**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**  
<>

**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.

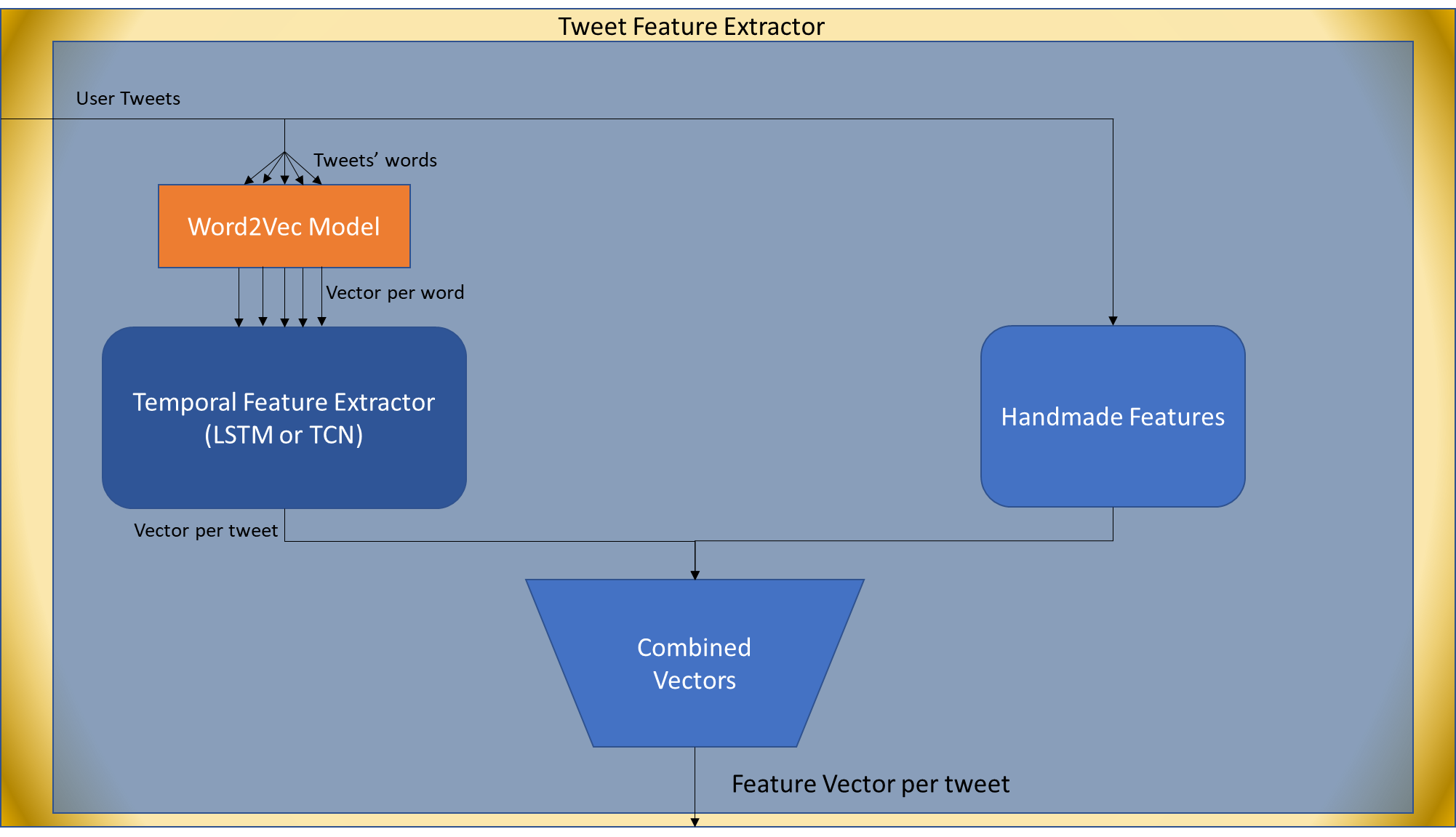
דבר שיוצר בעיות ביחד עם ההשפעה הזאת הוא קיומם של חשבונות מזויפים שעושים פוסטים באופן אוטומטי ומנסים להטות את דעת הקהל.  
מדברים קטנים כמו יצירת פרסום חינם למוצר עד לדברים מהותיים ביותר כמו השפעה על בחירות - הבוטים האלה גורמים לערעור היציבות של המרקם החברתי.

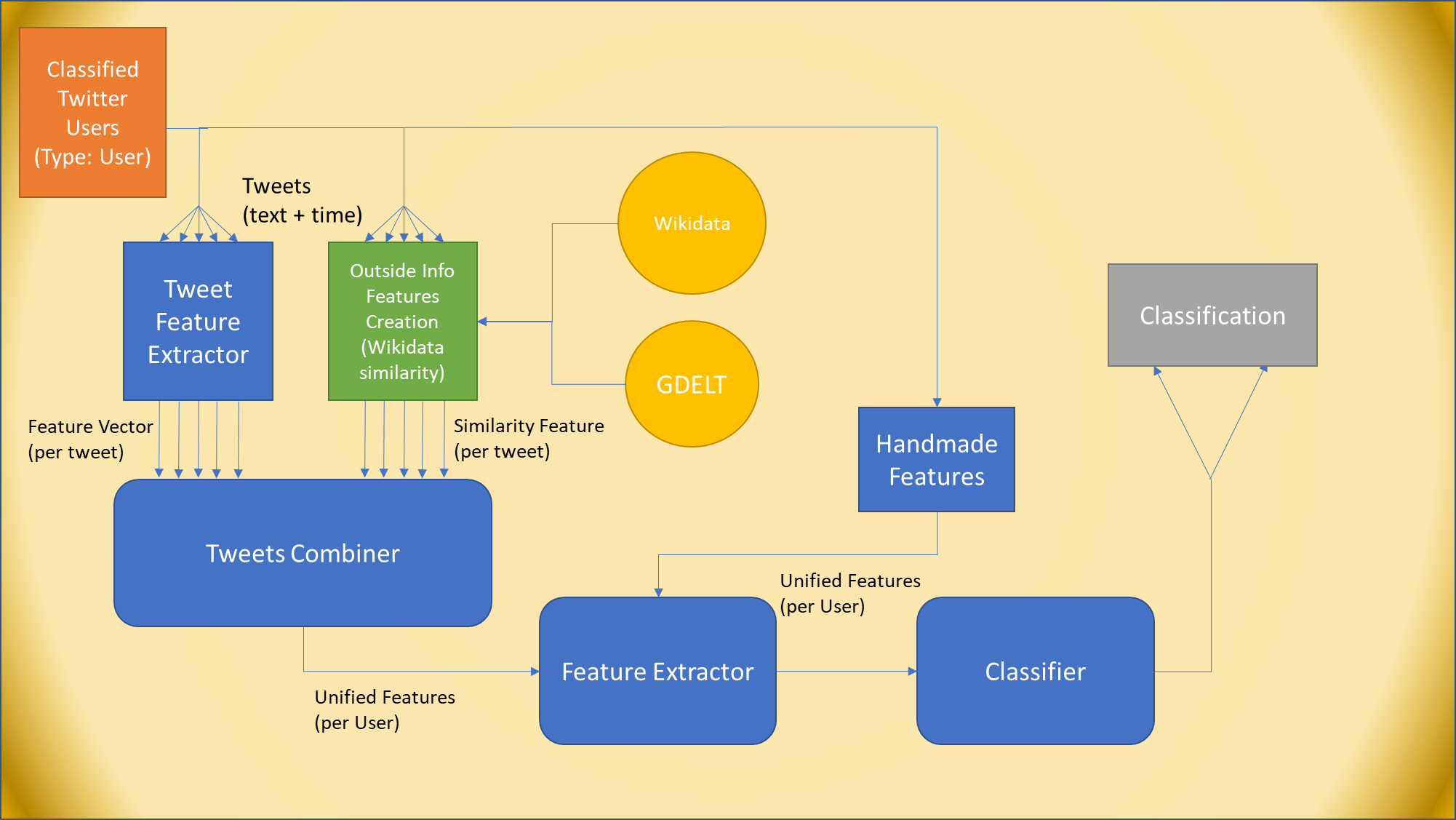
המטרה שלנו היא שבהינתן משתמש ב-Tweeter נרצה לסווג אם אותו המשתמש מופעל ע"י אדם אמיתי או ע"י בוט.

**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.

**תלויות (ספריות) ואופן התקנה:**  
<להכניס מהקובץ>

**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.

**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.
* נשים לב כי ל-TCN יש מספר פרמטרים גדול בהרבה מזה של LSTM, מכיוון ש-LSTM משתמש באותם הפרמטרים לכל החישובים הזמניים בעוד שב-TCN הפרמטרים נמצאים ברשת ולכן הם ממספר רב יותר.  
  בסוף הדו"ח ניתן למצוא נספח שבו נמצא ניתוח מתמטי של מספר הפרמטרים בשני המודלים – ובפרט במודל שלנו.
* 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 128 לכל ציוץ.  
אחרי זה אנו יוצרים מכל הציוצים של כל משתמש וקטור תכונות אחד גדול (בגודל 100\*128) 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 0.0001 ורגולריזצית L2 עם מקדם של 0.001 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 אלג' 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

המטריקות שלפיהן מדדנו את ביצועי ה-clustering שלנו הן:  
<הכנס מטריקות>

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

פרטי הניסויים:  
<ניסוי תוצאות אימון הרשת>  
<ניסוי בדיקת ביצועים של מסווגים קלאסיים>  
<ניסוי בדיקת clustering>

תוצאות הניסויים:  
<ניסוי תוצאות אימון הרשת>  
<ניסוי בדיקת ביצועים של מסווגים קלאסיים>  
<ניסוי בדיקת clustering>

**נספח ניתוח והשוואת מספר פרמטרים בין LSTM ל-TCN:**  
<הכנס את הניתוח מהדף שכתבתי>

**פרטים טכניים:**

אנחנו נכתוב את הפרויקט הזה בשפה **python**, ונשתמש במנהל החבילות **anaconda**.  
בכדי לשלוף מידע מטוויטר נשתמש בחבילה **tweepy** שמאפשרת לנו לעבוד עם הממשק של טוויטר ולהוריד ממנו את המידע הרלוונטי.  
נרצה להשתמש במימוש **Word2Vec** של **gensim** בכדי ליצור קידוד של מילים לוקטורים.  
נשתמש במאגר הנתונים **GDELT** בכדי להשיג הקשר לאירועים וב-**wikidata** וקטגוריות ויקיפדיה בכדי להשיג הקשר כללי.  
נשתמש ב-**scikit-learn** בשביל דברים כלליים של למידה ומסווגים פשוטים וב-**pytorch** בכדי לבנות מסווג מורכב יותר.

**סיכום:**  
<סיכום>

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