**סיווג חשבונות בוטים ב-Twitter**

**מגישים:**

**שם:** אסף חאייק ברוך **ת.ז:** 206783441  
**שם:** בן בנוז **ת.ז:** 207570573

**מנחים:**

אלעד נחמיאס ושאול מרקוביץ'





**הקדמה**  
כיום הרשתות החברתיות הן חלק משמעותי ובלתי נמנע מחיי היומיום שלנו בעולם המערבי.  
יש להן השפעה רבה על האופן שבו העולם מתנהל מבחינת פרסומות, ארגון קהילתי, פוליטיקה ודברים חשובים אחרים.

דבר שיוצר בעיות ביחד עם ההשפעה הזאת הוא קיומם של חשבונות מזויפים שעושים פוסטים באופן אוטומטי ומנסים להטות את דעת הקהל.  
מדברים קטנים כמו יצירת פרסום חינם למוצר עד לדברים מהותיים ביותר כמו השפעה על בחירות - הבוטים האלה גורמים לערעור היציבות של המרקם החברתי.

המטרה שלנו היא שבהינתן משתמש ב-Tweeter נרצה לסווג אם אותו המשתמש מופעל ע"י אדם אמיתי או ע"י בוט.

ניסינו גם להוסיף לציוצים של המשתמשים הקשר של מידע חיצוני באמצעות שימוש ב-GDELT בפרויקט הזה – מתוך מחשבה שהקשר של אירועים שקרו בזמן הציוצים יוכל לעזור לנו.

**תיאור הבעיה:**  
כיום הרשתות החברתיות הן חלק משמעותי ובלתי נמנע מחיי היומיום שלנו בעולם המערבי.  
יש להן השפעה רבה על האופן שבו העולם מתנהל מבחינת פרסומות, ארגון קהילתי, פוליטיקה ודברים חשובים אחרים.

דבר שיוצר בעיות ביחד עם ההשפעה הזאת הוא קיומם של חשבונות מזויפים שעושים פוסטים באופן אוטומטי ומנסים להטות את דעת הקהל.  
מדברים קטנים כמו יצירת פרסום חינם למוצר עד לדברים מהותיים ביותר כמו השפעה על בחירות - הבוטים האלה גורמים לערעור היציבות של המרקם החברתי.

המטרה שלנו היא שבהינתן משתמש ב-Tweeter נרצה לסווג אם אותו המשתמש מופעל ע"י אדם אמיתי או ע"י בוט.

**פתרון הבעיה:**  
הדרך שחשבנו שאיתה אפשר לפתור את הבעיה היא הדרך הבאה:

1. שימוש במאגר נתונים של משתמשים מסווגים.
2. ניקח מכל משתמש את האינפורמציה הרלוונטית עליו ועל כל הפוסטים האחרונים שלו.
3. נמיר את אובייקט המשתמש לטנזור פיצ'רים באמצעות Feature Extractor שנגדיר וכנראה נלמד, כולל המרת מילים לוקטורים באמצעות ספריה מסוג Word2Vec והוספת הקשר חיצוני מ-wikidata ומ-GDELT.
4. לאחר עוד כמה שלבי עיבוד מקדים (כולל סטנדרטיזציה) נעביר את הטנזור ואת הסיווגים לאלגוריתם למידה עמוקה.

**ספריות חיצוניות ותלויות**

Tweepy- זוהי הספרייה שבה השתמשנו כדי להתחבר לטוויטר ולשלוף ממנו את כל המידע של המשתמשים במאגר המידע שלנו. כדי להשתמש בספרייה היינו צריכים להירשם בתור מפתחים לטוויטר ולקבל מהם מפתח להתחברות לממשק. לקחנו את מזהי המשתמשים ממאגר המידע באמצעותם שלפנו מידע של המשתמשים ושמרנו באובייקטים שהגדרנו שמכילים את כל המידע על המשתמש ואת אותם אובייקטים שמרנו בקובץ לשם שליפה מהירה של המידע בזמן ריצה של המודל.

Langdetect- ספרייה שבה השתמשנו בשביל לזהות את שפת הציוצים של המשתמשים במאגרי המידע שלנו זאת כדי לסנן משתמשים שהציוצים שלהם לא באנגלית.

Genism- השתמשנו בספריית genism בשביל השימוש שלנו בword2vec. כדי להשתמש בword2vec היינו צריכים לבצע עיבוד מקדים על הטקסט של הציוצים של המשתמשים כדי שיוכלו לשמש כקלט למודל הנ"ל.

Nltk- השתמשנו בספרייה זו בשביל תהליך הtokenization שהציוצים אמורים לעבור לפני הכנסתם כקלט לword2vec ובנוסף בשביל מאגר של stop words באנגלית כלומר מילות הקשר ותחביר שאין צורך להרחיב בהגדרותיהם בשלב השימוש במקורות מידע חיצוניים ולכן התעלמות מהן בשלב הזה היא באמצעות ספרייה זו.

Pywikibot- זהו בוט של wikidata איתו יכולנו לשלוף מידע על כל מיני מונחים ואישים חשובים ממאגר המידע של wikidata. לשם השימוש בו היינו צריכים להגדיר קבצים עם פרטי התחברות לאתר wikidata שבאמצעותם הממשק שולף מידע מהאתר.

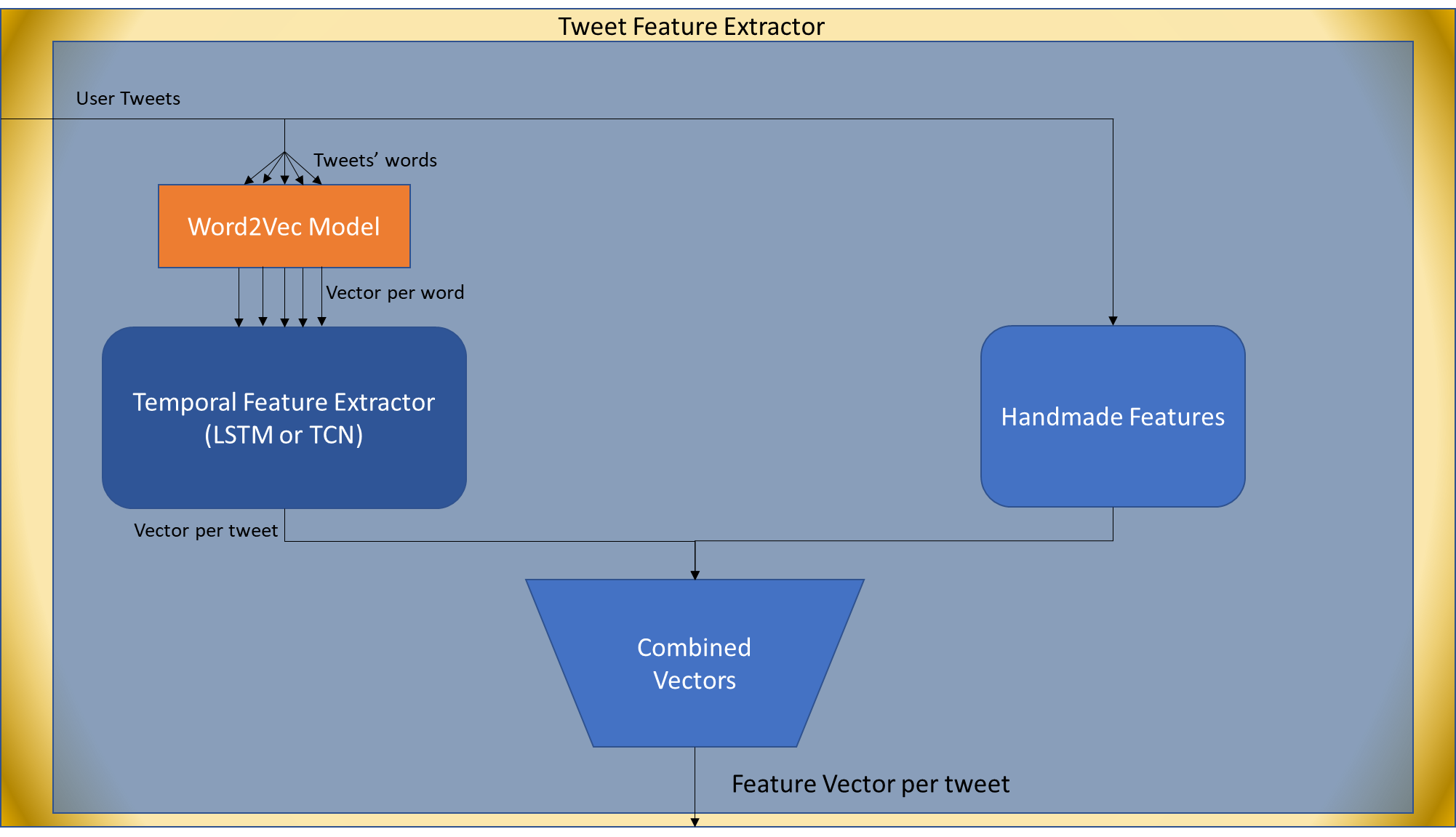
Gdelt- בהתחלה השתמשנו בספרייה gdelt כדי לנסות לשלוף מידע ממאגר המידע הזה אך המהירות בו השאילתות של שליפת המידע חזרו היו איטיות מדי בשביל כמויות המידע והזמן שהיה לנו, ולכן ניסינו להשתמש בgoogle BigQuery ונרשמנו לשירות הנ"ל של גוגל כדי לקבל גישה באמצעות שאילתות יותר מדויקות ומהירות למאגר הנתונים gdelt . לאחר ביצוע כמה שאילתות התברר כי השימוש בשירות עולה כסף ולכן השתמשנו באופצייה האחרונה של הורדת מקטעים של מאגר המידע לוקלית, ולכן כתבנו סקריפט שמוריד את הקבצים המתאימים לתאריכים מהם אנחנו רוצים לשלוף מידע במאגר, סידרנו את המידע ושלפנו ממנו בדיוק את המידע שלו היינו זקוקים לזמן האימון של המודל ושמרנו אותו בקובץ json לשליפה מהירה יותר מאוחר.

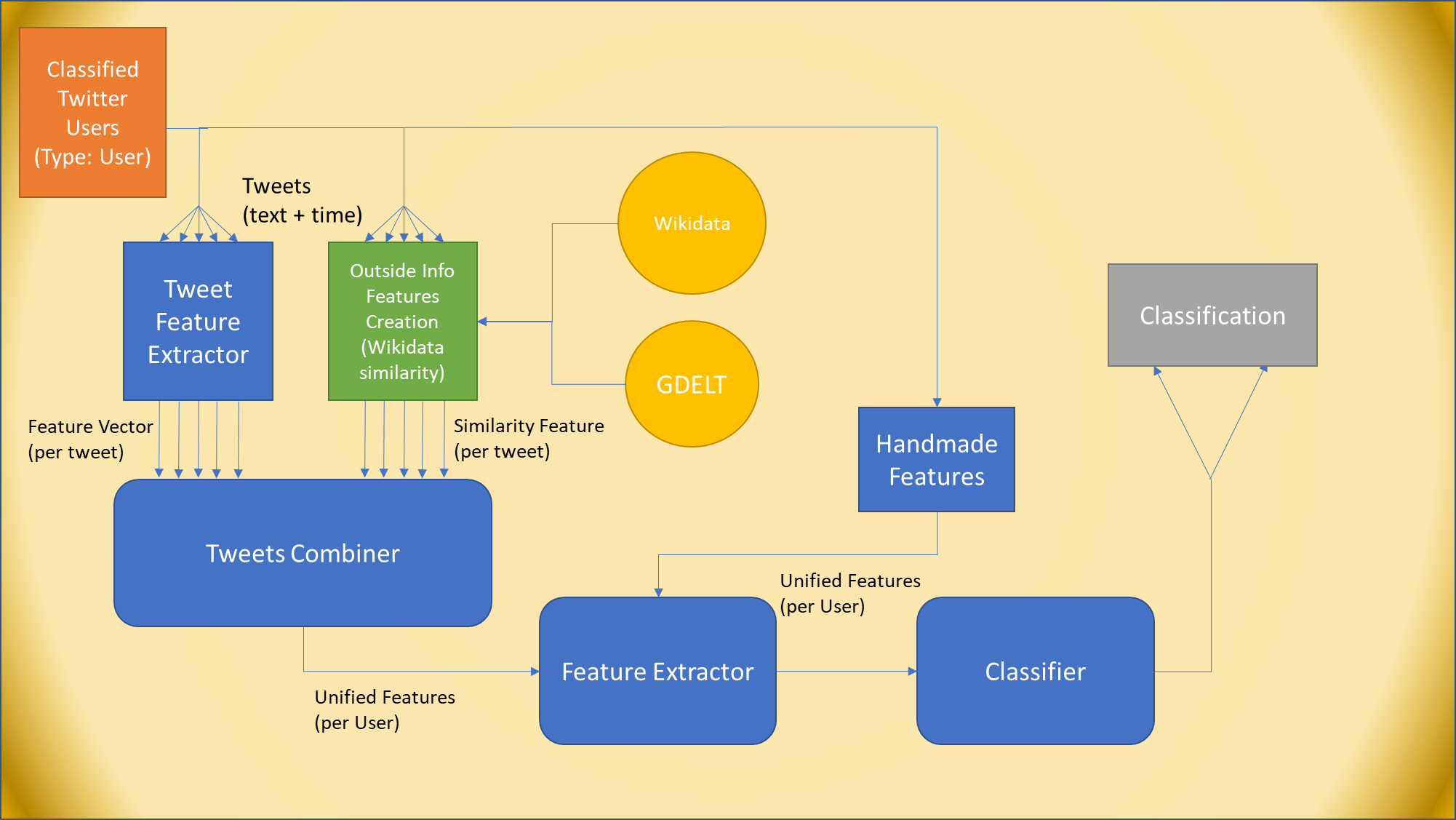
Pytorch- השתמשנו בספריית Pytorch בשביל כל המודלים של הלמידה חוץ ממודל ה-WordToVec, כלומר בכל מקום שמתואר בלוק או מודל של למידה בתוך המודל שלנו מדובר באחד שנכתב עם החבילה הזאת. בפרט ה-LSTM נמצא ישירות ב-Pytorch בעוד שה-TCN נמצא בספרייה אחרת אך ממומש ב-Pytorch בתוכה.

Astar-TCN- זאת הספרייה שבה ממומש מודל ה-TCN שהשתמשנו בו.  
היא כתובה באמצעות Pytorch.  
הספרייה מבוססת על המאמר שבו הוגדר המושג TCN במקור.

Matplotlib- השתמשנו בספרייה הזאת בכדי ליצור גרפים מתוצאות האימון (בפרט pyplot).

Scikit-learn – השתמשנו בספריה בשביל אלגוריתמי למידה קלאסיים (מסווגים ו-clustering), בנוסף למטריקות ולפונק' חלוקת סט מבחן ואימון באופן stratified.

**תיאור המערכת:**

****

**מאגר נתונים + יוצר אובייקטים:**  
מאגרי המידע שלנו מורכבים ממשתמשים אמיתיים (כלומר שמאחוריהם עומדים אנשים אמיתיים) ובוטים ברשת החברתית טוויטר. הם מכילים את מזהה המשתמש בטוויטר ואת סיווגו כבוט או לא בוט.

השתמשנו במאגר varol-2017 שניתן למצוא בקישור:

<https://botometer.iuni.iu.edu/bot-repository/datasets.html>

**שליפת מידע ואחסונו**:  
כדי לגשת למידע המלא על כל משתמש כמו הציוצים שלו והפרופיל שלו היינו צריכים להשתמש בממשק של טוויטר tweepy . זה הוא api שבהינתן מזהה משתמש בטוויטר נותן גישה מלאה לפרופיל שלו ולציר הזמן שלו ולציוצים שלו. לכל משתמש במאגר ביצענו אחסון של כל המידע שיכולנו לקחת ממנו באובייקטים מסוג user שיצרנו שמאפשרים גישה נוחה ומהירה לכל המידע על המשתמש שנרצה לעשות בו שימוש. לאחר מכן לקחנו את האוביקטים והסיווג שנוצרו לכל המשתמשים ושמרנו בקובץ כדי שלאחר מכן נוכל לטעון את המידע במהירות ובפשטות בזמן ריצת המודל.

**סינון מידע:**  
בעת הגישה למשתמשים באמצעות tweepy קיימים משתמשים שהוסרו מטוויטר או שהפכו את הגישה לפרופילים שלהם לפרטיים ולכן לא ניתן לגשת אליהם ולקחת מהם מידע ולכן הסרנו אותם מהמאגר.

בנוסף הניתוח שהמודל שלנו יעשה בהמשך על המידע של המשתמשים דורש ששפת הציוצים שלו תהיה באנגלית ולכן משתמשים שציוציהם לא היו באנגלית הוסרו גם הם מהמאגר.

**עיבוד מקדים של הציוצים לפני השימוש ב-word2vec:**\*הסבר קצר על word2vec- מדובר ברשת נוירונים רדודה שמקבלת סט רחב של מילים ושולח כל אחת מהמילים לווקטור שונה משלה בעל מימדים גדולים כאשר מילים שחולקות הקשר נשלחות לווקטורים שהמרחק ביניהם יותר קטן.

הציוצים אותם אנחנו מקבלים מהמשתמשים חייבים לעבור "ניקוי" כדי ש-word2vec יוכל לעבד אותם בצורה טובה ומועילה.

ראשית ציוצים מכילים בדרך כלל תיוג של משתמשים אחרים ברשת על ידי התו '@' ולאחר מכן שם המשתמש , לכן כדי שword2vec יוכל לתפוס אזכור של משתמש בתור קונספט במשפט ולא בתור מילה ספציפית ומיוחדת עבור כל אזכור שונה בכל ציוץ החלפנו את אזכורי המשתמשים בהופעת התו '@' בלבד.

לאחר מכן שימוש בקישורים בציוצים צריך לעבור אותו תהליך כי אינינו רוצים ש-word2vec יסתכל על כל קישור כמושג נפרד ושונה בעולם אלא כקונספט הכללי של לשים קישור בציוץ ולכן החלפנו הופעה של כל קישור בתו '$'.

לבסוף הסרנו את כל שאר התווים המיוחדים מציוצים שכן הם לא תורמים ואף גורעים מההבנה של word2vec את ההקשר של המילים.

לאחר מכן המילים של כל ציוץ עוברות tokenization כלומר הציוץ מפורק לרשימה של המילים במרכיבות אותו כאשר כל מילה או פתגם או ביטוי הופך לאיבר ברשימה זו כלומר לtoken. את הtokens ניתן להכניס לword2vec בתור קלט שהוא יכול להתאמן עליו וגם לפלוט ווקטורים מתאימים בזמן ריצת המודל המלא.

**יצירת ה-features:**

**המרת מילים לוקטורים באמצעות word2vec:**  
נמיר כל מילה (לאחר העיבוד המקדים שתואר קודם) לוקטור באמצעות מודל word2vec של genism.  
המודל בנוי מרשת נוירונים רדודה עם 2 שכבות, ונוטה לתת קידוד טוב של מילים למטלות למידה.  
מידע נוסף על המודל ניתן למצוא במאמר  
Distributed Representations of Words and Phrases and their Compositionality[[1]](#footnote-1).

**Tweet Feature Extractor:**מכיוון אנו יכולים להשתמש ב-word2vec רק בכדי לקודד מילים, עלינו להשתמש במודל שיכול לעבד וקטורים במימד הזמן (סדר המילים) בכדי להשיג וקטור מידע על *הציוץ*.  
ישנן 2 גישות מרכזיות לטפל בעיבוד מידע לאורך מימד הזמן:

1. הגישה הראשונה היא שימוש ברשתות נוירונים מסוג RNN (Recurrent Neural Network), שבהן מכניסים לשכבה כלשהי את הוקטורים אחד אחרי השני, עם שמירת מצב חבוי שחושב מהוקטור הקדם ומהמצב החבוי שלפניו (אפסים לראשון).  
   מתוך אלה בחרנו להשתמש במודל ה-LSTM (Long Short Term Memory), מכיוון שהמודל הזה נוטה לתת את הפתרונות הכי איכותיים ואת זמני האימון הכי קצרים.
2. הגישה השנייה היא להשתמש בקונבולציות זמן (t-convolutions) על הוקטורים.  
   אלו רשתות קונבלוציה שעובדות על קלט חד מימדי, כאשר התכונות השונות מיוצגות כערוצים והערכים לאורך הזמן כמימד הוקטור (כלומר עבור מימד זמן T ו-F תכונות זה יהיה וקטורים T-מימדיים עם F ערוצים).  
   בפרט בחרנו להשתמש במודל ה-TCN (Temporal Convolutional Network) שהופיע במאמר שהציג לראשונה את הקונספט של קונבולוציות זמן   
    An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling[[2]](#footnote-2).

לכל אחד מהפתרונות יש יתרונות וחסרונות ביחס לאחרים:

* מבחינת זמני ריצה, בגדלי מימד הזמן שאנו עובדים איתם LSTM מהיר בהרבה מ-TCN (נבדק אמפירית באימון שלנו). זה הגיוני מכיוון ש-TCN מתייחס לכל דגימות ב-batch כאילו הן באותו האורך, בעוד ש-LSTM יכול להימנע מחישובים מיותרים.  
  נשים לב כי עבור מימדי זמן גדולים יותר TCN יכול לעבוד מהר יותר.
* מבחינת מספר הפרמטרים וסביכות LSTM קבוע ללא קשר לגודל מימד הזמן, ומספר הפרמטרים של TCN קשור ישירות למימד הזמן המקסימלי שהוא יכול לעבד.  
  זהו יתרון של LSTM ברוב המקרים, אך כפי שגילינו באנליזה שנמצאת בנספח שבסוף הדו"ח, במקרה שלנו מימד הזמן המקסימלי גורם לכך שלשני המודלים מספר פרמטרים באותו סדר הגודל כתלות במימד החבוי (של ).
* יתרון משמעותי של TCN על פני LSTM הוא אורך שרשרת הגרדיאנטים – ב-LSTM אורך השרשרת תלוי במימד הזמן ויכול לגרום לבעיית 'הגרדיאנט הנעלם' (Vanishing Gradient), בעוד שב-TCN אורך השרשת קבוע ותלוי במספר השכבות ולכן הבעיה נפטרת.  
  לא ראינו בעיות בהתכנסות המודל שלנו עם LSTM, כנראה מכיוון שהרכיב הזמני של המודל שלנו הוא הרכיב הראשון שלומד ואף חלק אחר לא מסתמך על שרשרת הגרדיאנטים שלו.

הוצאת מידע מציוצים

**מקור הרעיון:**מקור המידע הראשי שממנו נרצה לשאוב מידע על המשתמש הוא ציוציו, לציוצים קיימים מספר תכונות שהשתמשנו בהם על מנת להוסיף עוד מידע למודל כדי שיוכל להשתמש בו לסיווג.

מאפיין מאוד בולט של בוטים ברשתות חברתיות הוא מספר גדול של ציוצים בהפרשי זמנים מאוד קצרים, ולכן משתמש שהפרשי הזמן בין ציוציו נורא קטנים יותר סביר להיות בוט, בנוסף הוספנו עוד תכונות של ציוצים כמו האם הציוץ הוא ציטוט או לא כערכים 0 או 1 , מספר הקישורים או אזכורים של משתמשים אחרים בציוץ, האם היה retweeted status לציוץ או לא כערכים 0 או 1, ומספר המשתמשים ששמו את הציוץ במועדפים שלהם.  
חישבנו את התוחלות ואת סטיות התקן של התכונות האלו בכדי לעשות סטנדטיזציה.  
הערכים האלה הם:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **תכונה:** | מס' אנשים שסימנו כמועדף | האם זה ציטוט | האם זה ציוץ מחדש | מס' קישורים ואזכורים |
| **תוחלת:** | 1.0901 | 0.0135 | 0.8929 | 0.8929 |
| **סטיית תקן:** | 17.6291 | 0.1156 | 0.3092 | 1.9421 |

**התהליך:**

1. שליפת התאריכים של הציוצים של משתמש מתוך מאגר המידע של המשתמשים שברשותנו, בנוסף נשלוף את היותו של ציוץ ציטוט או לא, מספר הקישורים והאזכורים של משתמשים אחרים בציוץ, והאם היה retweeted status, ומספר המשתמשים ששמו את הציוץ במועדפים שלהם.
2. חישוב של הפרשי הזמן בימים בין כל שני ציוצים עוקבים שלו.
3. שמירה של הפרשי הזמנים וכל אחד משאר הנתונים בווקטור מתאים משל עצמו לכל הציוצים.
4. הוספת הווקטורים בתור עוד features לפלט של הword2vec על מילות הציוצים, לאחר סטנדרטיזציה לפי תוכלת וסטיית תקן.

הוצאת מידע על user:

מהמשתמש עצמו ניתן להוסיף עוד תכונות כמו כמות העוקבים שלו וכמות החברים שלו, כאשר אינטואיציה אומרת שעבור בוט מספר העוקבים והחברים יהיה נמוך יותר למרות שקיימים בוטים שמתחזים טוב לאדם אמיתי והם בעלי מספר עוקבים גדול אך הנחנו כי הם חריגים, בנוסף מהציוצים ניתן להסיק מידע כללי על המשתמש ולכן בחרנו לקחת את ממוצע הפרשי הזמנים בין ציוצים עוקבים של המשתמש כאשר בוטים יהיו בעלי ממוצע יותר נמוך של הפרשי זמנים.  
גם לערכים האלה שמרנו תוחלות וסטיות תקן בכדי לעשות סטנדרטיזציה.  
התוחלות וסטיות התקן הן:

|  |  |  |  |
| --- | --- | --- | --- |
| **תכונה:** | מס' עוקבים | מס' חברים | הפרש זמן ממוצע בין ציוצים (בשניות) |
| **תוחלת:** | 3.2708e+03 | 6.6564e+02 | 5.6675e+04 |
| **סטיית תקן:** | 2.1951e+04 | 1.3241e+03 | 2.2061e+04 |

**התהליך:**

1. לאחר הוצאת התכונות של הציוצים מה-feature extractor נחשב את ממוצע הפרשי הזמנים בין ציוצים שונים של המשתמש ונשלוף לכל משתמש את מספר העוקבים והחברים שלו ונאחסן כל תכונה בווקטור מתאים משלו.
2. נבצע סטנדרטיזציה על וקטורי התכונה האלה.
3. נוסיף את הווקטורים כעוד features לפלט של ה-tweet combiner.

**Combiner:**  
לאחר יצירת התכונות של הציוצים אנו מעבירים אותם דרך שכבה לינארית שיוצרת וקטור תכונות משולב באורך 256 לכל ציוץ.  
אחרי זה אנו יוצרים מכל הציוצים של כל משתמש וקטור תכונות אחד גדול (בגודל 100\*256) ומעבירים אותו בשכבת BatchNorm ואז Relu, שאחריהן ישנה שכבה לינארית בשם combiner שיוצרת לכל משתמש וקטור תכונות בגודל 1024.  
לאחר מכן אנו מוסיפים לוקטור מספר תכונות ידניות שמתוארות למטה:

שימוש במקורות מידע חיצוניים

**מקור הרעיון:**בוטים ברשתות חברתיות כמו facebook ובמקרה שלנו twitter נוטים להיות בוטים שמצייצים בנושאים שהם כרגע "חמים" במדיה ובתקשורת במטרה לפגוע בנושאים שהם כרגע רגישים ולהטות דעת קהל של משתמשים אחרים ברשתות החברתיות. לכן משתמש שהציוצים שלו הם בעלי הקשר גבוהה לנושאים שמוזכרים הרבה בתקשורת באותה תקופה הוא משתמש שסבירותו להיות בוט פוליטי עולה, יש לציין שסדרת ציוצים כזו מאופיינת במרווח נורא קטן של זמן בין הציוצים השונים וגם בכך נעשה שימוש בנוסף לעוד מידע מהציוצים.  
על המדד הזה גם נעשה סטנדרטיזציה, יש לו תוחלת של 0.66 וסטיית תקן של 2.8933.

**כלים:**

1. Wikidata- מאגר מידע שמחזיק הגדרות, הסברים ומידע כללי עבור מונחים ואישים חשובים, באמצעותו אנחנו יכולים לחפש מילה במאגר המידע הזה ולשלוף כל מיני הגדרות והסברים עליה ובאמצעות כך להרחיב את ההקשר שלה במשפט.

(שליפת המידע ממאגר זה בוצעה באמצעות בוט שמתחבר לאתר ושולף ממנו את המידע המבוקש).

1. Gdelt- מאגר מידע שמחזיק מידע על אירועים כלל עולמיים שדובר עליהם בעיתונים ובתקשורת מסביב לעולם. מאגר המידע מסודר לפי תאריך ומכיל שמות של גורמים שמעורבים באירוע כמו מדינות, ממשלות, פוליטיקאים, ארגונים ומוסדות, ומכיל גם מדד לחשיבות הנושא בשם GoldensteinScale שמצביע על כמה שנושא קריטי למדינה או לארגון בו האירוע קרה, בשביל עוד מידע עליו ניתן לקרוא בראש העמוד בקישור הבא:

<http://web.pdx.edu/~kinsella/jgscale.html>

(שליפת המידע ממאגר מידע זה בוצעה על ידי הורדה של מידע מהמאגר באירועים שנמצאים רק בתאריכים שקרובים לציוצים של אחד המשתמשים שלנו, ולאחר מכן מיון ושליפת מידע מהקבצים שהורדו)

1. SequenceMatcher- כלי שמספק מדד להקשר בין מילה למשפט לפי חיפוש הופעות של המילה ומילים מאותו שורש במשפט, בסופו של דבר בהינתן מילה ומשפט מחזיר מספר שככל שגודלו גדול יותר הוא מצביע על קשר טוב יותר בין המילה למשפט.

**התהליך:**

עבור כל משתמש ביצענו את התהליך הבא:

1. לקחנו את תקופת הזמן שבו הוא צייץ בתדירות הכי גבוהה ומשם לקחנו 5 מהציוצים שלו (נלקחו רק 5 בגלל חסכון בזמן ריצת המודל)
2. באמצעות הכלי gdelt לקחנו את 10 הנושאים שהיו בעלי החשיבות הגבוהה ביותר באותה תקופה וזאת באמצעות מדד לחשיבות הנושאים שgdelt מספק בשם GoldsteinScale.
3. לאחר מכן לקחנו את הטקסט של כל אחד מ5 הטוויטים ולקחנו עבור כל מילה בציוצים את הגדרתה והרחבתה באמצעות הכליwikidata כאשר מילים שהם מילות הקשר או עצירה לא השתתפו בהרחבה הזאת מכיוון שהם לא יספקו מידע משמעותי שרלוונטי לזיהוי נושאים פוליטיים.
4. לאחר מכן השתמשנו בכלי בשם sequenceMatcher כדי לאמוד את טיב הקשר בין הנושאים שהיו חמים בתקופת הציוץ לבין הטקסט של ציוצים כולל ההרחבות שלהם שהתווספו לטקסט מwikidata, פשוט סכמנו את הפלט של הsequenceMatcher עבור כל אחד מהציוצים של משתמש עם הנושאים החמים שלו לכל אחד מהמשתמשים.

נשמור את מדדי ההקשר האלו של כל המשתמשים בווקטור אותו אנחנו מנרמלים ומוסיפים כעמודה נוספת לפלט של ה-feature extracture כמידע נוסף על כל user.

**Extractor:**  
לאחר יצירת הפיצ'רים הידניים אנחנו מעבירם את וקטור התכונות של כל משתמש בשכבות BatchNorm ואז Relu שלאחריהן שכבה לינארית שיוצרת וקטור תכונות באורך 1024 לכל משתמש.  
במצב הרגיל אחריה יש ראש מסווג לינארי רגיל (כפי שיפורט בהמשך) אך בניסויים שלנו ניסינו לראות אם שימוש במסווגים שאינם של למידה עמוקה לאחר השלב הזה משפיע על יכולת הסיווג שלנו.

אימון:  
אנחנו אימנו את המודל למשך 60 אפוקים, עם גודל batch של 8 דגימות וגודל סט אימון של 80%.  
השתמשנו ב-SGD עם קצב למידה של ורגולריזצית L2 עם מקדם של למשקלים.  
פונק' ההפסד שלנו היית cross-entropy.  
במהלך האימון השתמשנו במסווג הלינארי המקורי (שמתואר למטה) בכדי שתהיה שרשרת גרדיאנטים מפונק' ההפסד באופן שיאפשר למידה עמוקה.  
במהלך האימון שמרנו תוצאות, כך שבסוף שמרנו והשתמשנו בגרסא של המסווג שהיה לה את הדיוק הכי גבוה על סט המבחן.  
סט המבחן והאימון חולקו באופן stratified, ווידאנו שהם היו אותם הסטים לכל הניסויים והאימונים.

סיווג:  
המסווג המקורי שלנו היה שילוב של BatchNorm, אז RelU ואחרי שכבה לינארית שמחזירה 2 logits. את ה-logits העברנו ב-Softmax בכדי לקבל התסברויות שסכומן 1 ושמתאימות לסיווג רגיל.  
ניתן להרכיב מסווגים אחרים במקום השכבה הלינארית הזאת, דבר שניסינו לעשות לאחר תהליך האימון בכדי לראות אם יצרנו תכונות שמכילות מידע שימושי באופן כללי מעבר למסווגי למידה עמוקה.  
על המסווגים האחרים שניסינו מפורט בהמשך הדו"ח, בחלק הניסויים.

**ניסויים:**

* נרצה לבדוק השפעה של מסווגים שונים עם היפר פרמטרים שונים.
* נרצה לבדוק את ההשפעה שיש בכלל להקשר החיצוני שאנו מוסיפים לציוצים.
* השוואה בין LSTM ו-TCN כ-Temporal Extractor.
* בדיקת מסווגים אחרים על ה-features שניתן לקבל לאחר סוף האימון.
* בדיקת clustering על המידע בכדי שנוכל אולי להשיג עוד אינפורמציה על המשתמשים, בפרט על הפרדה בין סוגי בוטים.

מטריקות לתוצאות הסיווג:  
המטריקות שלפיהן מדדנו את ביצועי המסווגים שלנו הן:  
accuracy, precision, recall, f1-score.  
נשים לב כי 73.7% מן הדגימות במאגר שלנו הם בוטים, ולכן כל ערך דיוק שמתחת למספר הזה שקול לסיווג כל הדגימות כבוטים. לכן כל ערך דיוק שקטן או שווה ל-73.7% נחשב כישלון.

בנוסף לכך השתמשנו באופן שבו tsne מציג את המידע בשני מימדים בכדי לאמוד את יכולות ה-clustering של המודל שלנו.

פרטי הניסויים:  
***ניסוי מצב המודל לאורך האימון:***

נציג את שינוי הדיוק ופונק' ההפסד על סטי האימון והמבחן לאורך האימון, וננסה להסיק מסקנות מן התוצאות, שיוצגו אמצעות גרף.

***ניסוי מסווגים קלאסיים:***

נהפוך את המסווגים המאומנים ל-feature extractors באמצעות הוצאת הראש המסווג ונבדוק את יכולתם של מסווגים קלאסיים שאינם של למידה עמוקה לפטור את הבעיה.  
המסווגים שבדקנו הם:

**KNN:**  
בדקנו שתי מטריקות למשקול K השכנים הכי קרובים:

* Uniform, שבה לכל שכן יש משקל זהה.
* Distance, שבה המשקל נמצא ביחס הפוך למרחק השכנים מהנקודה.

הערך שלאורכו בדקנו את ביצועי המסווג היה מספר השכנים K.

**SVM:**  
בדקנו את 4 ה-kernel'ים הרגילים (Linear, Poly w/ deg=3, Rbf, Sigmoid),  
ובדקנו את ביצועי המסווגים לאורך ערכי C.

**עץ החלטה:**  
בדקנו עצים שונים לפי מספר הפיצ'רים המקסימלי שהם ייקחו (כולם, 60%, log2, שורש ריבועי ו-80%) לאורך הפרמטר min\_samples\_split, שבוא מספר הדגימות המינימלי שמותר שיופיעו בתוך עלה בעץ.

**Random Forest:**  
במקרה הזה השוונו את הביצועים של יערות עם מספר עצים שונים לאורך הפרמטר min\_samples\_split (זהה להגדרה בעץ החלטה).

**AdaBoost:**  
בדקנו רק את המודל עם עצי החלטה בעלי עומק של 1, ולאורך הפרמטר של מספר המסווגים.

***ניסוי בדיקת clustering (בפרט tsne)****:*

נעביר את הפיצ'רים שנוצרו על סטי המבחן והאימון (בנפרד) דרך אלגוריתם tsne למרחב דו-מימדי וסימנו את הסיווגים בצבעים שונים.  
ננסה לראות אם המרחב שכל מודל מספק לנו מאפשר הפרדה נוחה בין הסיווגים, ואם יש חלוקות נוספות של המידע מעבר לסיווג כאדם או בוט.

A close up of a map

Description automatically generatedתוצאות הניסויים:  
***תוצאות ניסוי מצב המודל לאורך האימון:***  
קודם, נראה השוואה בין שינוי ההפסד והדיוק של המודלים לאורך הזמן:

בנוסף לכך, נראה את מטריצות הבלבול ואת המטריקות הרלוונטיות על תוצאות הסיווג של סט המבחן במודלים הסופיים:

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

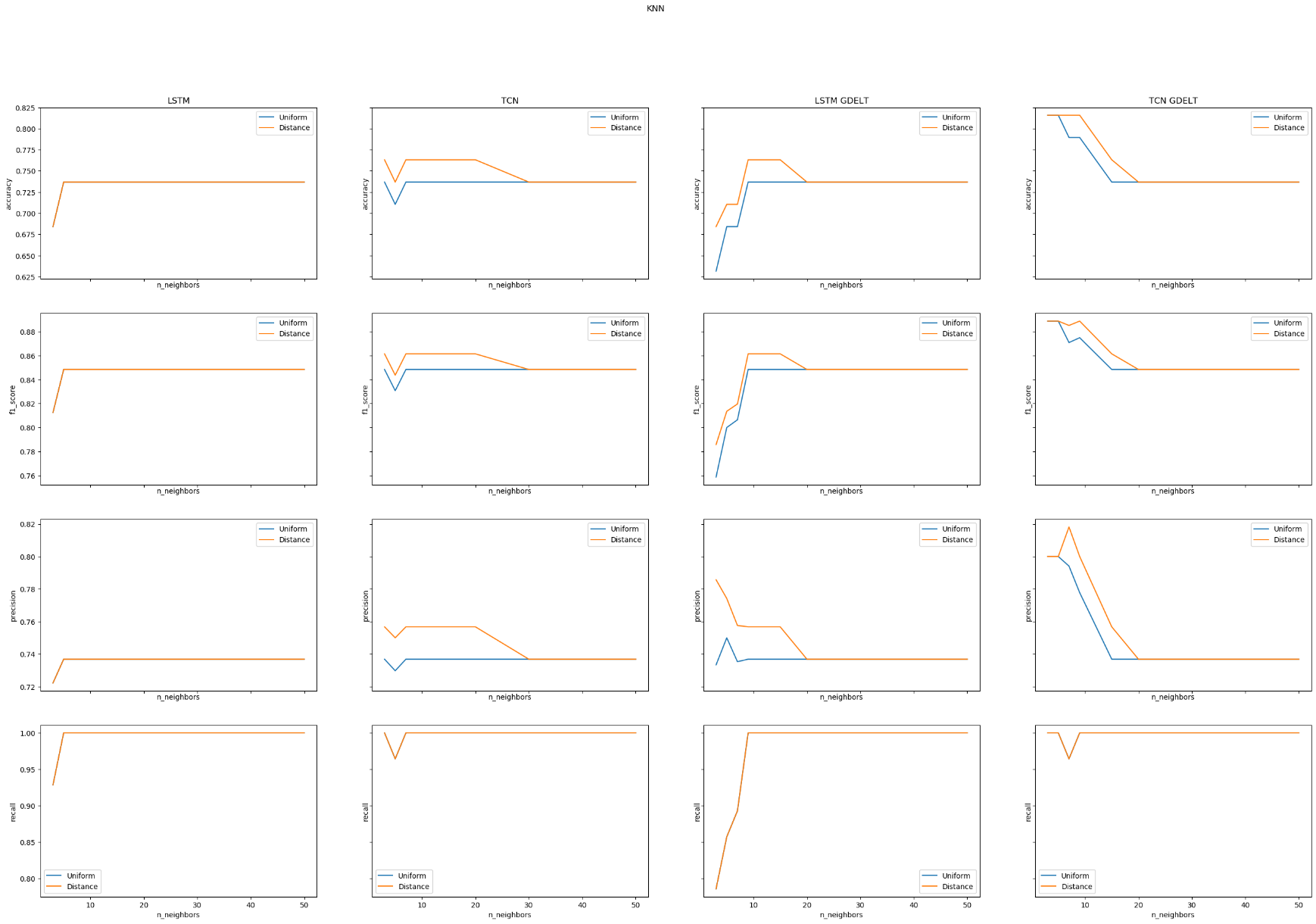
Description automatically generatedהמטריקות על תוצאות הסיווג:

ניתן לראות כי TCN נתן תוצאות קצת יותר טובות -LSTM בכל המקרים, וכי הוספת המידע החיצוני של GDELT גרמה להרבה יותר משתמשים להיות מסווגים כבוטים, מה שמצד אחד גרם לזה שאף בוט לא סווג כאדם אמיתי, אך גרם להרבה יותר טעויות סיווג על אנשים אמיתיים.

בסך הכלה הוספת GDELT גרמה לתוצאות גרועות יותר מבחינת הסיווג, ולכן ניתן לאמר **שבמודל שלנו** היא לא הייתה נחוצה בשביל בעיית הסיווג.  
יש סיכוי שהתכונה שיצרנו באמצעות המידע החיצוני לא טובה, אז אי אפשר להכליל את זה להוספת מידע חיצוני באופן כללי.

המודל המנצח מבחינת סיווגים היה ללא שאלה TCN ללא GDELT.

אם היה לנו יותר זמן, היינו מנסים להרחיב את מאגר המידע ליותר משתמשים, מכיוון שבסופו של יום השילוב של חוסר האיזון של מחלקות הסיווג ביחד עם גודל סט המבחן גרם לכך שיהיו מעט מאוד אנשים אמיתיים בסט המבחן.  
אז שינוי שהיה גורם לנו להרגיש יותר ביטחון בתוצאות ואולי גם מניב תוצאות יותר טובות הוא הגדלת כל מאגר הנתונים באופן שגם יהפוך אותו ליותר מאוזן מבחינת יחס הבוטים לאנשים אמיתיים.

***תוצאות ניסוי מסווגים קלאסיים:***  
**KNN:**

המידע שיוצר את הגרפים:

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generatedA screenshot of a social media post

Description automatically generatedA screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generatedבניגוד למסווגים הלינאריים, TCN עם GDELT הניב את התוצאות הכי טובות עבור שתי דרכי משקול הדגימות.

באופן כללי היו תוצאות טובות יותר עבור ערכי K קטנים, דבר שהגיוני בהינתן חוסר האיזון של מאגר הנתונים (עבור ערכים מספיק גדולים הכל מסווג כבוט).

התוצאה הזאת מרמזת על כך ש-TCN עם GDELT עושה את ההכללה הכי טוב מבחינת יצירת features, מכיוון ש-KNN נוטה להיות טוב עבור מרחבים שבהם קרבה של התכונות מייצגת דמיון אמיתי.

A close up of a map

Description automatically generated**SVM:**

המידע שיוצר את הגרפים:

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

גם במקרה הזה TCN עם GDELT הניב את התוצאות הכי טובות ברוב המטריקות.

בפרט גרעין מסוג Sigmoid הוא זה שהניב את התוצאות הכי חוץ ממטריקת ה-precision, שבה באופן מפתיע גרעין לינארי הניב בכל המקרים את התוצאות הכי טובות, וזאת למרות שבשאר המטריקות הגרעין הלינארי היה הכי גרוע.

מביצועי הגרעין הלינארי ניצן להסיק כי מרחבי התכונות שיצרנו אינם פרידים לינארית.

מאוד הגינוי שגרעין ה-Sigmoid עבד טוב יחסית, מכיוון שהוא שקול לרשת עמוקה עם 2 שכבות בעיקרון, אך מוזר שזה קרה רק עבור TCN עם GDELT.  
נשים לב שמעבר לשילוב הזה של גרעין ומודל, בכל שאר המקרים היו תוצאות יחסית דומות ל-Rbf ול-poly, וגם Sigmoid בד"כ הניב את אותן התוצאות או תוצאות קצת יותר גרועות.  
זה אומר שהם כנראה הגיעו להכללה דומה.

דבר נוסף שניתן לראות הוא שערכי C נמוכים אך לא קטנים מדי (סביבת ) מניבים את התוצאות הכי טובות, ושערכים גדולים בד"כ גרמו לתוצאות פחות טובות.

A close up of a map

Description automatically generated**עצי החלטה:**

המידע שיוצר את הגרפים:

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

באופן כללי אי אפשר לראות הבדלים מהותיים בביצועי עצי החלטה בין מרחבי התכונות שהמודלים השונים יצרו.  
התנהגות העצים ביחס למספר התכונות המקסימלי שהיה בהן שימוש גם לא היה עקבי במיוחד בין המרחבים השונים.

הדיוק המקסימלי שעץ הגיע עליו היה עם תכונות auto (שורש ריבועי של מספר התכונות) על המרחב ש-TCN ללא GDELT יצר, אבל גם זה היה רק דיוק של 78%, שאינו כל כך רחוק מ-73% (ערך עבור סיווג כולם כבוטים) ולכן קשה להעריך את זה.

לסיכום, עצי ההחלטה נתנו תוצאות שונות ומשונות שקשה להסיק מהן משהו, הדבר העיקרי שאפשר אולי להבין מהמקרה הוא שהמרחבים יחסית לא מוכללים מבחינת היכולת ליצור תחומים לפי הפרדתם עם ערך מסוים, כי אחרת היינו לפחות רואים איזושהי מגמה.

A close up of a map

Description automatically generated**Random Forest:**

המידע שיוצר את הגרפים:

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

במקרה הזה יצאו תוצאות שהיו די דומות לאלה של עצי החלטה מבחינת המגמות (די הגיוני בהינתן בזה שהמודל בנוי על עצי החלטה ומנסה לשפר את הביצועים של מספר עצים כאוסף),  
אך כן ניתן לראות מגמה אחת במיוחד – וזה כי מספר מסווגים בינוני (בין 10 ל-20) תמיד נתן תוצאות יחסית טובות (הדיוק הכי טוב שקיבלנו היה עם TCN ללא GDELT ו-10 עצים – 81.6%).

זה אומר כי במרחבים שיצרנו ניתן להרוויח מהשימוש במספר עצים שלוקחים תתי קבוצות של כלל הדגימות, מה שאומר שאולי יש המון רעש במרחבים שאנחנו יוצרים (מבחינת קרבת דגימות עם סיווגים שונים).

A close up of a map

Description automatically generated**AdaBoost:**

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedהמידע שיוצר את הגרפים:

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

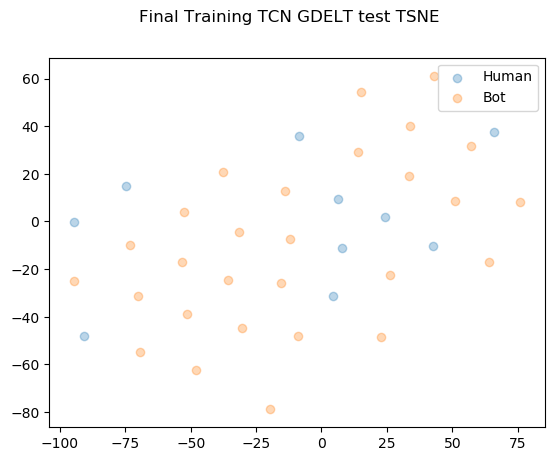
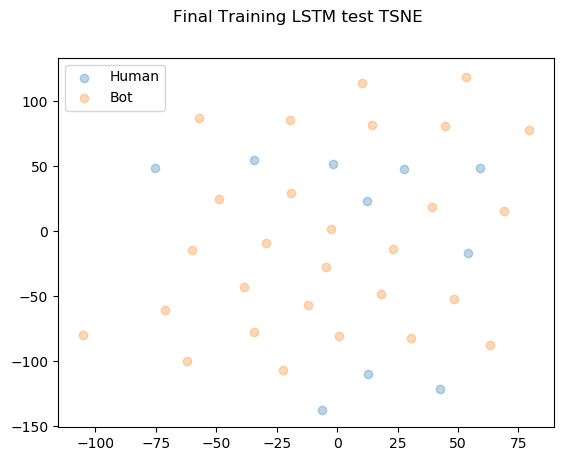
Description automatically generated

עבור AdaBoost בכל המקרים התוצאות הכי טובות היו במרחב שנוצר ע"י TCN עם GDELT, וניתן לראות כי אלגוריתם SAMME עבד הכי טוב כאשר השתמשנו ב-GDELT, וכי אלגוריתם SAMME.R עבד הכי טוב כשלא השתמשנו ב-GDELT.התוצאות הכי טובות היו 81.6% דיוק (עבור SAMME על מרחב של TCN ללא GDELT).

באופן כללי מספר בינוני (אזור ה-50) של מסווגים הניב את התוצאות הכי טובות, דבר שמרמז על overfitting במקרים האלו.

***תוצאות ניסוי בדיקת clustering:***  
תוצאות הבדיקה על סטי המבחן עם התכונות שארבעת המודלים שלנו הוציאו הן:

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

ניתן לראות שהתוצאות לא כאלה טובות - למרות שברוב המקרים קבוצות קטנות של אנשים אמיתיים מגובשות יחד באופן כללי אין הפרדה טובה לגושים.

**מסקנות סופיות מהניסויים:**  
קודם כל, כנראה היינו צריכים לעבוד על מאגר מידע יותר גדול ויותר מאוזן מבחינת הסיווגים.  
חוסר האיזון ביחד עם הגודל הקטן של סט המבחן גרמו לכך שהתוצאות שלנו היו הרבה יותר מפוקפקות ושהיה קשה למודלים להכליל בגלל שהכללות שסיווגו את רוב המשתמשים כבוטים הוציאו תוצאות שכביכול יותר טובות.  
זה היה בולט במיוחד בניסוי ה-clustering, בגלל שהרוב המוחלט של הדגימות היה בוטים היה קשה לראות אם יש גושים נפרדים של בוטים.

למרות ש-TCN ללא GDELT סיווג את הבעיה הכי טוב, הוספת GDELT נתנה את המרחב הכי טוב למסווגים הקלאסיים – דבר שמרמז על כך שהמודל ביצע את ההכללה הכי נכונה.  
יכול להיות שזה נובע מכך ש-TCN עם GDELT הוא המודל היחיד שלא נתקע על הסיווג של כל משתמשי סט המבחן כבוטים – הוא המשיך ללמוד והגיע לתוצאה הכי טובה שלו יחסית מאוחר באימון, בעוד שהשאר הגיעו לתוצאה הכי טובה שלהם מוקדם ואז נתקעו.

באף אחד מהמקרים לא עברנו את הדיוק של 81.6%, אך הצלחנו להגיע אליו אצל כל משפחות המסווגים – דבר שמרמז על הגעה להכללות דומות שאינן ממש טובות.

באופן כללי הראינו ש-TCN יכול לטפל בבעיה הזאת הכי טוב ביחס ל-LSTM, ושביחד עם השימוש במידע החיצוני הוביל להכללה הכי טוב.

אם היינו רוצים לשפר את התוצאות האלו, מעבר לשינוי במאגר הנתונים אולי היה כדאי שנשמור את המודל הכי טוב לפי ה-loss ולא לפי הדיוק – ה-loss המשיך לרדת במהלך כל האימון והמודל ששמרנו אותו הכי מאוחר נתן את התוצאות הכי טובות.  
כלומר אולי היה צריך לבדוק את זה לפי ה-loss ולהאריך את מספר ה-epochs באימון המודלים.

שיפור אפשרי נוסף שהיה יכול לעזור הוא הוספת מחלקות לסוגי הבוטים השונים – אולי ניסיון להבדיל בין בוטי פרסום, בוטים פוליטיים ועוד כמה סוגים היה פותר את בעיית חוסר האיזון של מאגר המידע, ועוזר להגיע להכללות יותר טובות.  
כמובן שזה היה משפיע על האופן שאנו מחשבים את המטריקות שלנו, אבל אנחנו מאמינים שזה לא יהיה יותר מדי להתחשב בסיווג שאינו בינארי.

למרות שהתוצאות שקיבלנו לא היו טובות כמו שרצינו, אנחנו עדיין שמחים שהצלחנו להוכיח את **היכולת** להשתמש בבינה מלאכותית בשביל סיווג בוטים ויצירת תכונות משתמשי twitter.

**נספח ניתוח והשוואת מספר פרמטרים בין LSTM ל-TCN:**  
בחלק הזה ננתח את מספר הפרמטרים שיהיה ל-LSTM ול-TCN בתוך המודל שלנו, כתלות במימדי הקידוד של וקטורי המילים ובמימדים של השכבות החבויות שבהן, בנוסף לגודל הקרנל ולמספר השכבות ב-TCN.

*נסמן:*

לפי אתר pytorch, מכיוון שה-LSTM מקבל קלט בגודל E ובעל מימד חבוי , מספר הפרמטרים הוא .

במקרה של TCN, מכיוון שהגדרנו שלכל השכבות יהיה מימד חבוי , והקלט של השכבה הראשונה הוא בגודל E, מספר הפרמטרים נקבע ע"י סכום הפרמטרים של שכבות הקונבולוציה.  
מספר הפרמטרים הוא: .

מכיוון שאורך הציוץ המקסימלי שנתקלנו בו (לאחר עיבוד מקדים) הוא 58, נאלצנו לבחור פרמטרים עבור ה-TCN שיחסו היסטוריה אפקטיבית של 58 (או חסם מלמעלה של הערך הזה).  
נוסחאת ההיסטוריה האפקטיבית ב-TCN היא: .  
לכן בחרנו את הפרמטרים כך שההיסטוריה האפקטיבית של המודל שלנו היא 91.

לכן, לאחר שגם נציב כי E=100, נקבל כי מספרי הפרמטרים במודל שלנו הם:

עבור LSTM, , ועבור TCN .

בכדי להתגבר על ההפרש המשמעותי במספר הפרמטרים, שבחרנו היה גדול מ-.

בחרנו ו-, כך שהתקיים כי מספר הפרמטרים עבור LSTM הוא 4,612,096 פרמטרים, ומספר הפרמטרים עבור TCN הוא 1,334,272.

יכולנו לבחור ב-LSTM מימד חבוי גדול יותר בגלל הפרשי המהירות בין המודלים, מה שאיפשר לנו לבחור עבורו מודל יותר סבוך.

מעבר לזה ראינו בעיות של underfit במהלך האימון, אז הגענו למסקנה שאנחנו עדיין לא בשלב שהוספת פרמטרים פוגעת באיכות המודל.

מסקנה: במקרה שלנו סביכות שני המודלים כתלות במימד החבוי פחות או יותר זהה.

**נספח תוצאות ושכפולן:**

בעיקרון כל התמונות שהוספנו שמורים בתיקייה graphs שבתוך הפרוייקט.

אנחנו ממליצים לעשות זאת במקרה של גרפי התוצאות, שגודלם גרם לכך שקשה לקרוא אותם בתוך הדו"ח עצמו.

לפני הרצת כל קוד אחר עליכם להריץ את הפקודות הבאות וש-anaconda יהיה מותקן אצלכם:

conda env create -f environment.yml

conda activate BotDetection

בכדי לשכפל את האימון נמליץ להריץ מאותה התיקייה את שורות הקוד:

python -m training.train\_classifer -n Final\_Training -c -e 60 --use\_SGD --plot\_results --compare\_gdelt -l 1e-5 -w 1e-3

python -m training.train\_classifer -n Final\_Training -c -e 60 --use\_SGD --plot\_results --compare\_gdelt -l 1e-5 -w 1e-3 -t --rec\_hidden\_dim 256 -r

או להריץ את המחברת jupyter בשם BotDetection.ipynb בתוך google colab ואז הורדת המודלים השמורים וגרף התוצאות משם (מומלץ).

שימו לב ששימוש בסקריפטים ייצור שני גרפים (אחד עבור LSTM ואחד עבור TCN), בעוד ששימוש במחברת ייצור גרף אחד עבור כולם.

אם ברצונכם להשתמש ב-random seed ששמור בתיקייה (שלפיהו חילקנו את סטי המבחן והאימון), תשתמשו בדגל -r גם בשורה הראשונה.

ניתן גם לשנות עוד מספר פרמטרים באימון, אך שאר הסקריפטים שלנו לא תומכים בשינויי מימדים ומבנה המודלים עצמם, או בשינוי בגודל סט המבחן.

בכדי ליצור את התכונות עם שימוש במודלים שאומנו עם הפקודות למעלה (או המודלים שאנחנו אימנו) ניתן להשתמש בפקודה:

python -m training.eval extract\_features

בכדי ליצור תמונות עבור תוצאות הסיווג הבינארי במודלים שלנו ובמודלים באחרים ניתן להשתמש בפקודה:

python -m training.eval evaluate\_classifiers

בכדי ליצור תוצאות עבור ה-tsne של סטי המבחן לאחר התכונות שהמודלים המאומנים יצרו, ניתן להשתמש בפקודה:

python -m training.eval plot\_tsne

**נספח קריאה נוספת:**

Wikidata – <https://www.wikidata.org/wiki/Wikidata:Main_Page>

GDELT – <https://www.gdeltproject.org/>

KNN – <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

SVM - <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

Decision Tree – <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

Random Forest – <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

AdaBoost – <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>

T-SNE - <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html>

1. https://arxiv.org/abs/1310.4546 [↑](#footnote-ref-1)
2. https://arxiv.org/abs/1803.01271 [↑](#footnote-ref-2)