**ZeroHex**

אבשלום תם, אורי פוגלר ושלמה רבינוביץ׳

להלן חלק מהשלבים בעבודה על הפרוייקט ואתגרים שחשבנו שכדאי לציין.

סקירת ספרות, קוד וקהילה

ראשית, סקירת ספרות.

למדנו על וקראנו את העבודה של google deep mind על alphaGo, alphaZeroGo וalphaZero, כמו גם שחזור (בדומיין אחר) מהאתר Medium.

השלמנו רקע תאורטי בלמידה, עץ חיפוש מונטה קרלו, resNet ועוד (חומר מהקורס המתקדם בבינה מלאכותית).

למדנו לעומק את המאמר ״Playing Chess at a Human Desired Level and Style״ שעליו התבססה העבודה שלנו עבור משחק הHex.

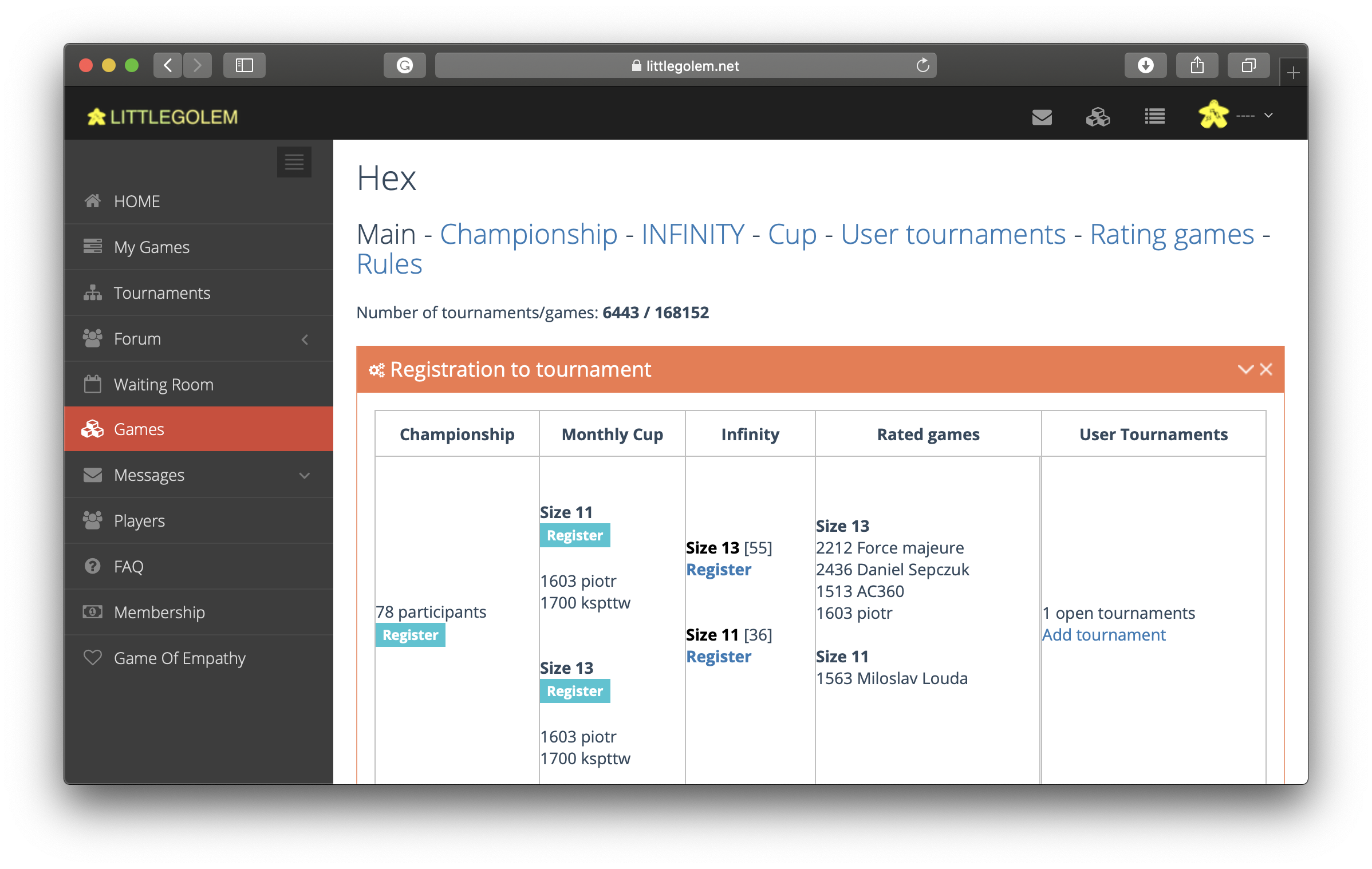
מצאנו סוכן hex מאומן באופן דומה לשל גוגל (אך חלש ממנו, כמובן. משאבי החישוב קריטיים) וכן מצאנו בסיס קוד java עם ממשק גרפי, תיקנו אותו וחיברנו לסוכן.

בשלב זה יצרנו אינטראקציה ראשונה עם המודל, שיחקנו מולו ומול גרסאות משונמכות שרירותיות שלו.

ניגשנו לחפש מקורות, דאטה ומומחים, עבור המשחק hex. מסתבר שהדבר מסובך וקשה בהרבה מאשר עבור משחק קונבנציונלי כגון שחמט, או אפילו גו (שקיבל תהודה לאור האתגר ליצור סוכן שמשחק את המשחק היטב).

ניסינו ליצור קשר עם אנשים שאספו נתונים על משחקי hex או שהשתתפו בטורנירי hex באינטרנט, להצטרף לקבוצת פייסבוק של חובבי hex וליצור התעניינות סביב הפרוייקט שלנו. ההיענות הייתה מוגבלת, בלשון המעטה.

האתר littlegolem שניסינו ליצור קשר עם חברי הקהילה שבו ולקחנו ממנו את המשחקים:



עבודה עם הדאטה

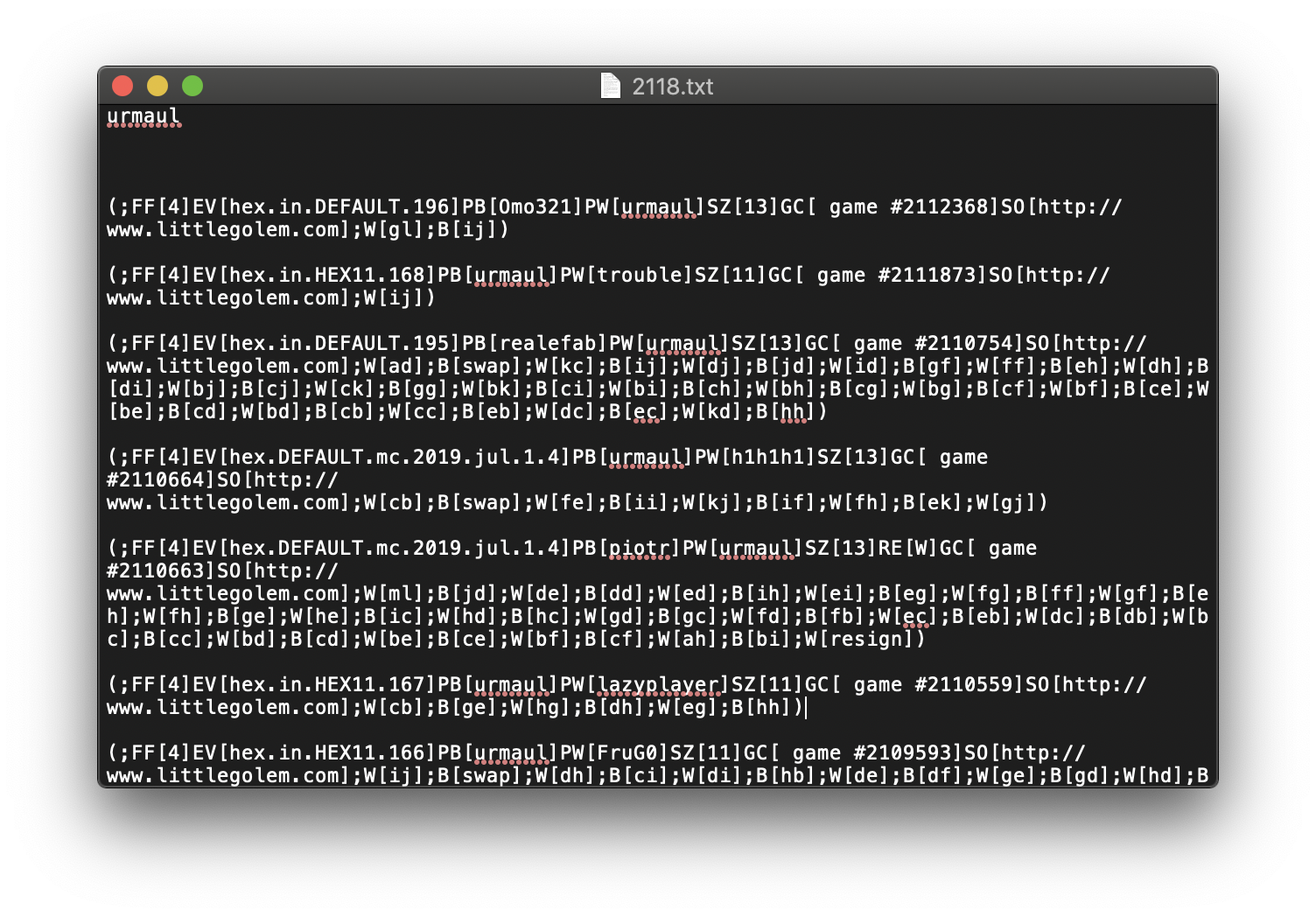
לבסוף את הדאטה הורדנו מאתר המרכז משחקים רבים באינטרנט.

עבדנו על נקיון הדאטה, שחלקו, בעיקר עבור המשחקים שכללו שחקנים בדירוג נמוך (שנוטים לעשות קצת מהלכים ולפרוש), פחות איכותי, שוב, בלשון המעטה.

בנוסף, מכיון שהסוכן בו השתמשנו אומן על גודל לוח 11x11, ומסתבר שגודל הלוח 13x13 יותר פופולרי, לפחות באתר ממנו לקחנו את הנתונים, היינו צריכים גם לסנן ולקחת רק את הנתונים הרלוונטיים לנו.

נצרכנו גם לסנן תווים או מילים ברשומות המשחקים שלא הכילו מידע רלוונטי עבורנו.

לשם נקיון הדאטה כתבתנו כמה סקריפטים וקטעי קוד.

כך נראו רשומות המשחק המקוריות:

לכל שחקן שמרנו קובץ עם כל רשומות המשחקים שלו, כאשר בשורה הראשונה מופיע שמו ובשם הקובץ מופיע הדירוג שלו.

ברשומות המשחק מופיעים פרטים שונים, וביניהם גודל הלוח (sz 13 משמע גודל 13) ושמות המשתתפים, על פיהם חילצנו באופן אוטומטי את שם השחקן שלו מתאים הקובץ ופילטרנו את המשחקים בגודל הלוח הרלוונטי.

לאחר נקיון הדאטה, שמורכב מרשומות של מהלכי המשחק בפורמט מסויים, היינו צריכים להמיר אותו לפורמט ברור וקריא שקל לטעון ולעבוד איתו בקוד.

למשל רצינו שהנתונים לגבי משחקים יכילו בצורה ברורה ובמיקום עקבי את שמות השחקנים ששיחקו במשחק, הדירוגים שלהם, הצעד שכל שחקן עשה בהינתן מצב הלוח שלפני תורו, ועוד.

חשבנו על פורמט בינארי (שבבסיסו רעיון שמסתבר שגם הצוות של גוגל השתמש בו) והמרנו את רשומות המשחק לפורמט הנוח לנו.

הערכת המודל והפרמטרים

לאחר מכן כתבנו קוד שתפקידו לקרוא רשומות משחק, לדעת לעבד מצב לוח ולתת תחזית להתפלגות מהלכים עבורו.

במקביל, שינינו את הקוד במטרה לאפשר הרצה של הסוכן מול עצמו, קוד שבבסיסו עשינו בהמשך ניסויים ובדקנו השפעה של שינוי ערכי פרמטרים, וערכנו טורנירים.

איתרנו פרמטרים שניתן להוסיף ולעשות להם שינויים בסוכן המאומן, וניגשנו לראות את ההשפעה שלהם על תוצאותיו.

חשוב לציין שהמודל לא פעל בדיוק כמו alphaZero. המודל לא דגם את המהלך הבא שלו מתוך התפלגות של מהלכים, אלא לקח את המהלך שעבורו נחזתה ההסתברות הגבוהה ביותר. כמו כן, לא היו למודל מלכתחילה פרמטרים של טמפרטורה או קבוע uct.

אם כן, עטפנו את המודל ועשינו שינויים בתוך הקוד להוסיף את הפרמטרים.

שינינו את האסטרטגיה של הסוכן מלקיחת המהלך בעל ההסתברות הגבוהה ביותר ללקיחת דגימה מתוך התפלגות מהלכים.

הבחנו שלאחר העברת המודל לדגימה, המודל נטה לנפק עבור מצבי לוח התפלגויות יחסית ״שטוחות״ (כלומר, המהלך הטוב, או המהלכים הטובים, לא קיבלנו הסתברויות טובות בהרבה מאשר המהלכים הלא טובים), וכדי לשפר את התוצאות של המודל (בהרבה!), נדרש היה ״לחדד״ את ההתפלגות (להפוך הסתברות יחסית גבוהה ליותר גבוהה וההפך).

מודל לא מחודד מול מודל מחודד הפסיד 0:100.

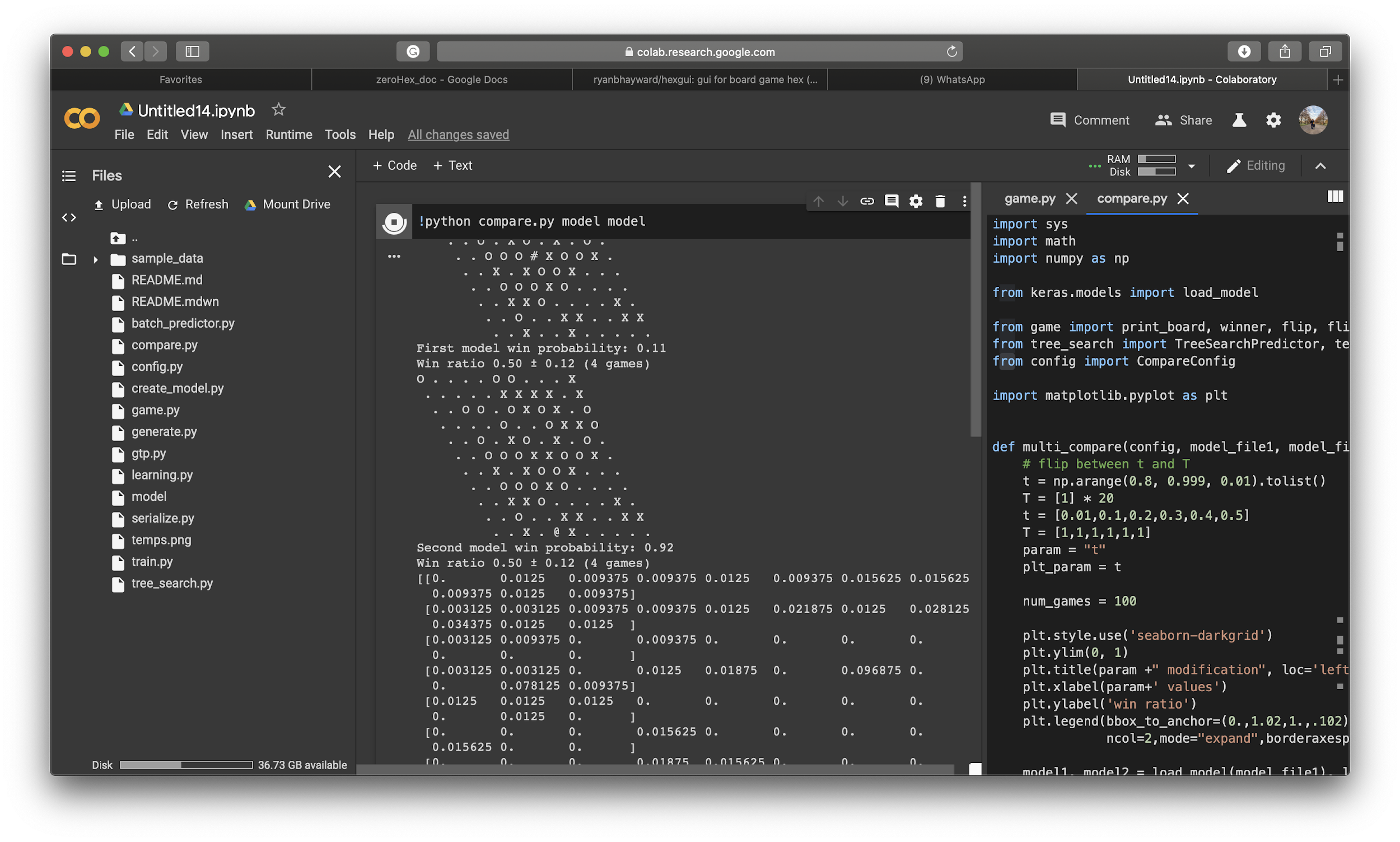
[מודל לא מחודד הינו למעשה מודל בעל פרמטרים ניטרליים (כגון ערכי טמפרטורה 1), וכפי שניתן לראות בהמשך, עבור השחקנים שלמדנו, ערכים אלה גרועים בהרבה מאשר הערכים של השחקן הגרוע ביותר שלמדנו].

השתמשנו בעיקר בפונקציונליות של הרצת הסוכנים אחד מול השני כדי לאמוד טיב של שינוי פרמטרים ואת ההשפעה שלהם.

עקבנו גם אחר לוחות המשחק, בעיקר בשלבים המתקדמים של המשחק, לראות דפוסים הגיוניים וחכמים של הסוכנים האיכותיים (ושטויות עבור סוכנים מקולקלים).

גם כמה המחשבים שברשותינו לא הספיקו לכל התחרויות והניסויים, ולכן חיפשנו פלטפורמת ענן עם חומרה של gpu שמאפשרת לעשות את ההרצות בזמן פחות בהרבה.

לבסוף השתמשנו בפלטפורמת google colab, על החסרונות שבו (לעיתים מקבלנו את החישובים ע״י פתיחת משתמשי גוגל חדשים).



אם כן, עשינו ניסויים ובדקנו אילו פרמטרים משפיעים על התנהגות הסוכן ואיך, והכנו גרפים שמראים אחוז נצחונות עבור הפרמטרים שהתמקדנו בהם - T טמפרטורה להסתברות בתוך הmcts, שכחלק ממדיניות הבחירה בין exploration לexplotation, וt טמפרטורה להתפלגות המהלכים הסופית הנחזית עבור מצב לוח.

שוב עשינו בדיקות של שינויי פרמטרים על הwinning ratio המתקבל מול המודל.

בעקבות תוצאות לא טובות במיוחד, עשינו כמה שינויים ששיפרו את הדיוק.

בחרנו להתעלם מהמלכים הראשונים של המשחק.

במהלכים אלו יש הרבה פחות אינפורמציה ופחות מעניין ללמוד, וכמו כן הגיוני שהתפלגות המהלכים עבור מהלכי הפתיחה תהיה דומה להתפלגות אחידה ולא מחודדת, מה שיגרור הרבה ענישה בloss לחינם.

עוד בחירה שעשינו היא להתייחס רק למשחקים בהם היו לפחות 22 מהלכים.

מטבע משחקים online, בפרט של שחקנים ברמות הנמוכות, לא כל המשחקים נגמרים או אפילו קרובים להסתיים.

הרבה פרשו מהמשחק באמצעו ואף בשלבים מוקדמים מאוד שלו.

בנוסף, במשחקים אלו מהלכים רבים ללא הגיון ואינטואיציה ברורים מאחוריהם.

שמנו לב שמשחקים אלו פחות אינפורמטיביים ולמידה עליהם לא תקדם אותנו.

שיפורים אלו הכפילו את אחוזי דיוק הניבוי.

הלמידה

ניגשנו להחליט כיצד לחלק את המידע המתוייג שאספנו, שכן לא שייך לערב בלמידה מהלכי משחק בדירוג נמוך עם מהלכי משחק עם דירוג גבוה.

על בסיס כמות המידע ואיכותו החלטנו לחלק את הדאטה לשלוש קבוצות של טווחי דירוג, ברמה נמוכה, בינונית וגבוהה.

השארנו margin בין הטווחים של קבוצות הדירוג, כדי לחדד את התוצאות שנקבל מהלמידה, בין רמות המשחק השונות שנלמד.

הדירוגים שבחרנו לבסוף ללמוד הם 1580~, 1830~ ו-2140~, כאשר לכל דירוג התייחסנו לכל המהלכים שאספנו מהשחקנים ברמות הדירוג בטווח 50 ממנו.

מספר מצבי הלוח המתוייגים לכל שחקן, בהתאמה, היו כ- 5300, 12100 ו-2700.

ניגשנו ללמוד את הפרמטרים המסויימים לכל אחד מטווחי הדירוג.

כתבנו פונקציית loss המקבלת תחזית של התפלגות מהלכים עבור מצב לוח (y hat) - וקטור בגודל 121 שרכיביו אי שליליים ונסכמים ל1, ומהלך אמת שקרה בפועל (אינדקס שמייצג את המשבצת בלוח שנבחרה), עבור שחקן שעליו מבצעים את הלמידה.

חשבנו על כמה פונקציות loss שונות וניסינו אותן, אך לבסוף בחרנו בפונקצייה שאינה מענישה אם אחד המהלכים האמיתיים נמצא בtop k של המהלכים בהתפלגות המהלכים שחזה המודל, ומענישה בnle עבור אפשרות אחרת.

הk נבחר לבסוף להיות 15.

מכיון שהlog עלול לפעול על הסתברות 0 (כלומר, הפרדיקציה עבור האינדקס האמיתי היא הסתברות 0), מה שיוביל לשגיאה, הוספנו epsilon=0.0001.

את הלמידה לא ביצענו בעזרת גזירת פונקציית הloss בהתאם לפרמטרים, כפי שעושים בלמידה רגילה, שכן הפונקצייה כאן קשה מאוד לגזירה.

במקום זאת ביצענו למידה בעזרת הגדרת הנגזרת, numeric gradient, כלומר חישוב loss עם הפרמטר ועם שינוי קטן שלו חלקי השינוי.

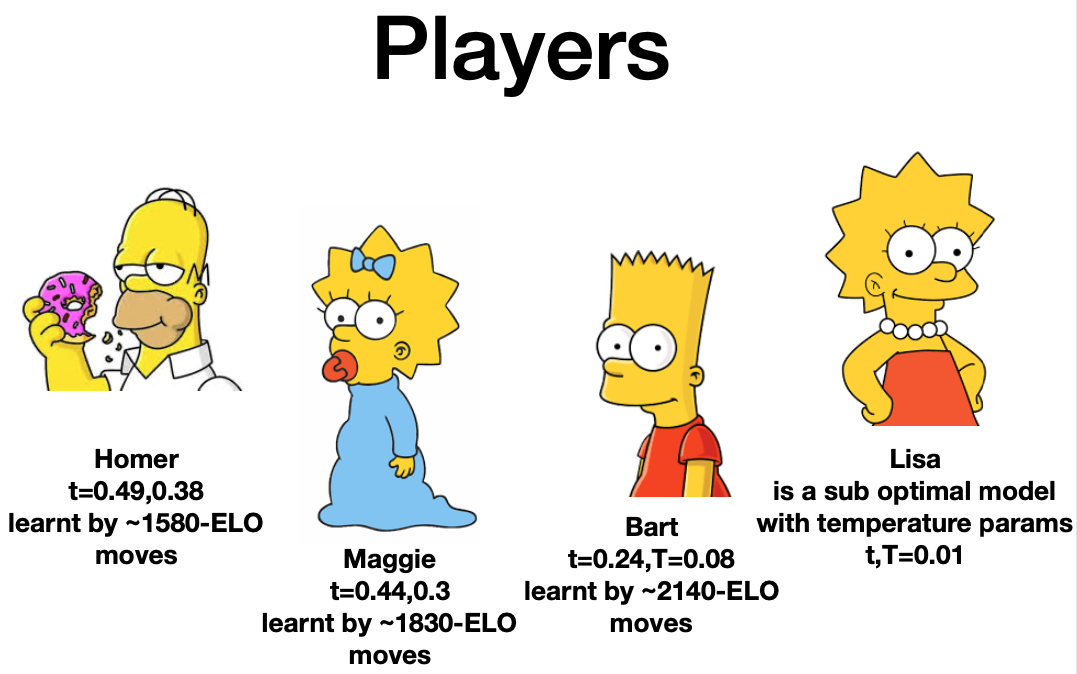
השינוי הקטן נבחר להיות 0.01.

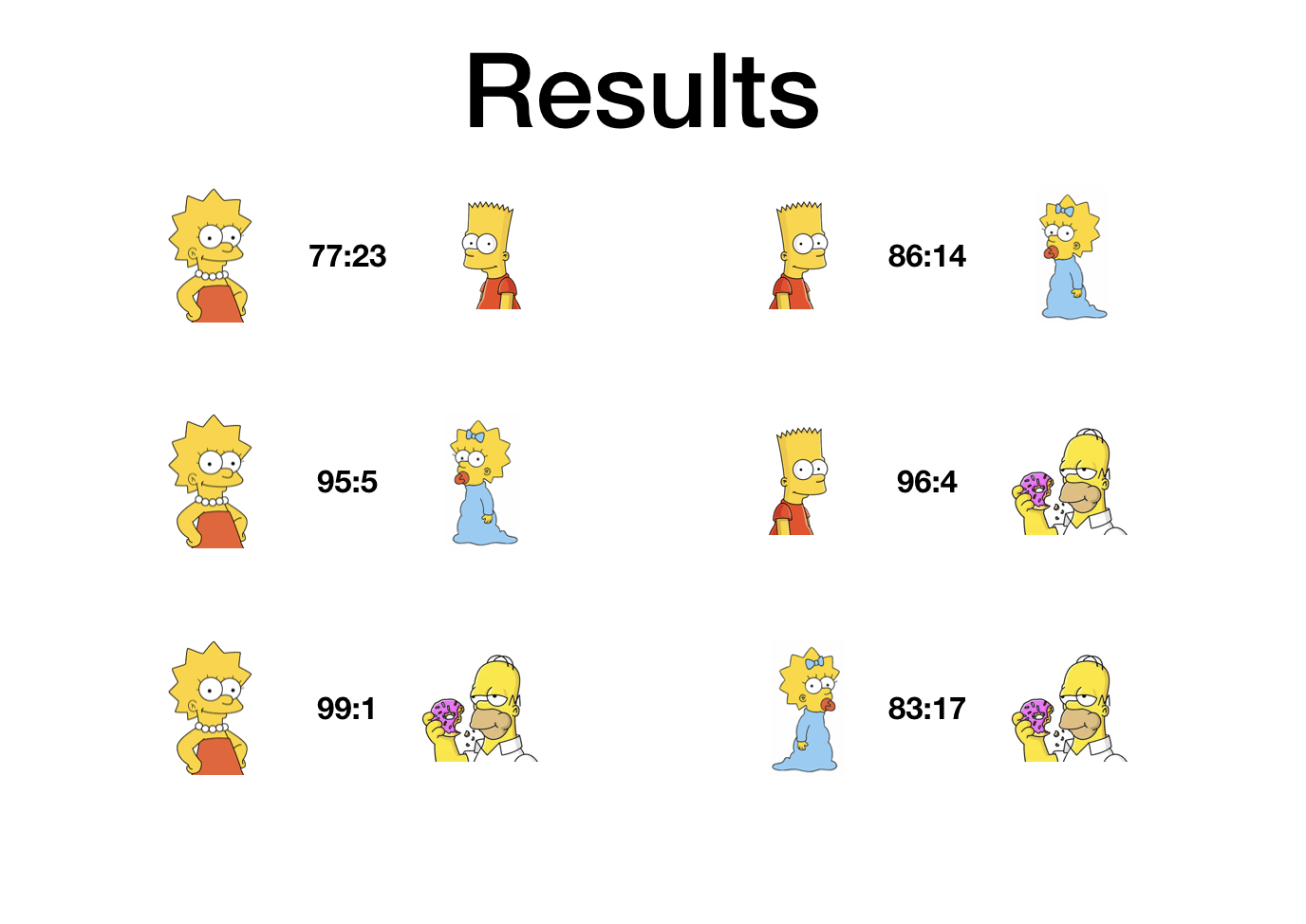
השתמשנו בoptimizer.

תוצאות

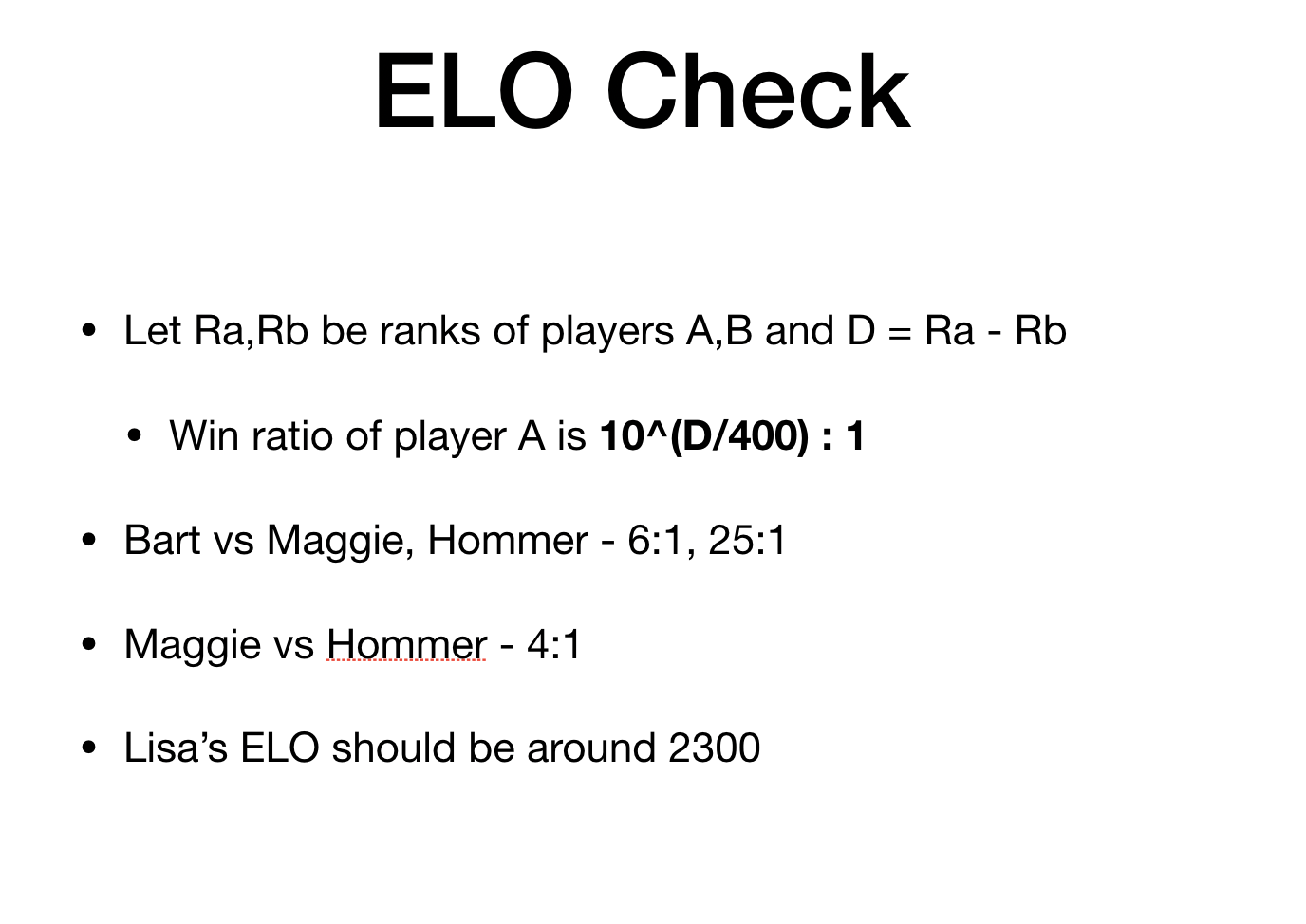
קיבלנו לבסוף את התוצאות הבאות, עבור שלוש רמות משחק שלמדו מהדאטה, ועבור המודל שהיווה מבחינתנו סטנדרט לשחקן איכותי מאוד:

ערכנו טורניר בין המודלים השונים ובדקנו את יחס התוצאות שלהם:





בדקנו את דירוג הelo וערכנו השוואה לדירוגים אותם ניסינו ללמוד, ואכן קיבלנו דמיון משמעותי:



משחק מול wolve

כעבודת המשך, ניסינו להסתנכרן עם הסוכן הקנדי wolve המשחק בשיטות הai המסורתיות.

את הסוכן ניתן למצוא כאן <https://github.com/cgao3/benzene-vanilla-cmake>.

שיחקנו מול הסוכן בעזרת הgui (לאחר כמה תיקונים):

<https://github.com/DebuggerOR/hexgui>

וראינו סגנון משחק שונה באופן מובהק מהסגנון של הסוכנים שלנו.

אכן, את הסוכנים שלנו הצלחנו לנצח, לעומת הסוכן הקנדי אותו לא הצלחנו.

סגנון המשחק היה שונה במובן זה שנראה שהסוכן הקנדי חשב כמה מהלכים הלאה (״לעומק״) והציב חיילים בנקודות מאוד מרוחקות בלוח, שרק בשלב מאוחר של המשחק התבררה חשיבות הצבתם שם.

הסוכנים שאנחנו אימנו, משחקים באופן אינטואיטיבי בהרבה. כל מהלך שלהם מובן והגיוני.

לאחר מכן ניגשנו לנסות להריץ את הסוכן הקנדי ואת הסוכנים שלנו זה מול זה.

בנינו סביבה שמריצה את הסוכן הקנדי והסוכנים שלנו ומעבירה את הקלטים והפלטים בין הסוכנים.

היינו צריכים ללמוד את פרוטוקול התקשורת של הסוכנים ולהשתמש בו.

תיעוד חלק מהפקודות:

genmove black

הסוכן משחק מהלך עבור הצבע השחור.

play black d4

הסוכן מציב חייל שחור במשבצת d4.

showboard

הסוכן מדפיס את הלוח לstdout.

boardsize 11

מגדיר את לוח המשחק כ11 על 11.

clear\_board

מנקה את לוח המשחק.

פקודות נוספות לwolve:

param\_wolve

מודפסת רשימה עם כל הפרמטרים של wolve.

param\_wolve max\_depth 2

שם בפרמטר עומק מקסימלי (בפיתוח העץ) להיות 2.

param\_wolve max\_time 2

שם בפרמטר זמן חיפוש מקסימלי להיות 2.

param\_wolve use\_cache\_book 0

שם בפרמטר האם להשתמש בספר הקש (מהלכים מיידיים) 0 (כלומר לא להשתמש).

הרצנו בסביבה טורניר בין הסוכנים שאימנו לבין wolve, לאחר ששמנו בפרמטרים של wolve את הערכים הגרועים ביותר האפשריים (ובמכונה וירטואלית דלת משאבים), ואלה התוצאות:



לאחר מכן הרצנו משחקים בין הסוכן המקורי שב<https://github.com/gwylim/hexnet>, ללא הרעה בביצועים כתוצאה מהאימון שלנו, וגם אותו הwolve ניצח ברוב המשחקים.

לאחר נסיונות נוספים (בקנפוג חומרה), הצלחנו לבצע טורניר יחסית שקול בין wolve לסוכן שלנו.

שמרנו את רשומות המשחקים.

כפרוייקט המשך, שווה לנסות לאמן מחדש את הhexnet, עם דאטה נוסף ועל חומרה טובה, ולבדוק את ביצועיו מול wolve.