

אוניברסיטת בר-אילן

**איסוף נתונים לשם מידול משתמשים
בסביבה של מספר משתמשים ומספר אפליקציות**

דני שמעוני

מנחה: דר. גל קמינקא

עבודה זו מוגשת כחלק מהדרישות לשם קבלת תואר מוסמך במחלקה למדעי-המחשב של
אוניברסיטת בר-אילן

תשס"ה

רמת-גן

תודות

תודה לדר. גל קמינקא
על ההכוונה, העצות והסבלנות הרבה
שנדרשה עבור עבודת מחקר זו

תודה לסטודנטים
מבית הספר להנדסה באוניברסיטת בר-אילן
על השתתפותם בניסויים

תוכן העניינים

1.....	תודות
3.....	רשימת טבלאות ואיורים
4.....	תקציר
5.....	פרק 1 - מבוא
10.....	פרק 2 - עבודות קודמות
12.....	פרק 3 - מערכת ה-TMA
13.....	ארכיטקטורה וזרימת מידע במערכת ה-TMA
15.....	פרק 3.1 - תוכנת ה-(HOOK SERVER) CLIENT
15.....	פרק 3.1.1 - טכניקת ה- <i>Hooking</i>
21.....	פרק 3.1.2 - סיכום יתרוגות וחסרוגות שימוש בטכניקת <i>Hooking</i> לביצוע <i>User Modeling</i>
23.....	פרק 3.2 - תוכנת ה-(DATASET BUILDER) SERVER
25.....	פרק 4 - שלב הניסויים
26.....	פרק 4.1 - תיאור ניסוי 1: ICQ-EXPLORER EVENT
28.....	פרק 4.2 - תיאור ניסוי 2: CLASSIFICATION
34.....	פרק 4.3 - תיאור ניסוי 3: PREDICTION
39.....	פרק 5 - סיכום והצעות לכיווני מחקר נוספים
40.....	נספחים
40.....	נספח 1 - קובץ קלט ל WEKA
43.....	נספח 2 - נתונים ותוצאות עבור אלגוריתם IPAM
45.....	רשימת מקורות
46.....	ABSTRACT
47.....	שער אחורי (באנגלית)

רשימת טבלאות ואיורים

14	איור 1 - ארכיטקטורת מערכת ה-TMA
18	קוד חדש עבור פונקציית API (EXTTEXTOUTW)
21	טבלת יתרונות של שיטת ה-HOOKING
22	טבלת חסרונות של שיטת ה-HOOKING
24	איור 2 - דוגמא למסך ראשי של תוכנת ה-SERVER
26	איור 3 - אוטומט לתיאור אירוע: ICQ-EXPLORER
33	איור 4 - קלסיפיקציה תוך שימוש באלגוריתמים שונים
36	איור 5 - חיזוי תוך שימוש באלגוריתם IPAM, כאשר $\text{ALPHA} = 0.2$
36	איור 6 - חיזוי תוך שימוש באלגוריתם IPAM, כאשר $\text{ALPHA} = 0.5$
37	איור 7 - חיזוי תוך שימוש באלגוריתם IPAM, כאשר $\text{ALPHA} = 0.8$
37	טבלת סיכום תוצאות ניסוי IPAM עבור שלושת ה-ALPHA
38	איור 8 - מגמות שינוי כפונקציה של ALPHA

מידול-המשתמש (User-Modeling) הינו תחום מחקר בבינה מלאכותית, הלומד את פרופיל המשתמש, על ידי איסוף נתונים לגבי פעולותיו. ידיעת מאפייני והרגלי עבודתו של המשתמש עשויה לתרום לסינון המידע המוזרם למשתמש או כדי לחזות את פעולותיו העתידיות. קיימות שיטות שונות המאפשרות השגת מידע ממנו נבנה מודל המשתמש. אולם עבודות קודמות מניחות הנחות מקלות בתהליך איסוף הנתונים, הנחות אשר אינן מתאימות לתנאים בעולם האמיתי. למשל, הנחה כי קוד המקור של תוכנה זמין לביצוע שינויים, בעוד שברוב המקרים תוכנות אינן מופצות בליווי קוד המקור שלהן. בנוסף, עבודות קודמות העוסקות במידול-המשתמש מבצעות מידול של משתמש יחיד. כלומר, אין התייחסות לאינטראקציות הקיימות בין המשתמשים.

בעבודה זו נציג מערכת תוכנה חדשה, הנקראת: **TMA: Tracking Multiple Applications and Multiple Users**. מערכת ה-TMA מאפשרת איסוף אוטומטי של מידע לגבי פעולות המשתמש, ללא צורך בביצוע שינויים בקוד מקור של התוכנות. המידע הנאסף בנוי מיחידות של פונקציות **API: Application Programming Interface**. מערכת ה-TMA יכולה לעקוב אחר כל אפליקציה הפועלת בסביבת מערכת הפעלה **Windows**. מערכת ה-TMA נבנתה כך שתוכל לאסוף בזמן אמת מידע ממספר משתמשים העובדים במקביל, ומכאן תרומתה לביצוע מידול משתמש הנמצא בקבוצה.

כדי לבחון את איכות המידע הנאסף על ידי מערכת ה-TMA, ביצענו שלושה ניסויים. בניסוי הראשון בחנו האם רצפים מוגדרים של פקודות ה-API יכולים לייצג תסריטים בעלי משמעות בנוגע לפעולות המשתמש. בניסוי השני בחנו האם רצפים פקודות API עשויים לשמש כקלט עבור אלגוריתמים ל **Classification**. בניסוי השלישי בחנו האם מערכת ה-TMA מאפשרת מעקב אחר זוגות משתמשים, כאשר הנתונים שנאספו בניסוי שימשו כקלט עבור אלגוריתם ל **Prediction**.

מידול-המשתמש (User-Modeling) עוסק בתהליך של איסוף ועיבוד מידע בנוגע למשתמש במחשב: הידע אשר אותו המשתמש מביא עימו, הרגלי העבודה שלו, מטרותיו והמשימות אשר אותן הוא מבצע. בהינתן ידע כזה, ניתן לשפר את התנהגות האפליקציות השונות עבור משתמשים שונים.

קיימות שיטות שונות המאפשרות השגת מידע ממנו נבנה מודל המשתמש. השיטה הפשוטה ביותר היא השיטה ה"מפורשת". בשיטה המפורשת קיימת דרישה לאינטראקציה עם המשתמש. לדוגמא, בקשה מהמשתמש לענות על מספר שאלות קבועות מראש. שאלות כגון: גיל, שפה, ידיעת שפות תכנות, ועוד. חסרון השיטה הוא בהיות הנתונים הנאספים סטטיים בנוגע לשינויים בהעדפות ובהרגלי המשתמש. ולכן על מנת לשמור על מידע עדכני, נדרשת אינטראקציה חוזרת ונשנית עם המשתמש.

על מנת לאסוף מידע לגבי פרופיל המשתמש, יש חשיבות לזהות את הפעולות החוזרות אשר מתבצעות על ידי משתמש המחשב. דוגמאות לפעולות כאלו הן: התוכנות אותן מפעיל המשתמש מדי יום, אתרי האינטרנט אליהם נכנס המשתמש, המילים אותן מחפש המשתמש במנועי החיפוש, רצף המילים אותם מקיש המשתמש במעבד התמלילים ועוד.

מידע אודות רצף פעולות כאלו משמש כקלט עבור אלגוריתמים שונים, כגון: קלסיפיקציה (Classification) של פרופיל המשתמש, חיזוי (Prediction) פעולותיו העתידיות של המשתמש, סינון מידע (Information Filtering) הרלוונטי למשתמש ועוד. שימוש במידע בנוגע לפרופיל המשתמש עשוי לשפר את ביצועי המחשב, מהיבט המשתמש.

לדוגמא, משתמש א' מבצע במשך רוב הזמן פעולות גלישה באתרי כלכלה ובורסה, בעוד שמשתמש ב' ברוב הזמן מבצע פעולות עריכת קוד וביצוע קומפילציה. הבחנה בנוגע לפרופיל המשתמש, עשויה לעזור בין השאר בביצוע החלטה בדבר סינון המידע אשר מתאים לפרופיל המשתמש, ובכך לחסוך משאבי תקשורת וזמן עיבוד.

במבט על רצף מספיק גדול של פעולות משתמש, ניתן לזהות תבניות של פעולות חוזרות. כלומר ניתן לזהות פעולות חוזרות ברצף פעולות המשתמש, אשר עשויות לתרום לחיזוי פעולתו הבאה של המשתמש, ובכך להכין את מערכת המחשב לתגובות טובות יותר. (בדומה לרעיון טכניקת ה-Cache הנמצאת בחומרה של המחשב, בה קיים ניסיון לזהות אזורים זיכרון אליהם ניגש המחשב יותר פעמים, על מנת להעביר אותם לזיכרון המהיר יותר).

אולם, מרבית המחקרים העוסקים באיסוף מידע עבור מידול-המשתמש מניחים הנחות מקלות בשלב תהליך איסוף המידע. ספציפית, הינן מניחות כי המידע לגבי פעולות המשתמש הינו נגיש וחשוף בפני התוכנה הממדלת את המשתמש. אולם הנחה זו אינה מעשית.

לעומת השיטות המפורשות, קיימות שיטות "אוטומטיות" לאיסוף מידע בדבר פעולות המשתמש. השיטות האוטומטיות נבדלות מהשיטות המפורשות, בכך שהן לא דורשות אינטראקציה עם המשתמש, כלומר הן פועלות באופן עצמאי.

דוגמא לשיטה אוטומטית לאיסוף מידע, היא כתיבה של תסריטים, המאפשרים לשנות את התוכנה כך שתדווח על פעולות המשתמש. אולם שיטה זו אינה ישימה עבור רוב התוכנות, כיון שקיימות מספר מצומצם מאוד של תוכנות התומכות בקלט של תסריט, וגם אלו שקיימות בנויות משפות תסריט בעלות מבנה שונה, כלומר אין תקינה.

שיטה אוטומטית אחרת, היא ביצוע שינויים בקוד המקור של תוכנה, כך שידווחו על הפריטים בתפריטים שנבחרים ביותר ע"י המשתמש, מהן אוצר המילים אותן מקליד המשתמש, מהן אתרי האינטרנט בהם גולש המשתמש ועוד. אומנם שיטה זו תורמת מידע רב התורם לביצוע מידול משתמש, אולם לצערנו מספר רב של התוכנות אינן משוחררות עם קוד המקור שלהן, ולכן מבחינה מעשית לא ניתן לבצע שינויים בקוד. בנוסף, בדרך זו יש צורך לבצע שינויים בכל תוכנה ותוכנה בנפרד, כיון שתוכנות מפותחות ללא קשר ביניהן.

בנוסף, ברוב המחקרים הקודמים, קיים ניסיון לבצע מידול של משתמש יחיד, ללא התייחסות לעובדה כי מרבית משתמשי המחשב מחוברים בתקשורת עם משתמשים נוספים, וקיימות פעולות אינטראקטיביות בין המשתמשים, העשויות לתרום מידע חיוני עבור תהליך ביצוע מידול- המשתמש הנמצא בסביבה של קבוצת משתמשים. לפיכך מרבית המחקרים הקודמים התמקדו בסביבות מסוימות, בהן האיסוף היה פשוט ומוגבל למידע בנוגע לתוכנה אחת, לדוגמא תוכנת גלישה באינטרנט או שורת הפקודות בסביבת Unix.

לכן יש צורך למצוא מכנה משותף עליהן עובדות כל התוכנות, ללא קשר לשפת קוד המקור בו כתובה התוכנה, ללא קשר להימצאותו של קוד המקור, ללא צורך באינטראקציה עם המשתמש וללא צורך בביצוע שינויים בכל תוכנה ותוכנה בנפרד. מכנה משותף כזה יאפשר איסוף מידע לצורך מידול משתמשים העובדים בקבוצה, בכמה תוכנות שונות.

מטרת עבודת מחקר זו היא לבחון פתרונות אפשריים לחסרונות הנ"ל. לשם כך נציג מערכת תוכנה חדשה הנקראת **TMA: Tracking Multiple Applications and Multiple Users**. מערכת TMA פועלת בסביבת מערכת-ההפעלה Windows, ומבצעת מעקב אוטומטי (ללא התערבות המשתמש) אחרי קבוצת משתמשים העובדים על מספר אפליקציות במקביל, תוך כדי איסוף מידע המייצג את "פעולות המשתמש" שבוצעו, כגון פתיחת קובץ, הזזת מיקום העכבר, הקשת תווים ועוד.

המידע הראשוני המתקבל ממערכת ה-TMA הוא רצף של פונקציות **API: Application Programming Interface**. פקודת API מכילה מידע לגבי שם הפונקציה, הפרמטרים, הזמן, שם המחשב ושם המשתמש שביקש לבצע את הפקודה.

מההיבט של פעולות המשתמש, לפקודת API בודדה אין משמעות. על כן הגדרנו רצפים של פונקציות API המייצגים פעולות, למשל רצף פקודות המתאר מתי המשתמש פתח מסמך במעבד התמלילים או למשל משך הזמן שבו עבד המשתמש על תוכנה מסוימת וכדומה.

מערכת ה-TMA מתבססת על טכניקה בשם Hooking, העוקבת ומקליטה את פעולות משתמש כיחידי או משתמשים העובדים בקבוצה. פעולות המשתמש מבוטאות כיחידות אטומיות של קריאות למערכת-ההפעלה באמצעות זימוני פונקציות API סטנדרטיות, המתרחשות במערכת-ההפעלה כתגובה לפעולות המשתמש. ביקשנו לבדוק האם טכניקת ה-Hooking הקיימת בסביבת מערכת-ההפעלה Windows, מצליחה להתגבר על הבעיות הנ"ל, ומהן הבעיות החדשות העולות משימוש בטכניקה זו.

טכניקת ה-Hooking מאפשרת ביצוע מעקב ואף שינוי התנהגותן של פונקציות ה-API המרכיבות את מערכת-ההפעלה, וזאת על-ידי החלפת הפונקציות המקוריות של מערכת הפעלה בפונקציות חדשות של המתכנת. כתוצאה מכך ניתן לקבל מידע לגבי רצף הקשות המקלדת (Key Logger) שביצע המשתמש, תנועת העכבר, קבצים שנפתחו ועוד פעולות רבות. למעשה כל תוכנה חייבת להשתמש בשכבת ה-API על מנת לעבוד בסביבת Windows (גם תוכנות הרצות על פלטפורמה של JAVA, משתמשות בסופו של דבר ב-API של Windows), ולכן כל הפעולות המתבצעות במחשב שקופות מנקודת המבט של שכבת ה-API.

עם זאת, טכניקת ה-Hooking סובלת מהחיסרון של העמסת מערכת-ההפעלה, ומכאן לאיטיות של עבודת המשתמש. לכן יש צורך לתכנן ביצוע Hooking סלקטיבי, כלומר בחירה מוגבלת של פונקציות ה-API. בנוסף, העובדה שה-API הינו המכנה המשותף של כל התוכנות הרצות ב-Windows, גורם לכך שהמידע המתקבל בשיטה זו, הוא גולמי בהשוואה למידע הדרוש לצרכים של מידול המשתמש. לכן פיתחנו מודול נוסף למערכת ה-TMA אשר מאפשר זיהוי רצפים שהגדרנו מראש, אותם רצפים אשר מייצגים מידע חדש לגבי פעולות המשתמשים.

מערכת ה-TMA מורכבת משני מודולים עיקריים:

1. תוכנת Client – תוכנה הפועלת ברקע של כל אחד מהמשתמשים, ומבצעת מעקב והקלטה של פעולות המשתמש ברמת פונקציות API, באמצעות Hooking.
2. תוכנת Server – תוכנה המרכזת את כל המידע הזורם ממחשבי ה-Clients, ומאפשרת ללכוד אירועים המוגדרים מראש. התוכנה גם מייצרת קבצי input מותאמים עבור האלגוריתמים בהם השתמשנו לביצוע Classification ו Prediction.

מערכת ה-TMA פותחה במבנה של ארכיטקטורת Client-Server, ובכך תורמת ליכולת ביצוע מידול של קבוצת משתמשים. כאשר רכיב ה-Client משתמש בטכניקת ה-Hooking לצורך מעקב אחרי פעולות המשתמש.

על מנת להראות כי מערכת ה-TMA תומכת בבניית פרופיל משתמש בסביבות בהן עד כה לא היה ניתן לאסוף מידע על המשתמש, ביצענו מספר ניסויים בשיתוף עם סטודנטים, כאשר מערכת ה-TMA מרכזת את איסוף הנתונים ועיבודם לקראת הכנסתם כקלט עבור אלגוריתמים ידועים של User-Modeling.

כדי לבדוק את יכולת מערכת ה-TMA לזהות רצפים בעלי משמעות, ביצענו ניסוי אשר מזהה רצפים של פקודות ה-API המגיעים ממחשבים של זוג משתמשים. משתמש ראשון שולח כתובת URL (קישור לאתר אינטרנט) למשתמש שני תוך שימוש בתוכנת ICQ (תוכנת מסרים). בתגובה, משתמש שני מפעיל את תוכנת הגלישה באינטרנט עם כתובת ה-URL שקיבל. התסריט מתואר כאוטומט מצבים, כאשר מצבי המעבר הם פונקציה של זיהוי פקודת API חדשה שמגיעה למערכת ה-TMA. הראנו בניסוי זה כי ניתן להשתמש ב-TMA לבצע מעקב אחר תהליכים רבי-משתמשים.

על מנת להראות את תרומת המידע לתהליך ביצוע מידול המשתמש, ביצענו שני ניסויים נוספים: בניסוי הראשון, בדקנו את תרומת המידע שנאסף לביצוע זיהוי הקבוצה אליה שייך המשתמש (סטודנט, עובד מעבדה וכדומה), דהיינו, הרצנו אלגוריתמי Classification. מטרת הניסוי הראשון היא לבדוק את איכות הנתונים הנאספים על ידי מערכת ה-TMA, לשם ביצוע הבחנה בין קבוצות משתמשים מאוכלוסיות שונות.

בניסוי זה השתמשנו במערכת ה-TMA על מנת לאסוף נתונים משתי קבוצות משתמשים. קבוצה ראשונה מורכבת מסטודנטים העובדים במעבדה של מדעי המחשב, והקבוצה השנייה היא סטודנטים ממחלקות שונות העובדים במעבדה המשרתת את כלל אוכלוסיית הסטודנטים באוניברסיטה. בניסוי לא הגבלנו את המשתמשים לביצוע פעולות מסוימות.

במשך מספר ימים מערכת ה-TMA אספה נתונים מקבוצות הניסוי, נתונים אשר מייצגים את פעולותיהם של המשתמשים והמובעות כיחידות של פקודות ה-API של Windows אשר התרחשו בחשבון המשתמש. שימוש באלגוריתם Decision Trees הניב תוצאה של 84.4% ביכולת זיהוי פרופיל המשתמש, ו-90.6% עבור אלגוריתם NNge (Non-Nested Generalized Exemplars). שני האלגוריתמים הפיקו תוצאות טובות בהרבה מהמצופה ב-BASELINE.

בניסוי השני, בדקנו את תרומת המידע שנאסף לביצוע חיזוי הפעולה הבאה (ברמת פקודת API) של קבוצת משתמשים. אספנו מידע על פעולותיהם של שני משתמשים שעבדו במקביל על משימות משותפות, כל משתמש במחשב שלו. בניסוי זה כינסנו זוגות של סטודנטים, והטלנו עליהם ארבע

משימות פשוטות, למשל יצירת גרף של טמפרטורה שבועי, או כתיבת תוכנית לחישוב שורש ריבועי. כל אחד מהמשתמשים מקבל שתי משימות ידועות מראש, ואילו שני המשימות האחרות אינן ידועות לו מראש, אלא מועברות אליו תוך כדי ביצוע הניסוי, תוך שימוש בתוכנת מסרים בין משתמשים. הפעלנו את האלגוריתם (Incremental Probabilistic Action Modeling) IPAM על הנתונים שנאספו על ידי מערכת ה-TMA.

שימוש באלגוריתם IPAM לעיבוד נתונים אלו, הניב תוצאה של 78.8% בחיזוי הפעולה הבאה, עבור הנתונים של קבוצת המשתמשים. עבור ממוצע הנתונים של כל משתמש בנפרד, קיבלנו תוצאה קטנה יותר של 75.8%. תוצאה זו מראה על שיפור איכות המודל על סמך נתונים הנאספים מתהליך של אינטראקציה בין משתמשים.

להלן נסרוק עבודות מחקר העוסקות בתהליך איסוף נתונים עבור בניית פרופיל המשתמש, תוך הדגשה של יתרונות שיטת ה-Hooking עליה מבוססת עבודת זו.

- Maes ו Metral, Lashkari הציעו בעבודתם [LMM] להוסיף פונקציות חדשות לקוד-מקור של תוכנת דואר-אלקטרוני קיימת. החיסרון בגישה זו, היא ההנחה כי קיימת גישה לקוד מקור של התוכנה, מה שרחוק מלהיות נכון בסביבת Windows. היתרון בשיטה זו הוא גם חסרונה. במידה ואכן יש גישה לקוד מקור של התוכנה, הרי שניתן לבצע שינויים המתאימים טוב יותר לצרכים של קבלת מידע לגבי פעולות המשתמש. אולם הנתונים המייצגים את המשתמש נגזרים מכל האפליקציות איתם עובד המשתמש. לכן טכניקה זו המבצעת שינוי קוד-מקור היא לא מעשית, בגלל שתוכנות נכתבות בשפות שונות, על ידי בתי-תוכנה שונים, ולכן קיים קושי לבצע הוספה ושינויים בקוד על מנת התאים לצרכים של מידול המשתמש.

- טכניקת ה-Hooking המוצעת במחקר זה, אינה מצריכה גישה לקוד מקור של התוכנות, כיון שטכניקת ה-Hooking פועלת ברמת הקוד הבינארי, דהיינו עבודה מול הקבצים שעברו הידור, ולא מול קבצי המקור.
- טכניקת ה-Hooking פועלת במרחב כל התוכנות המופעלות במחשב, לכן אין צורך לפתח קוד נוסף לכל תוכנה ותוכנה בנפרד, אלא מספיק פעם אחת לכתוב קוד עבור פונקציית API.

- Henri Lieberman מציע בעבודותיו [HL1] ו-[HL2] להוסיף סקריפט אשר יבצע מעקב אחר פעולות המשתמש. סקריפט הינו קוד חיצוני לתוכנה, המאפשר הגדרת פעולות נוספות לביצוע, ללא צורך בהידור (קומפילציה) חוזר של התוכנה. היתרון בשיטה זו הוא הקלות של כתיבת סקריפט ובכך שאין צורך בנגישות לקוד מקור של התוכנה. אולם הבעיה היא שמספר תוכנות מצומצם מאוד תומך בהרצת סקריפטים, כמו כן לא קיים תקן אחיד למבנה של כתיבת סקריפטים, כך שיש צורך לפתח סקריפט בעל מבנה שונה עבור האפליקציות השונות.

- שיטת ה-Hooking אינה תלויה ביכולת התוכנות להפעיל סקריפטים.
- שיטת ה-Hooking אינה דורשת פיתוח נפרד של סקריפט עבור כל תוכנה בנפרד.

- Poole ו Gorniak מציעים בעבודתם [ONISI] לבצע פעולת Hooking (דהיינו, החלפת פונקציית API המקוריות בפונקציות חדשות) על אפליקציה מסוימת וידועה מראש, הפועלת בסביבת JAVA.

החיסרון בטכניקה המוצעת הוא שיש לבצע פיתוח עבור כל אפליקציה בנפרד. כמו כן, יש לדעת מראש מהן האפליקציות אותן מתכנן המשתמש להפעיל. בעבודה זו הם מוצג האלגוריתם ONISI לחיזוי פעולתו הבאה של המשתמש, תוך כדי ביצוע השוואת התוצאות לאלגוריתם IPAM.

- שיטת ה-Hooking אינה דורשת פיתוח נפרד עבור כל תוכנה בנפרד, אלא ברגע שביצענו שינוי ברמת פקודת API הרי ששינוי זה חל על כל האפליקציות.
- שיטת ה-Hooking אינה דורשת מידע מראש לגבי התוכנות אותן מתכוון המשתמש להפעיל.
- שיטת ה-Hooking אינה תלויה בצורך של התוכנות לסביבת JAVA או כל סביבה אחרת שתהיה בעתיד, כיון שהשיטה פועלת על המכנה המשותף הקטן ביותר ברמת קבצים במערכת-ההפעלה, דהיינו שכבת ה-API.

● Fagot ו Ruvini מציעים בעבודתם [IBHYS] לעקוב אחרי פעולות המשתמש בסביבת Smalltalk על ידי מעקב אחרי תוכנות עריכה, תוכנות ניפוי (debug) וכדומה. החיסרון בשיטה המוצעת היא שישנם פעולות רבות המייצגות את פרופיל המשתמש שלא ניתן להבחין בהן בשיטה זו, פעולות כגון: הרצת סקריפטים, שימוש במקשי קיצור-דרך (Shortcut). בעבודה זו הם מציגים אלגוריתם חדש IBHYS לחיזוי פעולתו הבאה של המשתמש.

- שיטת ה-Hooking מכסה את כל מרחב האפליקציות הפועלות במחשב ואיננה תלויה בסביבה כלשהי.

● Hirsh ו Davison מציעים בעבודתם [IPAM] אלגוריתם חדש לביצוע חיזוי פעולתו הבאה של משתמש המחשב. לצורך הניסוי הם משתמשים במרחב הפקודות שנדרשו על ידי המשתמש בסביבה של ה-Shell של Unix. בעבודה זו לא נדרש צורך בשימוש בטכניקה מיוחדת לאיסוף נתונים, מעבר לשמירה של הפקודות שהוקשו ב-Shell. לכן אין מעקב אחרי פעולות המשתמש שבוצעו בשאר האפליקציות. עם זאת חשיבות עבודה זו למחקרנו, היא הצגת האלגוריתם IPAM לחיזוי, שבו אנו משתמשים בניסוי 3.

- שיטת ה-Hooking מתאימה לכל מרחב התוכנות ולא דווקא לסביבת Shell מסוים

מערכת ה-TMA בנויה משתי תוכנות עצמאיות, כאשר תקשורת TCP מאפשרת את העברת המידע בין התוכנות, ותורמת ליכולת ביצוע מעקב אחר מספר משתמשים במקביל, וזאת בניגוד לכל עבודות המחקר המתוארות ב-[פרק 2 עבודות קודמות](#) יתרון מערכת ה-TMA היא ביכולתה לעקוב אחרי כל הפעולות המתבצעות בכל האפליקציות של המשתמש (מסומן ב-[איור 1](#) כ Current Running Applications), ללא צורך בשינוי או בידע מוקדם לגבי אלו אפליקציות המשתמש יפעיל, ואף זאת בניגוד לכל עבודות המחקר הקודמות.

התוכנה הראשונה היא תוכנת ה-Client, מסומנת ב-[איור 1](#) כ Hook Server App, אשר תפקידה לבצע מעקב אחר המשתמש, ולכן מופעלת בנפרד בכל אחד מהחשבונות של המשתמשים. תוכנת ה-Client עוקבת, מסננת ומקליטה את פעולות שהתרחשו בחשבון המשתמש כתוצאה מבקשות קלט (עכבר, מקלדת) שביצע המשתמש. הפעולות של המשתמש מבוטאות כרצף של פקודות API שהתרחשו בחשבון המשתמש, ואשר מתרחשות כתוצאה מפעולותיו של המשתמש. תוכנת ה-Client אף עוקבת אחר הפעולות המתבצעות בחשבון הלקוח שלא נגרמו כתוצאה ישירה מפעולות קלט המבוצעות ע"י המשתמש (פעולות המתוזמנות אוטומטית, קבלת הודעות דוא"ל או מסרים ועוד), זאת לאור ההנחה כי גם פעולות אלו מייצגות את פרופיל המשתמש.

התוכנה השנייה היא תוכנת ה-Server, מסומנת ב-[איור 1](#) כ Dataset Builder App, אשר תפקידה לבצע ריכוז כל המידע המגיע ממספר תוכנות Clients הפועלים במחשבי משתמשים שונים, ולכן מופעלת על מחשב מרכזי יחיד. תוכנת ה-Server שומרת את המידע הרב הזורם מכל המשתמשים בקבצים יומיים ומתאימה את הפורמט של הנתונים לקלט הנדרש על ידי אלגוריתמי User-Modeling שונים, כפי שמתואר ב-[פרק 4 שלב הניסויים](#).

ארכיטקטורה וזרימת מידע במערכת ה-TMA

כדי להתניע את מערכת ה-TMA, תוכנות ה-Client המפוזרות בין המחשבים של המשתמשים, מבצעות את פעולת אתחול ה-Hooking, דהיינו החלפת פונקציות ה-API הנדרשות בפונקציות API חדשות.

ב-[איור 1](#) ניתן להבחין בזרימת מידע של New Hook Functions מתיבת Hook Server App אל התיבה של Windows OS. כלומר, פונקציות חדשות "נשתלות" במקום הפונקציות המקוריות של מערכת-ההפעלה, ובכך בעצם מועברת השליטה לתוכנת ה-Hook Server ממערכת-ההפעלה, בכל הקשור לאותן פונקציות שהוחלפו. מרגע שנשתלה פונקציה חדשה, כל פעם שתוכנה כלשהי מבצעת זימון של אותה פונקציה, תתבצע קריאה לפונקציה החדשה במקום לפונקציה המקורית של מערכת-ההפעלה.

יש לציין כי קיימות מאות פונקציות API במערכת-ההפעלה Windows, אבל לצורך המחקר החלפנו פונקציות אחדות בלבד, על מנת להראות את העיקרון.

בנוסף, למרות שתיאורטית ניתן לבצע החלפה של כל פונקציות ה-API, הרי שהחלפה כזו אינה מעשית, כיון שמערכת-ההפעלה נכנסת למצב של עומס גדול יותר עם כל החלפה שכזו, מה שמוביל למצב בלתי נסבל מבחינת תגובת המערכת לבקשות הקלט (עכבר, מקלדת) של המשתמש. לתהליך של החלפת מספר פונקציות חדשות במקום הפונקציות המקוריות, נדרשות בממוצע (כפונקציה של איכות חומרת המחשב ועומס התוכנות) מספר שניות קצר, זמן שלא מהווה בעיה מבחינת המשתמש.

לאחר סיום שלב השתלת הפונקציות החדשות, כל תוכנה שתזמן את אחת מהפונקציות שהחלפנו, תגרום לביצוע תוכן הפקודה כפי שנכתבה בתוכנת ה-Client.

זימוני פונקציות שכאלו מתבצעות כתוצאה מהפעילות המתרחשת בחשבון המשתמש וכתגובה לקלט של המשתמש, דהיינו, הזימונים מתרחשים כפונקציה של פעולות המשתמש.

כאשר תוכנת ה-Client מזהה קריאה לאחת מפונקציות ה-API, התוכנה שולחת בתקשורת TCP ל Server את המידע החדש שהתקבל. בנוסף מתבצע גיבוי של אותן נתונים לקובץ מקומי.

אחרי שליחת המידע ל Server, הפונקציות החדשות מבצעות זימון של הפונקציות המקוריות של Windows, וזאת על מנת לאפשר להתנהג בדיוק אות הדבר, לולא היינו מבצעים את החלפת הפונקציות.

מכאן רואים כי פעולת ה-Hooking למעשה הכפילה את העומס על מערכת-ההפעלה, כיון שכל זימון פונקציה שהוחלפה, מבצעת עתה עבודה כפולה, את הפקודות החדשות שהגדרנו ואת הפקודות המקוריות של מערכת-ההפעלה.

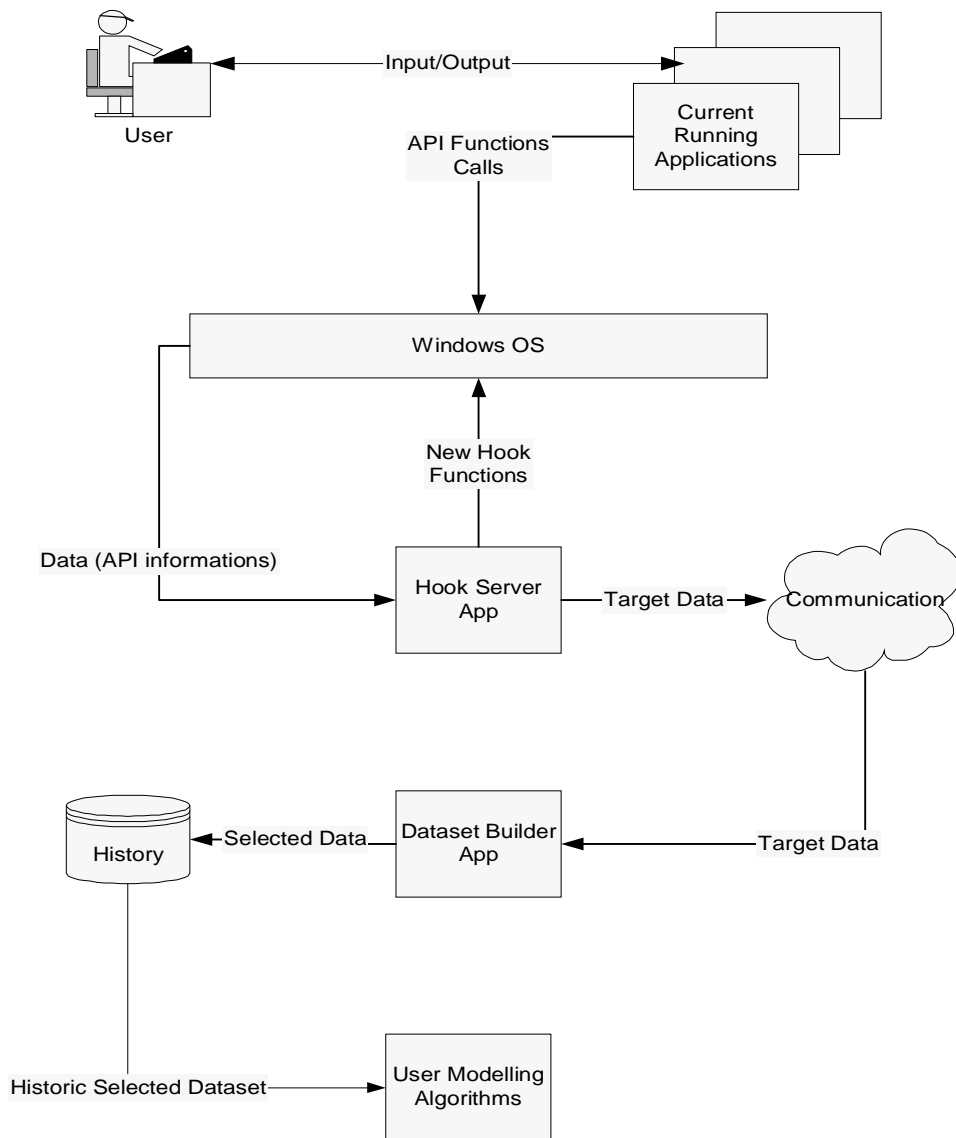
ברוב המקרים המשתמש מזרים את בקשותיו למחשב בעזרת מקלדת ועכבר. פעולות כגון הקלדת מקשים או תנועות העכבר מתבצעים על ידי קריאות לפונקציות API. בנוסף, המחשב מציג הפלט של עיבוד הנתונים למסך.

לכן, כאשר משתמשים בטכניקת ה-Hooking עבור פונקציות API הרלוונטיות (קלט ופלט), ניתן לקבל מידע חשוב לגבי אופי עבודתו של המשתמש.

המידע המסומן ב-[איור 1](#) כ Target Data נשלח בתקשורת לתוכנת ה-Server מכל מחשבי ה-Client.

בתוכנת ה-Server, המידע עובר עיבוד, סינון והתאמה לקראת הכנסתו כקלט עבור אלגוריתמים שונים, כפי שמתואר ב-[פרק 4 שלב הניסויים](#). יש לציין כי תוכנת ה-Server מסוגלת לעבד במקביל הן נתונים היסטוריים השמורים בקובץ, והן את אותם נתונים המגיעים בזמן אמת ממחשבי ה-Client.

להלן איור המתאר את הארכיטקטורה ואת זרימת המידע במערכת ה-TMA:



איור 1 - ארכיטקטורת מערכת ה-TMA

פרק 3.1 - תוכנת ה-Client (Hook Server)

מטרת תוכנת ה-Client היא לעקוב ברקע אחרי פעולות המשתמש, ללא צורך באינטראקציה עם המשתמש. הכוונה ב"פעולות המשתמש" היא לאותן פונקציות API המוזמנות על ידי מערכת-ההפעלה כתוצאה מפעולות קלט של המשתמש, או כתוצאה מתזמון פעולות אוטומטי הנעשה על ידי מערכת-ההפעלה בחשבוננו של המשתמש. תוכנת ה-Client מופעלת בכל אחד מתחנות המשתמשים, ומעבירה בתקשורת TCP את המידע על הפעולות שמצאה למחשב ה-Server.

תוכנת ה-Client פותחה כהמשך לעבודתו של Ivo Ivanov והמתוארת ב [IV]. Ivanov הכין תשתית תכנותית לסביבת Visual C++, המאפשרת ביצוע פשוט של פעולת Hooking על פונקציות ה-API של Windows. עבודתו של Ivanov הותאמה לסביבה של משתמש יחיד, ונשמר מידע מצומצם בדבר פעולת ה-API. בעבודה זו הרחבנו את תשתית עבודתו של Ivanov לסביבה של משתמשים בקבוצה, ע"י הוספת רכיבי תקשורת המעבירים מידע אל תוכנת ה-Server. בנוסף, כדי להתאים את המידע לצרכים של הניסויים שביצענו, הרחבנו את בסיס המידע הנאסף לשדות הבאים: שם המחשב, שם המשתמש, תאריך, שעה, שם הפונקציה והפרמטרים שלה. תוכנת ה-Client מבוססת על טכניקת ה-Hooking לשם ביצוע פעולותיה.

פרק 3.1.1 - טכניקת ה-Hooking

טכניקת ה-Hooking הנתמכת על ידי מערכת-ההפעלה Windows והמתוארת ב [MSDN], מאפשרת למתכנתים להחליף את פונקציות ה-API של מערכת הפעלה בפונקציות חדשות של המתכנתים. פונקציות ה-API הן אבן היסוד של מערכת-ההפעלה. כל אפליקציה אשר מבקשת למשל לפתוח קובץ בסביבה של Windows צריכה לבצע קריאה של פונקציית `OpenFile`¹. בד"כ מתכנת שמבצע פעולת Hook על פונקציה כלשהי, דואג בסוף הפונקציה החדשה שכתב, גם לבצע קריאה של פונקציית ה-API המקורית של Windows, כדי לאפשר המשך התנהגות רגילה של המערכת.

¹ תיאורטית אפשר לבצע את פעולות ה-API על ידי כתיבה ב-Assembler, אלא שבחירה כזו תגבה זמן פיתוח ארוך ומיותר, יחסית לשימוש בקריאה לפקודות API מוכנות ובדוקות

- כדי להחליף את פונקציית API בפונקציה חדשה משתמשים בפונקציה `SetWindowsHookEx`:

```
HHOOK SetWindowsHookEx(int idHook, HOOKPROC lpfn, HINSTANCE hMod, DWORD dwThreadId);
```

הפרמטר `lpfn` הוא הפרמטר החשוב ביותר, כיון שהוא מייצג את המצביע לפונקציה החדשה אותה אנחנו מעוניינים לשתול במקום הפונקציה הקודמת. שם הפונקציה ומספר הפרמטרים חייב להיות זהה לפונקציה הישנה המוחלפת. את הפרמטר החוזר מטיפוס `HHOOK` שומרים כדי לאפשר לזמן שחרור הפונקציה החדשה ממצב של `Hooking`.

- כדי לשחרר פונקציית API ממצב של `Hooking` משתמשים בפונקציה `UnhookWindowsHookEx`:

```
BOOL UnhookWindowsHookEx(HHOOK hhk);
```

הפונקציה מקבלת פרמטר יחיד `hhk` מטיפוס `HHOOK` כדי לשחרר את הפונקציה שהשתלנו. ערך הפרמטר `hhk` התקבל בשלב זימון הפונקציה `SetWindowsHookEx`.

דוגמה לביצוע `Hooking` בסביבת הקוד של מערכת ה-TMA

להלן נדגים כיצד יש לבצע `Hooking` על פונקציית API בסביבת הקוד של מערכת ה-TMA. יש לשים לב כי קוד מערכת ה-TMA מסתיר מהמתכנת את הצורך להשתמש ישירות בפונקציות `SetWindowsHookEx` ו `UnhookWindowsHookEx`, אלא מאפשר שימוש בפונקציות פשוטות יותר, שנציג להלן.

על מנת לבצע `Hooking` לפונקציית ה-API `ExitProcess`, יש להוסיף את שורת הקוד הבאה:

```
sm_pHookMgr->HookImport("kernel32.dll", "ExitProcess", MyExitProcess);
```

הפונקציה `HookImport` מקבלת שלושה פרמטרים:

1. `kernel32.dll` – מייצג את שם הקובץ בו מוגדרת פונקציית ה-API אותה אנו מבקשים להחליף
2. `ExitProcess` – שם פונקציית ה-API
3. `MyExitProcess` – שם הפונקציה החדשה שהגדרנו, שתחליף את הפונקציה המקורית של מערכת-ההפעלה. מספר הפרמטרים וטיפוסיהם חייב להיות זהה לאלו שקיימים בפונקציה המקורית.

מקובל לזמן מתוך הפונקציה החדשה את הפונקציה המקורית של Windows, וזאת על מנת להמשיך לבצע גם את אותן פעולות חיוניות שהוגדרו על ידי מערכת-ההפעלה.

על מנת לשחרר פונקציה שביצענו עליה פעולת Hooking, ולהחזיר את הפונקציה המקורית של Windows לפעילות, יש לבצע את הפעולה הבאה:

```
RemoveHook( pHook );
```

כאשר הפרמטר pHook מייצג את הערך החוזר שקיבלנו בשלב ביצוע ה-Hooking.

להלן דוגמא לשלושת השלבים הנדרשים לצורך ביצוע Hook על פונקצית API (ExtTextOutW):

1. הכנת פונקציה חלופית MyExtTextOutW לפונקציה ExtTextOutW

הפונקציה החדשה MyExtTextOutW חייבת להיות זהה מבחינת מספר הפרמטרים וטיפוסיהם לפונקציה המקורית ExtTextOutW. לכן, בדוגמא הנוכחית, הגדרנו שמונה פרמטרים כאשר טיפוסיהם זהים לטיפוס הפרמטרים בפונקציה ה-API המקורית. סטייה מעיקרון זה, תגרום לפעולות בלי מוגדרות של מערכת-ההפעלה, עד כדי השבתת מערכת-ההפעלה.

הפעולות הראשונות אותן אנו מבצעים בפונקציה MyExtTextOutW הן פעולות אשר יגרמו לזימון הפונקציה המקורית ExtTextOutW, כדי לשמור על פעולתה התקינה של מערכת-ההפעלה. דהיינו, מבחינת התוכנות המזמנות את הפונקציה ExtTextOutW אין הבדל בין אם בוצעה פעולת Hooking על הפונקציה ואם לאו. את הערך המוחזר על ידי הפונקציה המקורית והמציין אם הפונקציה ביצעה את פעולה כמתוכנן, אנו שומרים במשתנה זמני (bResult) על מנת להחזירו מאוחר יותר, אחרי השלב בו אנו נבצע את הפעולות שהוספנו.

לאחר שלב זימון הפונקציה המקורית, אנו מבצעים פעולות מיוחדות המשרתות את צרכינו לביצוע מידול המשתמש, כאשר ברוב המקרים אנו שולחים מידע לגבי האירוע שקרה, דהיינו העובדה כי תוכנה מסוימת זימנה את הפונקציה ExtTextOutW עם הפרמטרים השונים. הפרמטרים במקרה הנוכחי, מספקים מידע לגבי מה נכתב על המסך, ולכן הן נותנים לנו בין השאר את המידע לגבי אלו אתרי אינטרנט המשתמש גלש, מהן התווים אותן מקיש המשתמש ועוד.

יש לציין, כי לפעמים סדר הפעולות הינו הפוך. דהיינו, קודם אנו מבצעים את הפעולות הנדרשות לצורך ביצוע מידול המשתמש, ורק לאחר מכן ביצוע זימון פונקציה ה-API המקורית. לדוגמא, עבור הפונקציה ExitProcess אשר סוגרת תוכנה פועלת. אילו היינו מזמנים קודם את פונקציה ה-API המקורית, הרי שהתוכנה הייתה מסתיימת עוד לפני שהיינו מספקים לבצע את הפעולות

המיוחדות שלנו. במקרים של פונקציות API אחרות, העדפנו לבצע קודם זימון של פונקצית ה-API המקורית, וזאת כדי שמערכת-ההפעלה לא תעוכב עם הפעולות המיוחדות שהוספנו.

עם סיום הפעולות שהוספנו, אנו מחזירים מתוך הפונקציה החדשה את הערך של המשתנה (bResult) שקיבלנו כתוצאה מזימון הפונקציה המקורית

```
BOOL WINAPI CModuleScope::MyExtTextOutW(
    HDC hdc,           // handle to device context
    int X,            // x-coordinate of reference point
    int Y,            // y-coordinate of reference point
    UINT fuOptions,   // text-output options
    CONST RECT *lprc, // optional clipping and/or opaquing rectangle
    LPCWSTR lpString, // points to string
    UINT cbCount,     // number of characters in string
    CONST INT *lpDx   // pointer to array of intercharacter spacing)
{
    char strData[MAX_BUFF];
    BOOL bResult;
    int nMin;
    size_t sz;

    ::ZeroMemory((PBYTE)strData, MAX_BUFF);
    bResult = FALSE;

    try
    {
        bResult = ::ExtTextOutW(
            hdc,
            X,
            Y,
            fuOptions,
            lprc,
            lpString,
            cbCount,
            lpDx);

        nMin = min((MAX_BUFF - 2) / 2, cbCount);
        sz = wcstombs(strData, lpString, nMin);
        if (sz == -1)
            throw (sz);
        strData[nMin] = '\\0';
    }
    catch (...)
    {
        WriteExceptionFunction(__FUNCTION__);
        return bResult;
    }
    PrepareLogMsg("ExtTextOutW", "N/A", strData, bResult);
    return bResult;
}
```

קוד חדש עבור פונקצית API (ExtTextOutW)

2. דריסת הפונקציה ExtTextOutW בפונקציה החדשה MyExtTextOutW

```
sm_pHookMgr->HookImport("gdi32.dll", "ExtTextOutW",  
(PROC)CModuleScope::MyExtTextOutW);
```

בשלב זה, אנו מבקשים ממערכת ההחלפה להחליף את הפונקציה ExtTextOutW, שהגדרת נמצאת בקובץ gdi32.dll, בפונקציה החדשה שהגדרנו MyExtTextOutW.

3. החזרת פונקציות ה-API המקוריות

```
RemoveHook( pHook );
```

כאשר אנו סוגרים את מערכת ה-TMA, אנו מבטלים את פעולת ה-Hooking, על ידי שליחת הפרמטר pHook, שהתקבל בשלב השני.

להלן רשימת פונקציות ה-API עליהן ביצענו פעולת Hooking, כדי לקבל מידע על פעולות המשתמש:

1. ExitProcess – בקשה לסגירת אפליקציה
2. ExtTextOutW – כתיבת טקסט גרפי על המסך
3. ShowWindow – הצגת החלון
4. OpenFile – פתיחת קובץ
5. CreateFileA – יצירה קובץ חדש או פתיחת קובץ קיים, גרסת ASCII
6. CreateFileW – יצירה קובץ חדש או פתיחת קובץ קיים, גרסת WIDE
7. ReadConsoleW – כתיבה למסך DOS
8. ReadFile – פתיחת קובץ לקריאה
9. WriteConsoleW – כתיבה למסך DOS
10. GetMessageW
11. GetMessageA
12. PeekMessageW
13. PeekMessageA
14. SendMessageW
15. SendMessageA
16. PostMessageW
17. PostMessageA
18. GetMenuItemInfoW

פונקציות 10-18 שייכות למשפחה של פונקציות מיוחדות ל Windows המאפשרות שליחת הודעות בין החלונות הפעילים. הודעות אלו הן מסרים לתוכנות אלו פעולות הן צריכות לבצע. פעולות

כאלו יכולות להיות זימון של פונקצית API. כך לדוגמא, כדי לסגור חלון, יש אפשרות שהתוכנה תזמן את הפונקציה `CloswWindow`, או לחילופין שהתוכנה תקבל מסר `WM_CLOSE` שמבקש ממנה לבצע זימון של הפונקציה `CloseWindow`.

החשיבות של שימוש בפונקציות ההודעות לסביבת `User-Modeling` היא בכך שפעולות שזיהינו בהודעות ואנו מסיקים שצריך לחזור על אותן פעולות (כיון שזיהינו שהפעולה מייצגת את המשתמש), ניתן לבצע שליחת הודעה לתוכנה שקיבלה את אותה הודעה בעבר, באופן יזום על ידי תוכנה חיצונית, מבלי שהמשתמש יצטרך לבצע פעולה כלשהי.

לכן, בדומה למצב בו אנו לומדים על פעולות המשתמש ללא שינוי קוד המקור, אנו יכולים בעתיד גם להשפיע על פעולות המשתמש ללא שינוי בקוד מקור, בהתאם למסקנות החדשות שקיבלנו. לדוגמא, ניתן לבצע בדיקה ללא התערבות המשתמש, האם השתנה מידע באתר אינטרנט, ובמידה ואכן יש שינוי לעדכן את המשתמש על השינוי.

פרק 3.1.2 - סיכום יתרונות וחסרונות שימוש בטכניקת

להלן טבלה המסכמת את היתרונות של שיטת ה-Hooking המוצגת בעבודה זו:

יתרון	הסבר
1	אין צורך בגישה לקוד מקור בניגוד לכל השיטות המוצגות במחקרים קודמים, שיטת ה-Hooking לא דורשת גישה לקוד המקור של התוכנות, כיון שהשיטה מבוססת על ניטור הפעולות המתבצעות ברמת ה-API של Windows. יתרון זה משמעותי ביותר, כיון שתוכנות אינן משוחררות בצרוף קוד המקור שלהן
2	אפשרות מעקב אחר כל התוכנות שיטת ה-Hooking מאפשרת ביצוע מעקב אחר כל התוכנות במערכת-ההפעלה Windows, ולא רק מעקב אחר פעולות המתרחשות בתוכנה אחת ספציפית, כמו שמוצג ברוב המחקרים
3	אין צורך בידע מוקדם לגבי זהות התוכנות אותן יפעיל המשתמש כיון ששיטת ה-Hooking יכולה לבצע מעקב אחר כל התוכנות, אין צורך בידע מוקדם לגבי התוכנות מהן אנו צריכים לשאוב מידע לגבי פעולות המשתמש
4	אין תלות בגרסאות חדשות של התוכנות גם אם מוחלפות תוכנות לגרסאות חדשות יותר, שיטת ה-Hooking ממשיכה לפעול כרגיל, כיון שהשיטה אינה תלויה בתוכנה עצמה, אלא במערכת-ההפעלה. מערכת-ההפעלה Windows שומרת תאימות אחורה לרוב פונקציות ה-API, כך שגם עבור גרסאות חדשות של Windows עדיין תפעל מערכת ה-TMA
5	אין תלות בפלטפורמה כלשהי (JAVA או .NET) גם עבור תוכנות שנכתבו לסביבת JVM או .NET, שיטת ה-Hooking עדיין עובדת, כיון שהיא שהסביבות הנ"ל בנויות על שכבת ה-API. כלומר שכבת ה-API עליה מבוססת שיטת ה-Hooking מהווה מכנה משותף לסביבות של JVM או .NET.
6	נתמכת על ידי מערכת-ההפעלה Windows שיטת ה-Hooking נתמכת על ידי חברת Microsoft, כך שניתן לבסס עליה שימוש לטווח ארוך. כלומר השיטה היא סטנדרטית בסביבת מערכת-ההפעלה Windows

טבלת יתרונות של שיטת ה-Hooking

להלן טבלה המסכמת את החסרונות של שיטת ה-Hooking המוצגת בעבודה זו:

הסבר	חסרון	
שיטת ה-Hooking גורמת למעשה לזימון כפול של פונקציות ה-API, בהשוואה למצב נורמאלי (ללא Hooking) של מערכת-ההפעלה. על מנת להתגבר על חסרון זה, מערכת ה-TMA מבצעת פעולת Hooking סלקטיבית. דהיינו, רק מספר פונקציות רלוונטיות למשימת מידול המשתמש, עוברות תהליך Hooking	העמסה של פעולות מערכת-ההפעלה	1
פונקציות ה-API הנאספות בשיטת ה-Hooking הן גולמיות בהשוואה למידע הנאסף בשיטות האחרות, ולכן קיים צורך לבצע עיבוד נוסף של המידע. על מנת להתגבר על חסרון זה, בניסויים הראשונים הגדרנו אירועים המתרחשים במערכת, כמורכבים מאוסף "אטומים" של פקודות API	מידע גולמי	2
לא קיים פתרון תכנותי לחסרון זה, אולם הנחת העבודה של מחקר זה היא שמערכת-ההפעלה Windows היא המערכת הנפוצה ביותר למחשבי Desktop (~90%), לכן Windows מספיק חשובה בשביל לפתח כלי המותאם רק לסביבה זו	נתמכת רק בסביבה של Windows	3

טבלת חסרונות של שיטת ה-Hooking

עבור הניסוי השני (Classification) ועבור הניסוי 3 (Prediction), השארנו את המידע כרצף אטומים של פקודות API, וזאת כיון שהאלגוריתמים בניסויים הנ"ל לא מייחסים חשיבות למשמעות המידע אלא לרצף עצמו.

פרק 3.2 - תוכנת ה-Server (Dataset Builder)

מטרת תוכנת ה-Server היא לרכז את כל הנתונים המגיעים מתוכנות ה-Clients של המשתמשים השונים, ולהמיר אותם לפורמט המתאים עבור אלגוריתמי ה-Classification שבתוכנת [WEKA] ועבור האלגוריתם [IPAM] ל Prediction.

להלן נתאר את החלק הטכני של תוכנת ה-Server, אשר פותחה בעזרת הקומפיילר C#. תוכנת ה-Server פותחת ערוץ תקשורת להאזנה, אשר מאפשר קבלת נתונים אודות פעולותיהם של משתמשי המחשב השונים, שהותקנו במחשבם תוכנת Client. כתוצאה מכך, נוצרה אפשרות לבצע איסוף נתונים מקבוצה של משתמשים העובדים במקביל, תכונה אשר לא קיימת בעבודות קודמות.

במבט על [איור 2](#) המציג צילום מסך של תוכנת ה-Server, ניתן להבחין ביחידות המידע הנשלחות ממחשבי ה-Client. כל פיסת מידע שכזו, מכילה את תשעת השדות הבאים:

1. Computer – שם המחשב ממנו הגיע הנתון
ערך לדוגמא: IBM-BF5EBDA0C5F
2. User – שם המשתמש ממנו הגיע הנתון
ערך לדוגמא: Danny
3. Date – התאריך בו הגיע הנתון
ערך לדוגמא: 06-07-2005
4. Time – השעה בה הגיע הנתון עם מידת דיוק של מילי שניות
ערך לדוגמא: 13:50:48:501
5. Process – שם התוכנה ממנה הגיע הנתון
ערך לדוגמא: iexplore.exe
6. PID (Process ID) – מספר שונה עבור כל אפליקציה שמופעלת, אשר מוקצה על ידי מערכת-ההפעלה
ערך לדוגמא: 542
7. API (Application Programming Interface) – שם הפונקציה שהופעלה בחשבון המשתמש
ערך לדוגמא: CreateFileW
8. Data – הפרמטר הראשון שקיבלה פונקציה ה-API
ערך לדוגמא: SendMessageW
9. Event – הפרמטר השני שקיבלה פונקציה ה-API
ערך לדוגמא: <http://www.google.com>

למרות שפונקציות API לא חייבות לקבל שני פרמטרים, דהיינו, ישנן פונקציות API המקבלות יותר או פחות משני פרמטרים, עדיין הגדרנו את בסיס הנתונים כך שלכל מידע על פונקציה אנו שומרים מקסימום שני פרמטרים (Event ו Data). הסיבה לכך היא שביקשנו לשמור על אחידות מבנה הנתונים, על מנת שיותאמו בקלות יותר לשימוש כקלט עבור האלגוריתמים בהם השתמשנו ב-[ניסויים](#).

תוכנת ה-Server מאפשרת לאסוף נתונים בזמן אמת, דהיינו תוך כדי ריצת תוכנות ה-Client על המחשבים של המשתמשים השונים. כמו כן תוכנת ה-Server מאפשרת טעינה של נתונים היסטריים שנשמרו בקבצים מקומיים.

לאחר שנאספו מספיק נתונים, תוכנת ה-Server מאפשרת את עיבוד הנתונים ויצירת נתונים חדשים על סמך הנתונים הגולמיים שהגיעו ממחשבי ה-Client. לדוגמא, יצירת ערכים עבור התכונה FirstQuarter, המתארת ביחידות של אחוזים, את זמן העבודה הכי גדול של רבעון יומי.

תוכנת ה-Server מכילה גם מודול המאפשר יצירה אוטומטית של קבצי $arff^2$ המיועדים לתוכנת [\[WEKA\]](#). לשם הרצת אלגוריתמים ל Classification.

Computer	User	Date	Time	Process	PID	API	Data	Event
IBM-EPFE B...	Danny	06-07-2005	13:48:57.304	explorer.exe	1538	CreateFileW	C:\Documents and...	ist1.css
IBM-EPFE B...	Danny	06-07-2005	13:50:00.319	explorer.exe	1538	CreateFileW	C:\Documents and...	fbfMain[1].css
IBM-EPFE B...	Danny	06-07-2005	13:50:00.339	explorer.exe	1538	SendMessageW	WM_SETTEXT	Ynet - news and content from Israel (Yedioth Ahronoth web site) - news
IBM-EPFE B...	Danny	06-07-2005	13:50:00.349	explorer.exe	1538	SendMessageW	WM_SETTEXT	http://www.fbi-online.co.il/web/Processing?loginType=U&language=
IBM-EPFE B...	Danny	06-07-2005	13:50:00.409	explorer.exe	1538	SendMessageW	WM_SETTEXT	FbiMainPage - Microsoft Internet Explorer

איור 2 - דוגמא למסך ראשי של תוכנת ה-Server

² פורמט מיוחד לקובצי הקלט המיועדים לתוכנת WEKA

שלושה ניסויים שונים ערכנו כדי לבדוק את תרומת מערכת ה-TMA לביצוע User-Modeling.

בניסוי הראשון ICQ-Explorer Event, הראנו כיצד רצף פקודות API יכול לייצג אירוע מורכב, המתאר אינטראקציה בין זוג משתמשים. חשיבות הניסוי היא בהוכחת הטענה כי בחירת רצף אטומים של פקודות API נותן מידע חדש שלא היה יכול להתקבל אלמלא השימוש במערכת ה-TMA.

הניסוי הראשון מראה כיצד ניתן להתגבר על חיסרון מערכת ה-TMA, שהמידע הראשוני המתקבל הוא גולמי, דהיינו, דרוש עיבוד נוסף על מנת לקבל מידע ברמה המתוארת ב- [פרק 2 עבודות קודמות](#).

בניסוי השני Classification, בחנו את תרומת רצף פקודות ה-API שאספנו ממשתמשים שונים לזיהוי סוג המשתמש (סטודנט או עובד מעבדה), כאשר המשתמשים עבדו על המחשבים ללא הגבלות מצידנו.

חשיבות הניסוי היא להראות כי גם רצף פקודות API פשוט, ללא שכבת עיבוד נוספת כפי שתיארנו בניסוי הראשון, עדיין מספיק טובה על מנת להשתמש ברצף הפקודות כקלט עבור אלגוריתמים ידועים.

בניסוי השלישי Prediction, בחנו את תרומת רצף פקודות ה-API שאספנו מזוגות לביצוע חיזוי של פקודת ה-API הבאה, כאשר המשתמשים הם זוגות שהטלנו עליהם ארבעה משימות משותפות.

הניסוי השלישי מראה כיצד ניתן לבצע User-Modeling גם עבור משתמשים הנמצאים בקבוצה, והעובדים על אפליקציות שונות. כל זאת בניגוד לעבודות הקודמות, בהן בוצעו הפעולות רק על ידי משתמש יחיד ובסביבה של אפליקציה בודדת.

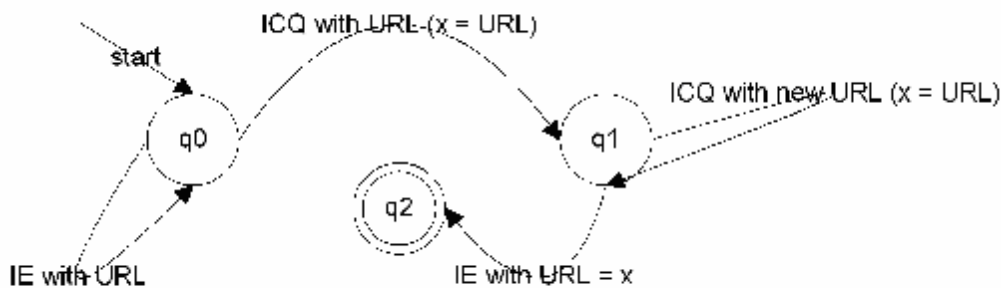
יתרון זה נובע משתי תכונות חשובות של מערכת ה-TMA. התכונה הראשונה היא עובדת היות מערכת ה-TMA מושתתת על טכניקת ה-Hooking, טכניקה אשר מסייעת לביצוע איסוף מידע לגבי פעולות המשתמש המתרחשות בסביבה של מספר אפליקציות במקביל. התכונה השנייה היא עובדת היות מערכת ה-TMA מושתתת על ארכיטקטורת Client-Server, דבר המאפשר מעקב אחר פעילות של מספר משתמשים במקביל.

פרק 4.1 - תיאור ניסוי 1: ICQ-Explorer Event

בהשוואה לשיטות איסוף נתונים המתוארות בעבודות קודמות, שיטת ה-Hooking סובלת מחיסרון של איסוף נתונים גולמיים בלבד, דהיינו, איסוף פקודות API חסרי משמעות. על כן פיתחנו מודול נוסף במערכת ה-TMA שמאפשר זיהוי רצפים מיוחדים של פקודות API, כך שלרצף יש משמעות ("אירוע") בנוגע לפעולות המשתמש.

כפי שתיארנו במאמר [KS], להלן דוגמה לאירוע ICQ-Explorer המורכב מרצף פקודות API. האירוע אותו ניסינו לזהות, הוא מצב בו המשתמש עבר לכתובת URL (Unified Resource Locator) באינטרנט כתוצאה מהודעה שנשלחה אליו ממשתמש אחר על ידי תוכנת המסרים ICQ. דוגמה זו ממחישה את יכולת מערכת ה-TMA לזהות אירועים המתרחשים במערכת של יותר ממשתמש אחד.

בניסוי זה השתמשנו בפונקציית API אחת בשם ExtTextOutW, כדי לזהות את האירוע שהגדרנו. פונקציה זו מאפשרת קבלת מידע לגבי הטקסט שנכתב בחלון התוכנה. כך למשל, הטקסט המוקלד בתוכנת ה-ICQ "נתפס" על ידי הפונקציה ExtTextOutW, כמו גם הלינק שמופיעה בחלון ה-Explorer. מכאן, שפונקציה זו, תורמת מידע רב בבואנו לבצע מידול משתמש, שכן היא מאפשרת לנו לקבל מידע לגבי כל מה שרואה המשתמש על המסך.



איור 3 - אוטומט לתיאור אירוע: ICQ-Explorer

האירוע מתואר בעזרת אוטומט DFA, כאשר המעברים בין מצבי האוטומט מתרחשים בעקבות זיהוי מערכת ה-TMA מצב של פקודת API שהתרחשה (פקודת ExtTextOut במקרה זה). במצב ההתחלתי (q0) האוטומט ממתין שהמשתמש יפעיל את תוכנת Explorer ויגלוש לכתובת אינטרנט כלשהי - URL. כל זמן שהמשתמש לא קיבל הודעת ICQ המכילה כתובת URL,

האוטומט נשאר במצב q0. ברגע שמתקבלת הודעת ICQ עם כתובת URL, האוטומט עובר למצב q1, ושומר בזיכרון (מסומן x) את כתובת ה-URL. במצב q1 האוטומט נשאר כל זמן שמתקבלות עדיין הודעות ICQ עם כתובת URL, כאשר זיכרון האוטומט מעדכן מחדש את הכתובת ה-URL האחרונה שהגיעה בהודעה. במצב q1, ברגע שמוזהה האוטומט מעבר של המשתמש לאותה כתובת URL האחרונה שעודכנה לפי הודעת ה-ICQ, עובר האוטומט למצב סופי q2, דהיינו זיהוי האירוע. תוצאות הניסוי הראשון הן זיהוי מלא של האירוע שביקשנו בכל המקרים שהתרחשו במציאות במערכת.

החיסרון של הניסוי הראשון הוא בכך שהניסוי מתייחס לאירוע שהיה לנו מידע מלא לגביו, עוד בשלב הפיתוח של מערכת ה-TMA. לכן, יכולנו לבצע שינויים בקוד מקור של תוכנת ה-TMA, כאשר שינויים אלו "תפורים" ספציפית עבור הניסוי הנוכחי. לעומת זאת, במידה ונרצה להגדיר אירוע אחר, הרי שנאלץ להוסיף קוד חדש. כלומר עבור כל אירוע חדש שנרצה לבדוק, נצטרך להוסיף קוד ולקמפל את התוכנה מחדש.

ניתן לפתור בעיה זו על ידי יצירת קובץ קלט עבור מערכת ה-TMA, שיכיל מידע לגבי האירועים אותם אנו מצפים שהמערכת תזהה. קובץ הקלט צריך להגדיר אוטומט כפי שתיארנו לעיל, ומערכת ה-TMA צריכה לדעת לקרוא את הקובץ, ולהכניס נקודות בדיקה במצב ריצה שלה המערכת, כפונקציה של המצבים שהוגדרו באוטומט.

במשך שלבי העבודה, ניסינו לבצע קבצי קלט של אוטומט, אולם נתקלנו בקשיים בהגדרת נקודות הבדיקה המתארים את מצבי האוטומט. הבעיה קשה יותר כאשר אנו מנסים להגדיר אירועים המתארים פעולות של משתמשים בקבוצה, אירועים שאף עשויים להתרחש במקביל. על כן, בעבודה זו לא המשכנו ביישום קבצי קלט אוטומט עבור מערכת ה-TMA. עם זאת אנו משוכנעים שהדוגמא שהצגנו בניסוי הראשון, מראה את היתכנותו של עיקרון בניית אירוע בעל משמעות מרצף פקודות API.

פרק 4.2 - תיאור ניסוי 2: Classification

המטרה בניסוי השני היא לבחון את איכות הנתונים ברמת פונקציות API לשם ביצוע קלסיפיקציה של המשתמש. לשם כך השתמשנו בתוכנת [WEKA] המאפשרת ביצוע הרצה של אלגוריתמים לקלסיפיקציה. קובץ הקלט אותו הכנו נמצא ב-[נספח 1](#).

בניסוי השתתפו 32 משתמשים: 8 סטודנטים במעבדה למדעי המחשב ו 24 סטודנטים מתחומים שונים במעבדה לשרות כל הסטודנטים באוניברסיטה. המשתתפים בניסוי לא קיבלו הוראות מיוחדות, אלא המשיכו לעבוד על המחשב שלהם באופן רגיל, אם כי המשתתפים ידעו כי פעולותיהם מוקלטות ברקע. בכל אחד מהמחשבים של המשתמשים הותקנה תוכנת Client, אשר ביצעה את המעקב והרישום לקובץ של פקודות ה-API אשר התרחשו בחשבון המשתמש. הנתונים לא הועברו בתקשורת לתוכנת ה-Server, כיון שהניסוי לא בודק פעולות של המשתמש בקבוצה. הנתונים נשמרו בקבצים יומיים, ובסוף הניסוי איחדנו כל את כל הקבצים היומיים של כל משתמש לקובץ יחיד.

לאחר איסוף ועיבוד הנתונים מכל המשתתפים, הגדרנו 11 "מאפיינים" עבור קובץ ה-WEKA:

1. FirstQuarter – בחלוקה של שעות היום לארבע רבעונים, אחוז הזמן הראשון הגבוה ביותר של רבעון מסוים, בו היה המשתתף בניסוי פעיל
2. SecondQuarter - בחלוקה של שעות היום לארבע רבעונים, אחוז הזמן השני הגבוה ביותר של רבעון מסוים, בו היה המשתתף בניסוי פעיל
3. App1Name – שם האפליקציה הראשונה הפעילה ביותר
4. App2Name – שם האפליקציה השנייה הפעילה ביותר
5. App3Name – שם האפליקציה השלישית הפעילה ביותר
6. Event1 – ערך שדה ה-Event שהתקבל בתפוצה ראשונה הכי גבוהה
7. Event2 - ערך שדה ה-Event שהתקבל בתפוצה שנייה הכי גבוהה
8. Event3 - ערך שדה ה-Event שהתקבל בתפוצה שלישית הכי גבוהה
9. Data1 - ערך שדה ה-Data שהתקבל בתפוצה ראשונה הכי גבוהה
10. Data2 - ערך שדה ה-Data שהתקבל בתפוצה שנייה הכי גבוהה
11. Data3 - ערך שדה ה-Data שהתקבל בתפוצה שלישית הכי גבוהה

מערכת ה-TMA מאפשרת לקבל מידע לגבי משך הזמן בו עבדו המשתמשים על התוכנות השונות. תוך שימוש באפשרות זו, הוספנו את המאפיינים FirstQuarter ו SecondQuarter, המייצגות את זמני העבודה של המשתמש.

הסיבה שיצרנו את המאפיינים FirstQuarter ו SecondQuarter היא ההנחה כי שעות העבודה של סטודנטים המגיעים למעבדה כללית בד"כ לא מתפרשות לכל אורך היום, אלא מקובצות ומרוכזות ברבעון יומי מסוים. בעוד שסטודנטים השייכים למחלקה כלשהי ועובדים באופן קבוע באותו מחשב, אצלם יתפרשו שעות העבודה ליותר מרבעון אחד. לכן הוספת מאפיינים ייחודיים תקל על אלגוריתמי ה-Classification לתת תוצאות טובות יותר.

להלן דוגמא לשורה נתונים:

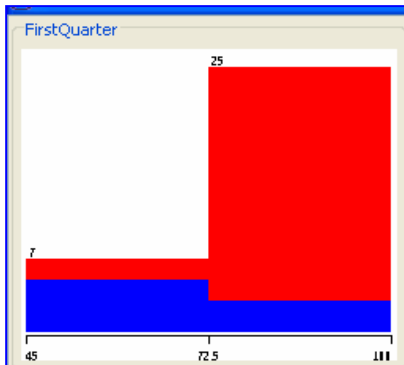
49, 37, notepad, winword, acro32, CreateFileW, SendMessageW, ?, shdocvw.dll, shell32.dll, svcctl

שורה זו מייצגת את המידע הבא:

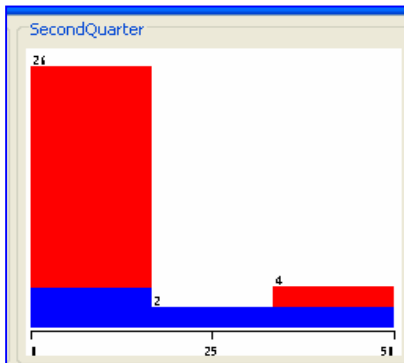
- ברבעון היומי העמוס ביותר, עבד המשתתף בניסוי 49% מהזמן
- ברבעון היומי העמוס השני, עבד המשתתף בניסוי 37% מהזמן
- האפליקציות בהן עבד המשתתף בניסוי הן: notepad, winword, acro32 בהתאמה (משמאל לימין)
- שדות ה-Data שהתקבלו בתפוצה הכי גבוהה הן: CreateFileW, SendMessageW, ? בהתאמה (משמאל לימין)
הסימן "?" מייצג מצב של נתון ריק
- שדות ה-Event שהתקבלו בתפוצה הכי גבוהה הן: shdocvw.dll, shell32.dll, svcctl בהתאמה (משמאל לימין)

להלן ניתוח הנתונים שנאספו עבור הניסוי.

עמודות הצבע האדום (הבהיר יותר) מתארות את הנתונים עבור המופעים של סטודנט, ועמודות הצבע הכחול (הכהה יותר) מתארות את הנתונים עבור המופעים של עובד מעבדה.
ציר ה-X בגרפים מתאר את טווח הערכים של המאפיין.
ציר ה-Y בגרפים מתאר את כמות הנתונים ואת סוגם (לפי הצבע).

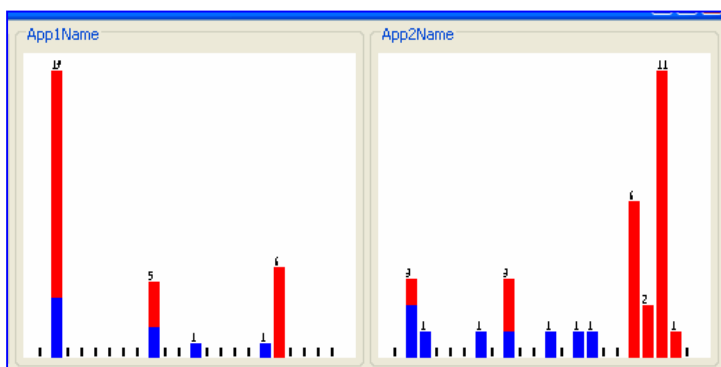


- עבור המאפיין FirstQuarter, ניתן לראות כי 22 דוגמאות של סטודנטים מתוך 24 (91.6%), עבדו ברבעון היומי העמוס ביותר, יותר מ 72.5% מהזמן הכולל.
לעומת זאת, רק 5 דוגמאות של עובדי מעבדה מתוך 7 (71.4%) עבדו ברבעון היומי העמוס ביותר, יותר מ 72.5% מהזמן הכולל.
נתונים אלו מחזקים את ההנחה כי סטודנטים שמגיעים למעבדה באופן אקראי, עבדו רוב הזמן ברבעון יומי יחיד. ומכאן חשיבות הגדרת מאפיין זה לסיווג המשתמשים.



- עבור המאפיין SecondQuarter, ניתן לראות כי 20 דוגמאות של סטודנטים מתוך 24 (83.3%), עבדו ברבעון היומי העמוס השני, בין 0% ל 16% מהזמן הכולל. כלומר רוב הסטודנטים לא עבדו יותר משני רבעונים.
לעומת זאת, רק 4 מתוך 7 (57.1%) מעובדי המעבדה עבדו רק 2 רבעונים, ולפחות עוד (42.9%) יעבדו רבעון נוסף.

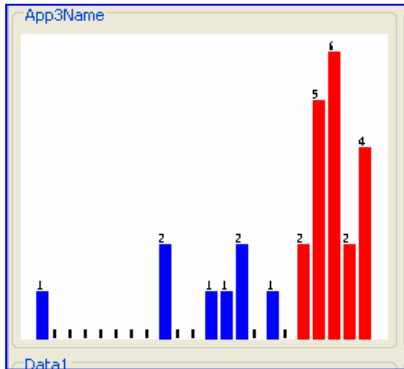
גם נתון זה מחזק את הערכתנו כי עובדי המעבדה פורשים את שעות עבודתם לאורך מספר רבעונים יומיים, בניגוד לסטודנטים.



- נתוני המאפיינים App1Name ו App2Name המתארים את זמן השימוש הראשון, השני והשלישי הארוך ביותר בתוכנה כלשהי,

נותנים הבחנה חדה, כי ישנן תוכנות נפוצות מאוד בין שני סוגי המשתמשים (למשל תוכנת הגלישה באינטרנט Internet Explorer ותוכנת userinit שהינו תוכנה פנימית של מערכת-ההפעלה).

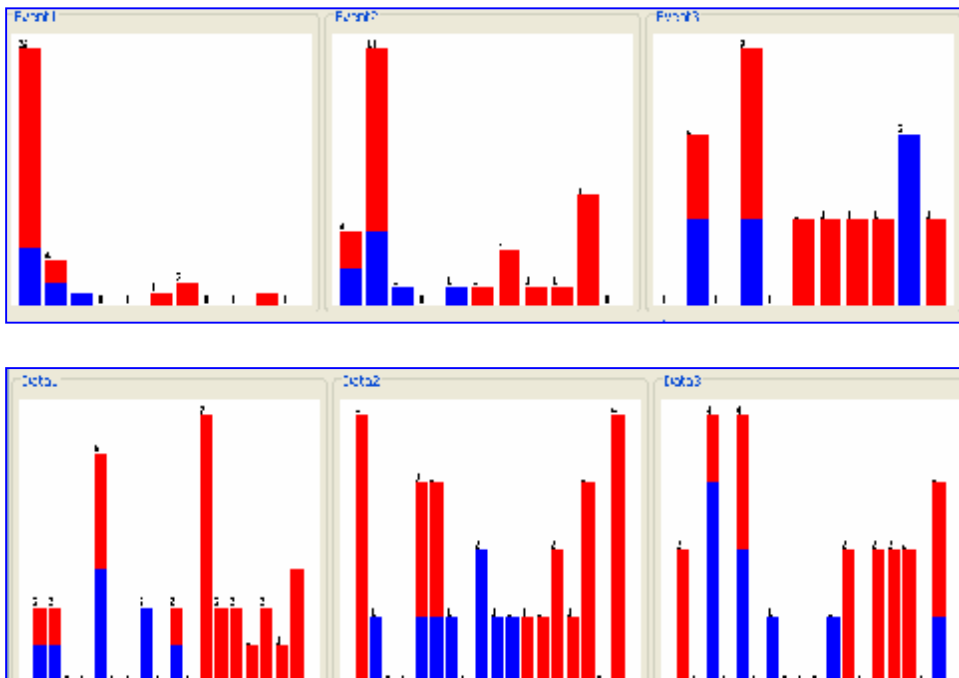
- המאפיין App3Name מלמד אותנו כי ישנן תוכנות הנפוצות רק לקבוצה מסוימת.



לדוגמא: תוכנות כגון WinWord או TaskMgr שהופעלו בתדירות גבוהה יותר רק אצל עובדי המעבדה ניתן להבחין כי ככל שאנו מתקרבים לאפליקציות עם תדירות שימוש נמוכה יותר, הרי שאותן אפליקציות שייכות לסוג משתמש מיוחד יותר, לעומת האפליקציות הכלליות יותר השייכות לשני סוגי המשתמשים.

- המאפיינים Data1, Data2, Data3, Event1, Event2, Event3 קשורים ישירות למאפיינים App1..3, כיון שהם הפרמטרים של מאפיין זה.

כיון שפרמטרים אלו עשויים להיות משותפים למספר תוכנות, למשל פרמטר SendMessageW יכול להגיע כמאפיין Event1 בצימוד לערכים של מאפייני App1Name, App2Name או App3Name, הרי שמידת פיזורם גדול יחסית, ולא ניתן להשתמש בהן להבחנה בין סוגי המשתמשים.



להלן תיאור תוצאות שימוש בנתונים שאספנו עבור אלגוריתמים שונים ל Classification.

אלגוריתם ZeroR

הינו אלגוריתם פשוט, אשר סורק את כל דוגמאות הלמידה שהתקבלו, ובוחר את ה-class המופיע הכי הרבה פעמים כ class עבור הדוגמאות החדשות שהתקבלו. בניסוי שלנו, כיון שהיו 24 דוגמאות של class סטודנט ו 8 דוגמאות של class עובד מעבדה, הרי שאלגוריתם ZeroR בוחר את ה-class סטודנט כחיזוי עבור דוגמאות חדשות. בגלל פשטותו של אלגוריתם ZeroR משתמשים באלגוריתם זה כנקודת השוואה עבור מדד איכות אלגוריתמים אחרים. כלומר, רק אלגוריתמים אחרים אשר תוצאותיהם טובות יותר מאלגוריתם ZeroR יחשבו לאלגוריתמים מתאימים עבור הניסוי. בהמשך נציג שני אלגוריתמים אשר תוצאותיהם טובות מ 75%.

אלגוריתם NNge

(Nearest Neighbor like algorithm using Non-Nested Generalized Exemplars)

אלגוריתם NNge מייצר כללים לחיזוי דוגמאות חדשות, במבנה של if... then... הפרמטרים של חלק התנאי (if) מבוססים על קומבינציות של ערכים עבור המאפיינים שהגדרנו. החלק של המסקנה (then) מבוסס על אחד הסיווגים (class) האפשריים.

לדוגמא:

```
class student IF : 45.0<=FirstQuarter<=100.0 ^ 0.0<=SecondQuarter<=50.0
class lab IF : 49.0<=FirstQuarter<=100.0 ^ 0.0<=SecondQuarter<=49.0
```

על בסיס שני המאפיינים FirstQuarter ו SecondQuarter, אלגוריתם NNge מחליט שאם בדוגמא חדשה שנקבל, המשתמש עבד ברבעון היומי העמוס ביותר יותר מ 45% מהזמן הכולל וגם פחות מ 50% מהזמן העבודה הכולל מהוה הרבעון היומי השני הכי עמוס, אז הדוגמא תסווג כסטודנט.

אם בדוגמא חדשה שנקבל, המשתמש עבד ברבעון היומי העמוס ביותר יותר מ 49% מהזמן הכולל וגם פחות מ 49% מהזמן העבודה הכולל מהוה הרבעון היומי השני הכי עמוס, אז הדוגמא תסווג כעובד מעבדה.

הרצת האלגוריתם עם נתוני הניסויים הניבו 90.625% הצלחה בחיזוי דוגמא חדשה, כאשר דוגמאות חדשות נוצרות על ידי ביצוע Cross Validation של 10% על הנתונים.

אלגוריתם TreesJ48

(Decision Tree)

אלגוריתם TreesJ48 מייצר עץ החלטות לחיזוי דוגמאות חדשות. האלגוריתם בונה עץ החלטות, אשר עליו מייצגים את ה-class-ים האפשריים, עבור סיווג דוגמא חדשה. הצמתים מייצגים את ערכי המאפיינים במסלולים שונים אפשריים.

הרצת הנתונים בניסוי זה יצרה עץ מאוד פשוט:

FirstQuarter <= 98: lab (10.0/3.0)

FirstQuarter > 98: student (22.0/1.0)

מספר העלים בעץ הוא שתיים: lab או student.

האלגוריתם יסווג דוגמאות חדשות כסטודנטים, אם הרבעון היומי הכי עמוס גדול מ-98% מהזמן הכולל. האלגוריתם יסווג דוגמאות חדשות כעובדי מעבדה, אם הרבעון היומי הכי עמוס קטן מ-98% מהזמן הכולל.

תוצאה זו מחזקת את ההנחה כי סטודנטים ממחלקות שונות המזדמנים למעבדה כללית, עובדים מספר שעות מוגבל לחלק מהיום, בעוד ששעות העבודה של עובדי מעבדה נפרשות לאורך היום. הרצת האלגוריתם עם נתוני הניסויים הניב 84.375% הצלחה בחיזוי דוגמא חדשה, כאשר דוגמאות חדשות נוצרות על ידי ביצוע Cross Validation של 10% על הנתונים.

אלגוריתם Classification ל	מספר דוגמאות שסווגו לא נכון	אחוז דוגמאות שסווגו לא נכון	מספר דוגמאות שסווגו לא נכון	אחוז דוגמאות שסווגו לא נכון
ZeroR	24	75%	8	25%
NNge	29	90.625%	3	9.375%
TreesJ48	27	84.375%	5	15.625%

איור 4 - קלסיפיקציה תוך שימוש באלגוריתמים שונים

פרק 4.3 - תיאור ניסוי 3 : Prediction

המטרה בניסוי השלישי היא לבחון את איכות הנתונים שאספנו בעזרת מערכת ה-TMA, לשם ביצוע חיזוי הפקודה הבאה של משתמשים הנמצאים בקבוצה.

ארכיטקטורת ה-Client-Server של מערכת ה-TMA, מאפשרת את ביצוע איסוף הנתונים ממספר משתמשים בקבוצה בזמן אמת. אלגוריתם [IPAM] שימש אותנו כדי לבצע חיזוי על הנתונים שנאספו.

לביצוע הניסוי כינסנו כ-25 זוגות סטודנטים והטלנו עליהם את ארבע משימות. חלק מהנתונים של כ-10 זוגות סטודנטים לא הובאו בחשובים הסופיים, עקב חוסר שלמות בנתונים שנאספו במהלך הניסוי. השתמשנו בנתונים שלמים של 15 זוגות סטודנטים.

שתי משימות חולקו לכל משתתף בניסוי, כאשר שתי המשימות האחרות של משתמש אחד אינן ידועות מראש למשתמש השני, והן מועברות אליו תוך כדי מהלך הניסוי על ידי שימוש בתוכנת מסרים.

להלן תיאור ארבע המשימות שהטלנו על זוגות הסטודנטים:

1. השוואת כמות קבצי ה-dll בספריית c:\windows\system

משתמש ראשון צריך למצוא את כמות קבצי ה-dll בספרייה הנ"ל.

אח"כ לשלוח את מספר הקבצים למשתמש שני תוך שימוש ב-ICQ.

משתמש שני צריך לחשב גם כן את מספר קבצי ה-dll בספרייה הנ"ל במחשב שלו,

ולהכריז למי יש יותר קבצי dll, ע"כתיבה לקובץ חדש בשם moredll.txt, בקובץ שישמר

בספרייה c:\experiment number\moredll.txt. משתמש ראשון צריך לבצע הכרזה דומה

במחשב שלו.

2. יצירת גרף טמפרטורה

משתמש ראשון צריך ליצור קובץ נתונים של הטמפרטורה בארבעת הימים הקרובים,

באזור גוש דן.

אח"כ לשלוח את הקובץ למשתמש שני תוך שימוש ב-ICQ.

משתמש שני צריך להשתמש ב-excel ליצור גרף עמודות של נתוני הטמפרטורה בארבעת

הימים הקרובים, ולשמור את הקובץ בספרייה c:\experiment number\temp.xls

3. פיתוח תוכנית לחישוב שורשים ריבועיים

משתמש ראשון צריך לפתח את פונקציה השורש בקובץ נפרד.

אח"כ לשלוח את הקובץ למשתמש שני תוך שימוש ב-ICQ.

משתמש שני צריך להשתמש בפונקציה לחישוב השורש הריבועי של כל המספרים מ 1 עד 100, ולשמור את התוכנית בספרייה c:\experiment number

4. הכנת רשימה של שמות המדינות בארה"ב ובאירופה
 משתמש ראשון מחפש את רשימת המדינות בארה"ב. שולח את הרשימה למשתמש שני
 תוך שימוש ב-ICQ. משתמש שני מחפש את רשימת המדינות באירופה. מחכה לקבלת
 הרשימה של משתמש ראשון, ואז מבצע שמירה של הנתונים לקובץ
 c:\experiment number\states.txt

טבלת הנתונים עבור הניסוי מרוכזת ב [נספח 2](#).

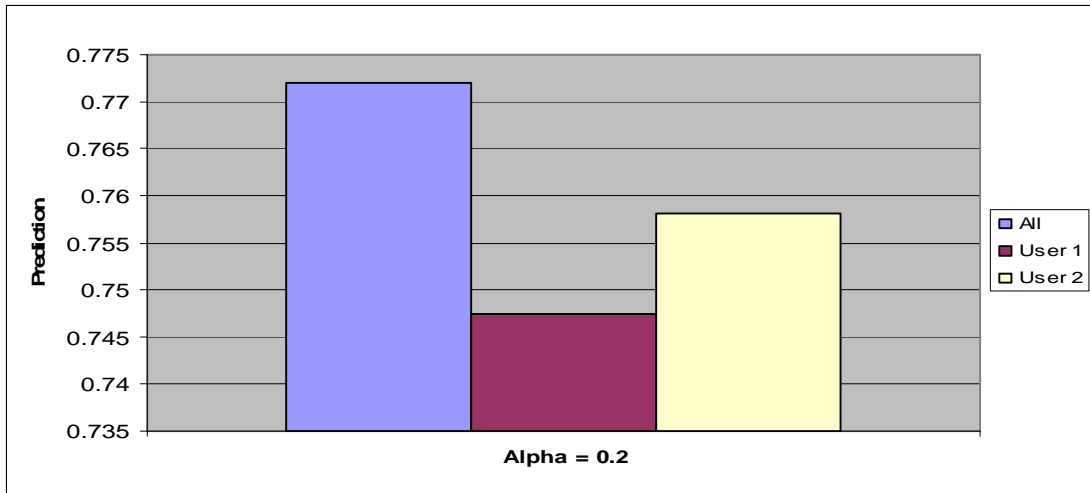
להלן תוצאות הרצת אלגוריתם IPAM על הנתונים שנאספו בניסוי השלישי.

- הנתונים לאלגוריתם נבנו מתוך השדות של שם המחשב, שם האפליקציה ופקודת ה-API מהם הגיעו הנתונים.
 להלן דוגמא לרצף נתונים:

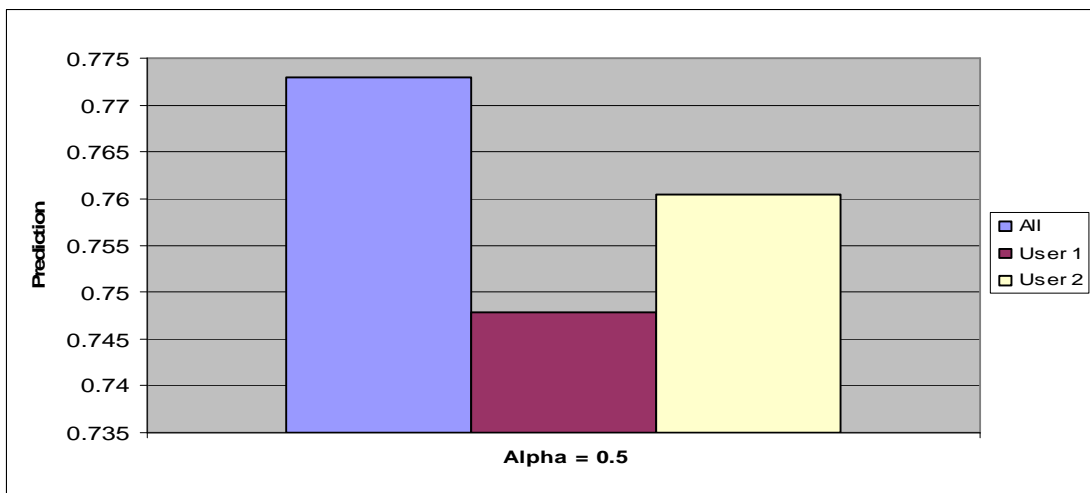
User	Application	API
bisli	EXCEL.EXE	SendMessageW
bisli	ICQLite.exe	ExtTextOutW
capuchino	javaw.exe	ExtTextOutW
capuchino	ICQLite.exe	ExtTextOutW
bisli	javaw.exe	ExtTextOutW

- ציר ה-Y מייצג את מידת הצלחת החיזוי.
- העמודה השמאלית (תכלת) מייצגת את תוצאות החיזוי עבור הנתונים שמערכת ה-TMA אספה משני המשתמשים בזמן אמת.
 מערכת ה-TMA מאפשרת לשמור על הנתונים של כל משתמש בנפרד, כמו גם לשמור על הנתונים של המשתמשים העובדים במקביל. כלומר כל הנתונים של המשתמשים זורמים בזמן אמת למחשב מרכזי (Server), כך שנוצרת לנו קבוצה אחת של נתונים עבור מספר משתמשים.
- העמודה האמצעית (בורדר) מייצגת את תוצאות החיזוי עבור הנתונים שמערכת ה-TMA אספה ממשתמש ראשון
- העמודה הימנית (צהוב) מייצגת את תוצאות החיזוי עבור הנתונים שמערכת ה-TMA אספה ממשתמש שני
- כל גרף מייצג שימוש ב-Alpha שונה עבור אלגוריתם IPAM.
 עבור האלגוריתם IPAM, קיים קבוע Alpha אשר ערכו נע בטווח $0 \leq \text{Alpha} \leq 1$.

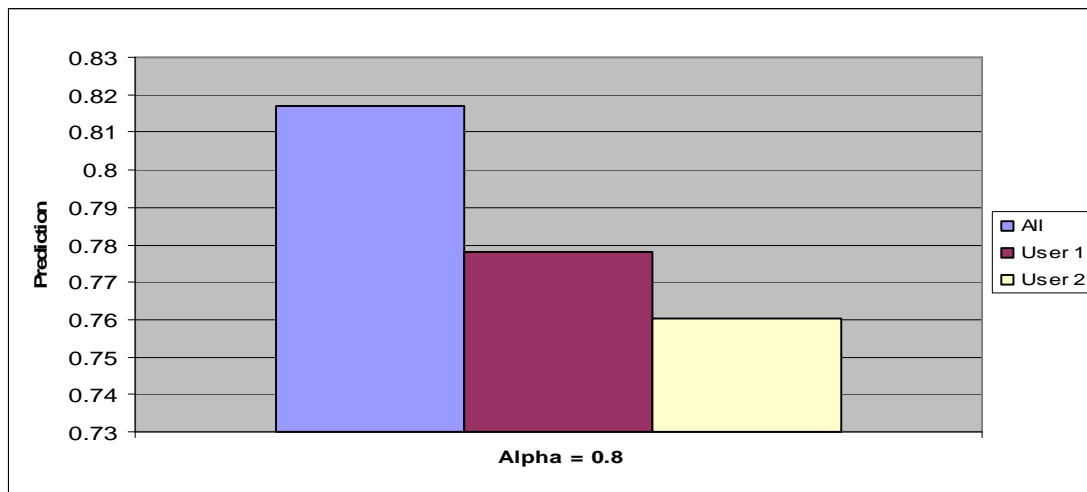
ככל שהקבוע Alpha קטן יותר, ישנו משקל קטן יותר לערכים ישנים שנאספו. לכן חיזוי פעולה הבאה במקרה של $\text{Alpha} = 0$, יהיה מבוסס על המקרה האחרון שקיבלנו. ככל שהקבוע Alpha גדול יותר, ישנו משקל גדול יותר לערכים ישנים שנאספו. לכן עבור $\text{Alpha} = 1$, אנו שומרים על הסתברות שווה עבור כל אפשרויות החיזוי שכבר התרחשו.



איור 5 - חיזוי תוך שימוש באלגוריתם IPAM, כאשר $\text{Alpha} = 0.2$



איור 6 - חיזוי תוך שימוש באלגוריתם IPAM, כאשר $\text{Alpha} = 0.5$

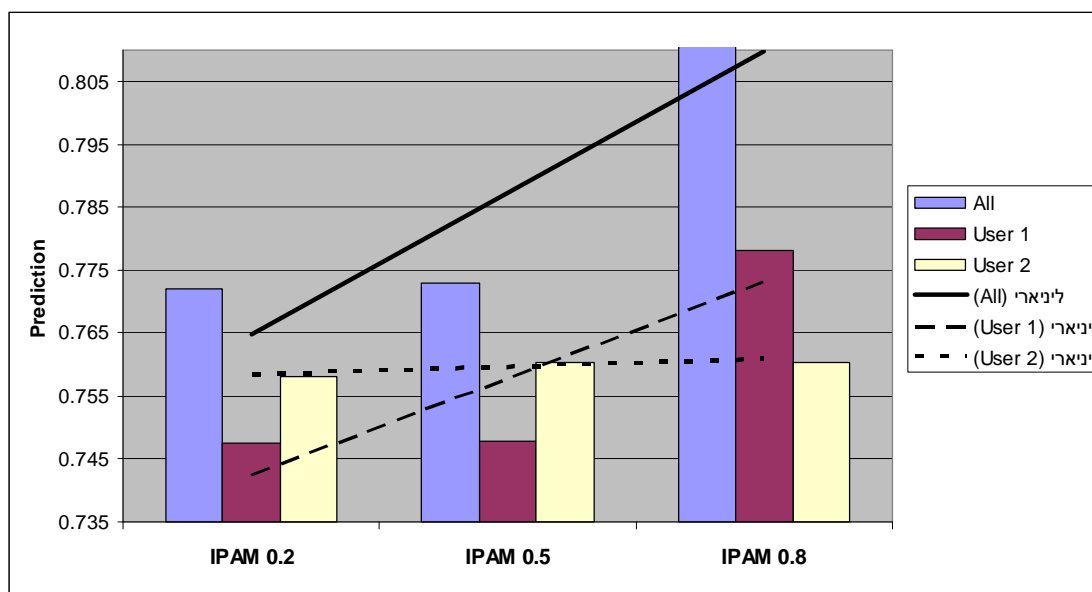


איור 7 - חיזוי תוך שימוש באלגוריתם IPAM, כאשר $\alpha = 0.8$

עבור שלושת ה- α שהצבנו כקבועים באלגוריתם IPAM, קיבלנו תמיד תוצאת החיזוי טובה יותר עבור הנתונים שנאספו בנוגע לכל המשתמשים, לעומת הנתונים שנאספו ממשתמש בודד. הסיבה לכך היא שכמות המידע עבור רצף הנתונים של כל המשתמשים ("All") גדולה יותר מכמות המידע עבור משתמש בודד, על כן קל יותר לאלגוריתם IPAM לבצע חיזוי כשהקלט ארוך יותר.

	Alpha = 0.2	Alpha = 0.2	Alpha = 0.2
All	0.772	0.772933	0.816933
User 1	0.747467	0.7478	0.778133
User 2	0.758067	0.7604	0.7604

טבלת סיכום תוצאות ניסוי IPAM עבור שלושת ה- α



איור 8 - מגמות שינוי כפונקציה של Alpha

קו מגמה עולה חזק, קיבלנו עבור הנתונים של משתמש אחד.
 קו מגמה עולה חלש, קיבלנו עבור הנתונים של משתמש שני.
 קו מגמה עולה חזק מאוד, קיבלנו עבור הנתונים של קבוצת "All", נתונים שנאספו משני המשתמשים במקביל.

ניתוח המגמה של תוצאות הניסוי כפונקציה של Alpha, מלמד אותנו כי ככל ש-Alpha גדולה יותר, כך קיימת עליה באיכות התוצאה.
 תוצאה זו מחזקת את הערכתנו כי עבור פעולות המשתמש המיוצגות על ידי רצפים של פקודות API, קיימת חשיבות לנתונים ההיסטוריים, כיון שבאלגוריתם של IPAM, Alpha גדולה מייחסת חשיבות גדולה ליותר נתונים היסטוריים.

מכאן שהפקטורים של כמות הנתונים וגודל ה-Alpha, ככל שהם גבוהים יותר, נקבל תוצאות חיזוי טובות יותר באלגוריתם של IPAM, עבור הנתונים של פעולות המשתמש, כפי שהם נאספים על ידי מערכת ה-TMA.

בעבודה הצגנו מערכת תוכנה חדשה TMA, המאפשרת איסוף מידע אוטומטי לגבי פעולות המשתמש.

למערכת ה-TMA קיימות מספר יתרונות בולטים בהשוואה לשיטות המוצגות בעבודות קודמות. למשל, מערכת ה-TMA אינה דורשת גישה לקוד מקור של התוכנות, ואינה תלויה בשינוי גרסאות. עם זאת, מערכת ה-TMA מייצרת נתונים גולמיים, אשר דורשים עיבוד נוסף כדי להגיע לרמת מידע שהיה מתקבל, לו היינו אוספים מידע על ידי שינוי קוד המקור של התוכנות.

ניתן להתייחס למערכות שלמות המבצעות מידול משתמש, כבנויות משלושה חלקים:

1. החלק האחראי על איסוף נתונים על פעולות המשתמש.
עבודת מחקר זו הראתה שהמידע הנאסף ע"י מערכת ה-TMA תורם לאיסוף מידע הדרוש, ללא צורך בהתערבות של המשתמש.
2. החלק האחראי על ניתוח הנתונים שנאספו מהחלק הראשון לשם יצירת מידע חדש.
הראנו שימוש באלגוריתמים לחיזוי וקלסיפיקציה אשר מייצרים תובנה חדשה המייצגת את מאפייני המשתמש.
3. החלק האחראי על הטמעת המידע החדש שנוצר בחלק השני לטובת המשתמש.
לא הראנו בעבודה זו כיצד המידע החדש שנוצר מוטמע במערכות מחשב אמיתיות.

החלק השלישי, אותו כאמור לא ביצענו בעבודה זו, יכול להוות כר נרחב לעבודות הבאות. נניח ולמדנו כי משתמש מסוים ברוב הזמן מפעיל תוכנות קומפילציה ובמספר מרים גדול המשתמש מחפש ב-Google מידע עבור שפות תוכנה, נוכל להסיק כי מידע בנושא תכנות עשוי לעניין את המשתמש, ולכן לשלוח אליו מידע אודות גרסאות חדשות של כלי פיתוח וכדומה.

נספח 1 – קובץ קלט ל WEKA

קובץ הקלט עבור WEKA מחולק לשני חלקים.

- החלק הראשון מייצג את מרחב הערכים האפשריים (בדידים או רציפים) עבור המאפיינים שבחרנו.
- החלק השני מייצג את ערכי המופעים האמיתיים שאספנו עבור שני סוגי המשתתפים בניסוי: סטודנטים השייכים למעבדה ועובדים לאורך היום על המחשב, וסטודנטים המגיעים למעבדה ממחלקות שונות, ועובדים באופן אקראי על המחשב.

```
@relation Users_Classification
```

חלק ראשון

```
@attribute FirstQuarter numeric
@attribute SecondQuarter numeric
@attribute App1Name
{acrord32,explorer,excel,calc,cmd,ICQLite,javaw,defrag,iexplore,imapi,lsass,notepad,ta
skmgr,winword,wmpplayer,spoolsv,soffice,internat,mobsync,userinit,igfxtay,vptry}
@attribute App2Name
{acrord32,explorer,excel,calc,cmd,ICQLite,javaw,defrag,iexplore,imapi,lsass,notepad,ta
skmgr,winword,wmpplayer,spoolsv,soffice,internat,mobsync,userinit,igfxtay,vptry}
@attribute App3Name
{acrord32,explorer,excel,calc,cmd,ICQLite,javaw,defrag,iexplore,imapi,lsass,notepad,ta
skmgr,winword,wmpplayer,spoolsv,soffice,internat,mobsync,userinit,igfxtay,vptry}
@attribute Event1
{CreateFileW,SendMessageW,ShowWindow,PeekMessageW,ExtTextOutW,'C:\\WINNT\\Web\\',WM_SE
TTEXT,'C:\\','C:\\WINDOWS\\system32\\',ExitProcess,WM_KEYDOWN}
@attribute Event2
{CreateFileW,SendMessageW,ShowWindow,PeekMessageW,ExtTextOutW,'C:\\WINNT\\Web\\',WM_SE
TTEXT,'C:\\','C:\\WINDOWS\\system32\\',ExitProcess,WM_KEYDOWN}
@attribute Event3
{CreateFileW,SendMessageW,ShowWindow,PeekMessageW,ExtTextOutW,'C:\\WINNT\\Web\\',WM_SE
TTEXT,'C:\\','C:\\WINDOWS\\system32\\',ExitProcess,WM_KEYDOWN}
@attribute Data1
{shdocvw.dll,shell32.dll,svcc1,'C:\\',lsarpc,STDOLE2.TLB,NETLOGON,ExtTextOutW,Show,WM
_SETTEXT,WM_KEYDOWN,folder.htt,http://www.cs.biu.ac.il/LayerStorage.dat,srvsvc,wkssvc
,http://www.google.co.il,Exit}
@attribute Data2
{shdocvw.dll,shell32.dll,svcc1,'C:\\',lsarpc,STDOLE2.TLB,NETLOGON,ExtTextOutW,Show,WM
_SETTEXT,WM_KEYDOWN,folder.htt,http://www.cs.biu.ac.il/LayerStorage.dat,srvsvc,wkssvc
,http://www.google.co.il,Exit}
@attribute Data3
{shdocvw.dll,shell32.dll,svcc1,'C:\\',lsarpc,STDOLE2.TLB,NETLOGON,ExtTextOutW,Show,WM
_SETTEXT,WM_KEYDOWN,folder.htt,http://www.cs.biu.ac.il/LayerStorage.dat,srvsvc,wkssvc
,http://www.google.co.il,Exit}
@attribute class {lab,student}
```

@data

49,37,notepad,winword,acrord32,CreateFileW,SendMessageW,?,shdocvw.dll,shell32.dll,svcc
tl,**lab**

100,0,explorer,iexplore,notepad,CreateFileW,SendMessageW,?,lsarpc,STDOLE2.TLB,NETLOGON
,**lab**

49,11,iexplore,excel,winword,SendMessageW,ShowWindow,PeekMessageW,ExtTextOutW,Show,lsa
rpc,**lab**

98,2,soffice,explorer,iexplore,SendMessageW,CreateFileW,ExitProcess,WM_SETTEXT,WM_KEYD
OWN,Exit,**lab**

59,31,iexplore,explorer,taskmgr,CreateFileW,SendMessageW,ExitProcess,lsarpc,NETLOGON,s
vcctl,**lab**

85,13,explorer,wmpplayer,iexplore,ShowWindow,CreateFileW,?,ExtTextOutW,Show,lsarpc,**lab**

51,49,explorer,javaw,spoolsv,CreateFileW,ExtTextOutW,SendMessageW,lsarpc,WM_SETTEXT,WM
_KEYDOWN,**lab**

49,33,explorer,notepad,winword,CreateFileW,SendMessageW,?,shell32.dll,lsarpc,svcctl,**la
b**

100,0,explorer,userinit,vptray,CreateFileW,ExitProcess,?,Exit,?,?,**student**

100,0,internat,userinit,igfxtray,WM_SETTEXT,'C:\\WINNT\\Web\\',?,folder.htt,?,?,**studen
t**

100,0,internat,userinit,vptray,CreateFileW,ExitProcess,?,Exit,?,?,**student**

100,0,explorer,internat,userinit,CreateFileW,SendMessageW,'C:\\WINNT\\Web\\',folder.ht
t,wkssvc,lsarpc,**student**

100,0,internat,userinit,mobsync,SendMessageW,CreateFileW,WM_SETTEXT,folder.htt,Exit,?,
student

100,0,internat,userinit,mobsync,CreateFileW,ExitProcess,?,WM_SETTEXT,folder.htt,Exit,**s
tudent**

100,0,internat,mobsync,userinit,CreateFileW,ExitProcess,?,folder.htt,Exit,?,**student**

100,0,explorer,userinit,?,ExitProcess,CreateFileW,SendMessageW,folder.htt,Exit,?,**stude
nt**

100,0,explorer,internat,userinit,'C:\\WINNT\\Web\\',WM_SETTEXT,?,folder.htt,?,?,**studen
t**

100,0,iexplore,userinit,mobsync,WM_SETTEXT,'C:\\',WM_KEYDOWN,<http://www.cs.biu.ac.il/>,
Exit,?,**student**

100,0,iexplore,internat,userinit,CreateFileW,WM_SETTEXT,'C:\\',http://www.cs.biu.ac.il
/ ,LayerStorage.dat,shdocvw.dll,**student**

50,50,explorer,userinit,?,CreateFileW,'C:\\WINDOWS\\system32\\',?,shell32.dll,lsarpc,E
xit,**student**

100,0,internat,userinit,igfxtray,SendMessageW,WM_SETTEXT,?,Exit,[http://www.cs.biu.ac.i
l/](http://www.cs.biu.ac.il/),?,**student**

100,0,explorer,userinit,?,CreateFileW,ExitProcess,?,shdocvw.dll,STDOLE2.TLB,LayerStora
ge.dat,**student**

100,0,explorer,?,?,CreateFileW,ExitProcess,?,lsarpc,srvsvc,LayerStorage.dat,**student**

95,5,explorer,internat,mobsync,CreateFileW,SendMessageW,?,LayerStorage.dat,STDOLE2.TLB
,wkssvc,**student**

100,0,explorer,iexplore,userinit,CreateFileW,SendMessageW,?,http://www.google.co.il,wk
ssvc,srvsvc,**student**

100,0,explorer,userinit,internat,CreateFileW,SendMessageW,PeekMessageW,LayerStorage.da
t,lsarpc,svcctl,**student**

100,0,explorer,igfxtray,userinit,CreateFileW,SendMessageW,?,lsarpc,wkssvc,shdocvw.dll,
student
45,35,explorer,internat,?,CreateFileW,SendMessageW,'C:\\WINDOWS\\system32\\',wkssvc,La
yerStorage.dat,lsarpc,**student**
100,0,explorer,iexplore,vptray,CreateFileW,SendMessageW,?,srvsvc,shdocvw.dll,wkssvc,**st**
udent
100,0,explorer,internat,vptray,CreateFileW,SendMessageW,?,folder.htt,shdocvw.dll,srvsv
c,**student**
100,0,explorer,mobsync,internat,CreateFileW,SendMessageW,?,wkssvc,shdocvw.dll,folder.h
tt,**student**
100,0,iexplore,explorer,mobsync,CreateFileW,SendMessageW,PeekMessageW,lsarpc,shdocvw.d
ll,folder.htt,**student**

נספח 2 – נתונים ותוצאות עבור אלגוריתם IPAM

- עמודה Experiment מייצגת את מספר הניסוי, כאשר כל ניסוי מנותח ברמה של מבט על כל הנתונים של כל זוג המשתתפים בניסוי ורמת כל אחד מהמשתתפים בנפרד.
- עמודה # of prediction test מייצגת את מספר המופעים של נתוני API
- עמודות IPAM 0.2, IPAM 0.5 and IPAM 0.8 מייצגות את תוצאות הניסויים עבור שימוש באלגוריתם IPAM עם ערכי Alpha שונים
- עמודה Task [min] מייצגת את משך הזמן עד לסיום המשימות עבור כל משתתף בניסוי, ביחידות של דקות (לא התייחסנו לנתון זה)

Experiment		# of prediction test	Alpha 0.2	Alpha 0.5	Alpha 0.8	Task [min]
1	All	508	0.742	0.74	0.754	
1	User 1	712	0.858	0.861	0.872	13
1	User 2	119	0.538	0.538	0.496	28
2	All	961	0.766	0.765	0.796	
2	User 1	424	0.767	0.764	0.757	38
2	User 2	517	0.768	0.768	0.809	63
3	All	1936	0.836	0.838	0.861	
3	User 1	1213	0.943	0.946	0.946	71
3	User 2	648	0.636	0.639	0.699	63
4	All	842	0.536	0.534	0.643	
4	User 1	421	0.591	0.591	0.677	73
4	User 2	420	0.619	0.619	0.726	97
5	All	2119	0.869	0.874	0.904	
5	User 1	895	0.853	0.857	0.894	51
5	User 2	1147	0.875	0.881	0.907	57
6	All	2031	0.789	0.791	0.827	
6	User 1	307	0.866	0.863	0.837	63
6	User 2	1594	0.886	0.891	0.9	88
7	All	1483	0.61	0.61	0.712	
7	User 1	853	0.621	0.623	0.709	26
7	User 2	592	0.789	0.792	0.836	63
8	All	816	0.773	0.776	0.821	
8	User 1	812	0.9	0.903	0.926	269
8	User 2	91	0.385	0.396	0.44	54
9	All	946	0.572	0.571	0.667	
9	User 1	729	0.632	0.631	0.722	11
9	User 2	404	0.639	0.636	0.708	26
10	All	512	0.754	0.758	0.801	
10	User 1	438	0.678	0.68	0.731	39
10	User 2	439	0.724	0.727	0.761	38
11	All	5534	0.9	0.9	0.913	
11	User 1	368	0.815	0.815	0.84	
11	User 2	6269	0.906	0.906	0.918	
12	All	6296	0.998	0.998	0.999	
12	User 1	1848	0.748	0.75	0.798	30
12	User 2	924	0.876	0.878	0.891	90
13	All	1848	0.748	0.75	0.798	

13	User 1	1854	0.908	0.91	0.934	36
13	User 2	924	0.876	0.878	0.891	35
14	All	928	0.775	0.775	0.828	
14	User 1	123	0.439	0.439	0.48	49
14	User 2	1218	0.892	0.892	0.916	44
15	All	3722	0.912	0.914	0.93	
15	User 1	113	0.593	0.584	0.549	
15	User 2	5191	0.962	0.965	0.973	

- [HL1] H. Lieberman, *Attaching Interface Agent Software to Applications*, AAAI Press (March 1994).
- [HL2] Henry Lieberman, *Integrating user interface agents with conventional applications*. In *Intelligent User Interfaces*, pages 39--46, 1998.
- [IBHYS] Jean-David Ruvini and Christophe Fagot. *IBHYS: a new approach to learn users habits*. In Proceedings of ICTAI'98, pages 200--207. IEEE Computer Society Press, 1998.
- [IPAM] Brian D. Davison and Haym Hirsh, *Predicting sequences of user actions*. In *Predicting the Future: AI Approaches to Time-Series Problems*, pages 5--12, Madison, WI, July 1998. AAAI Press. Proceedings of AAAI-98/ICML-98 Workshop, published as Technical Report WS-98-07.
- [IV] Ivo Ivanov, API hooking revealed, <http://www.codeproject.com/system/hooks.asp>, 2002.
- [LMM] Lashkari, M. Metral, and P. Maes, *Collaborative Interface Agents*. In Proceedings of the Twelfth National Conference on Artificial Intelligence, volume 1. AAAI Press, Seattle, WA, 1994.
- [MSDN] <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/windowing/hooks.asp>.
- [MTR] MITRE Wosit: Widget observation simulation inspection tool, 2001.
- [ONISI] P. Gorniak and D. Poole, [*"Predicting Future User Actions by Observing Unmodified Applications"*](#), Proceedings of the National Conference on Artificial Intelligence, 2000
- [SW] C. Stan II and D. L. Waltz. *Toward memory-based reasoning*. Communications of the ACM, 29:1213-1228, December 1986
- [WEKA] <http://www.cs.waikato.ac.nz/ml/weka/>
- [KS] Kaminka, G. A. and Shimoni, D. 2003. [*Infrastructure for Tracking Users in Open Collaborative Applications: A Preliminary Report*](#). In *Proceedings of the Workshop on User and Group Models for Web-Based Adaptive Collaborative Environments*, part of the User Modeling conference (UM-2003).

Abstract

User-Modeling is a research field in artificial intelligence, which deals with the acquisition and use of a user model, by gathering data about her actions. Information about a user's profile and her repeated actions can be used for filtering the information the user gets and to predict the user's next action. Previous work has a number of assumptions for the data-gathering process, assumptions which are not practical. For example, an assumption that the source code of the applications is available for changing, while most applications do not come with their source code. In addition, previous work that deals with user modeling typically addressed only the case of a single user.

In this work we present a new system, TMA (Tracking Multiple Applications and Multiple Users). The TMA automatically gathers data about the user's actions, without need for making changes in the source code of the applications. The data is composed from information about API: Application Programming Interface functions. The TMA system can monitor every application which works in Windows environment. The TMA system was built for gathering data in real-time from numbers of users, thus it contributes for modeling group of users.

In order to examine the quality of the data which was gathered by the TMA system we made three experiments. In the first experiment we examined if sequences of API functions can be used to build scripts which has meaning for the user's actions. In the second experiment we examined if sequences of API functions can be used as input for classification algorithms. In the third experiment we examined if the TMA system can track couples of users, while the data of this experiment was used as input for prediction algorithm.

Bar-Ilan University

**Gathering Data for User-Modeling
in Environment of
Multiple Users and Multiple Applications**

Danny Shimony

Advisor: Dr. Gal Kaminka

This work is submitted in partial fulfillment of the requirements for Master's degree
in the Department of Computer, Bar-Ilan University

Ramat-Gan

2005