הגבוהות ביותר.
- מודלים כלליים הפיקו תוצאות טובות יותר עם סיכומים- מודלים שלא יועדו במקור לאחזור קוד כמו BGE או E5 הציגו שיפור ניכר בביצועים כאשר קיבלו סיכומים במקום קוד גולמי. ממצא זה מחזק את הטענה שסיכום איכותי יכול להרחיב את השימושיות של מנועי אחזור כלליים גם לתחום הקוד, ובכך לחסוך את הצורך במודלים ייעודיים בכל מקרה.
- כל שיטות ההשוואה הווקטורית הראו שיפור- ההשוואה בין שלוש שיטות מדידת דמיון וקטורי (Cosine Similarity, Euclidean Distance ו־ Dot Product) הראתה שהשיפור באחזור נשמר בכל השיטות, גם אם הייתה שונות מסוימת בין הביצועים. המשמעות היא שהאפקטיביות של הסיכום אינה תלויה בשיטה מסוימת אלא נובעת מאיכותו של הסיכום עצמו.
- סיכום איכותי מגשר על פערים בין שפת המשתמש לקוד- הפער הסמנטי הקיים בין שאילתות בשפה טבעית לבין קוד תכנותי מהווה אתגר באחזור. הסיכום בשפה טבעית מאפשר לגשר על פער זה, על ידי הצגת משמעות הפונקציה במונחים קרובים לאופן שבו משתמשים חושבים ומנסחים שאלות. זהו יתרון קריטי במערכות אחזור במערכי קוד גדולים.

שילוב בין סיכום טקסטואלי איכותי לבין מנוע אחזור קוד מהווה גישה מבטיחה שיכולה לשפר משמעותית את נגישות המידע והקוד במאגרים קיימים. פרויקט זה מציע תשתית ראשונית שניתן לפתח ממנה פתרונות מעשיים לשימוש חוזר בקוד, חיפוש חכם במאגרים גדולים, ותמיכה במפתחים בתחנות העבודה היומיומית.

פרק 11- ביבליוגרפיה

[1] X. Lin, W. Wang, Y. Li, S. Yang, F. Feng, Y. Wei, and T.-S. Chua. Data-efficient fine-tuning for LLM-based recommendation. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 10, ACM, Washington, DC, USA, 2024. DOI: https://dl.acm.org/doi/abs/10.1145/3626772.3657807

[2] Y. Ge, W. Hua, K. Mei, J. Ji, J. Tan, S. Xu, Z. Li, and Y. Zhang. OpenAGI: When LLM meets domain experts. In Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS), pages 3, NeurIPS, 2023. https://proceedings.neurips.cc/paper_files/paper/2023/hash/1190733f217404edc8a7f4e15a57f301-Abstract-Datasets_and_Benchmarks.html

[3] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers. Using an LLM to help with code understanding. In Proceedings of the 46th International Conference on Software Engineering (ICSE), pages 1-13, ACM, Lisbon, Portugal, 2024. https://dl.acm.org/doi/abs/10.1145/3597503.3639187

[4] S. I. Ross, F. Martinez, S. Houde, M. Muller, and J. D. Weisz. The Programmer's Assistant: Conversational interaction with a large language model for software development. In Proceedings of the 28th International Conference on Intelligent User Interfaces (IUI), pages 24, ACM, Sydney, NSW, Australia, 2023. https://dl.acm.org/doi/abs/10.1145/3581641.3584037

[5] S. Haefliger, G. von Krogh, and S. Spaeth. Code Reuse in Open Source Software. *Management Science*, 54(1), pages 180–193, 2008. 10.1287/mnsc.1070.0748.

[6] E. A. AlOmar, A. Venkatakrishnan, M. W. Mkaouer, C. D. Newman, and A. Ouni. How to Refactor this Code? An Exploratory Study on Developer-ChatGPT Refactoring Conversations. *Proceedings of the 21st International Conference on Mining Software Repositories (MSR 2024)*, Lisbon, Portugal, April 2024. https://dl.acm.org/doi/abs/10.1145/3643991.3645081

[7] E. Bandel, Y. Perlitz, E. Venezian, et al. Unitxt: Flexible, Shareable and Reusable Data Preparation and Evaluation for Generative AI. *IBM Research*, 2024. GitHub: https://github.com/IBM/unitxt. https://arxiv.org/abs/2401.14019

[8] B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prather, and E. A. Santos. Programming Is Hard – Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE 2023)*, pages 7, ACM, Toronto, Canada, March 2023. DOI: 10.1145/3545945.3569759.

[9] D. Liao, S. Pan, X. Sun, et al. A3-CodGen: A Repository-Level Code Generation Framework for Code Reuse with Local-Aware, Global-Aware, and Third-Party-Library-Aware. *arXiv preprint arXiv:2312.05772*, 2024. https://arxiv.org/abs/2312.05772.

[10] N. Janakarajan, I. E. Morales, M. Alberts. Unified Lookup Tables: Privacy-Preserving Foundation Models. In Proceedings of the Conference on Machine Learning and Privacy, OpenReview, 2024. https://openreview.net/pdf?id=1caxcqCiIp.