Report - Text Classification and Authorship Attribution Task

(Maxim Katz 322406604, Yuval Levi 325120384)

עיבוד מקדים לנתונים:

<u>ניקוי הנתונים:</u> בחלק זה ביצענו ניקוי לטקסט הטוויטרים, תהליך זה כלל מספר פעולות.

בתחילה, חילצנו פיצ'רים מהפוסטים שבטוויטר שישמשו אותנו בתהליך למידת המודל, נרצה שהמודל ילמד על אופי הטקסט בפוסט הטוויטר של טרמפ לעומת עוזריו. הפיצ'רים שחילצנו כוללים מספר הHashtags, מספר האזכורים@, מספר כתובות HTTP, מספר סימני השאלה?, מספר סימני הקריאה!, מספר אותיות גדולות בטקסט ואורך הפוסט בטוויטר. כמו כן, הוצאנו פיצ'רים על נתוני הזמן בשביל שהמודל ילמד על שעות הפעילות של טרמפ, נתונים אלו כללו - שנה, חודש, יום, שעה, דקות ושניות. עבור שורות שלא היה בהם timestamp עם הפורמט המתאים והיה קישור לטוויטר החלפנו בtimestamp שהתחבא בטקסט את מה שכתוב במקום הtimestamp בשורה הרלוונטית כך הצלחנו לשמור על כל השורות בטבלה ושכולם תקינות.

לאחר מכן, התעסקנו בטקסט של הפוסטים בטוויטר כך: הימרנו את הטקסט לאותיות קטנות משום שהוא מעודד עקביות ומונע כפילויות, בנוסף הסרנו תווים שאינם ASCII משום שתווים אלו אינם ניתנים לפירוש ועלולים לשבש את הניתוח למרות שחשבנו שיכול להיות תווים אלו דווקא כן יכולים לרמוז על מי כתב את הטוויטר אז מחקנו תווים שהם בטוח לא רלוונטיים על מנת לנתח את הטקסט עצמו ויכולים להבא כרעש למודל כמו ספרות, אזכורים (hashtags), כתובות אתרים URL, סימני פיסוק ו stopword. לבסוף, ביצענו Lemmatization Tokenization שהוא פיצול המשפט למילים והצגתם לפי שורשם, פעולות אלו מורידות כפילויות מה שיוביל לשיפור בביצועי המודל. כלל הפעולות עוזרות למודל להתמקד וללמוד את התוכן הטקסטואלי.

צעד הבא בניקוי היה המרת הטקסט לייצוג Embedding כאשר עבור כל מילה בטוויט מסוים העברנו אותה לייצוג Embedding, בעל ממד בגודל 300, על ידי שימוש במודל וקטוריזציה של ספריית spacy('en_core_web_lg'). spacy מספק וקטורי מילים מאומנים מראש הלוכדים משמעויות סמנטיות ויחסים בין מילים.

לאחר מכן, הורדנו עמודות לא רלוונטיות כמו 'tweet_id', 'user_handle', 'tweet_text', 'timestamp', 'device', 'corrected_timestamp', 'cleaned_tweet_text' כי אלו עמודות שמבלבלות את המודל ולא נותנות מידע חדש על הפוסט ומי שכתב אותו. כמו כן, הורדנו כפילויות וNAN ערכים חסרים לא רצויים שהמודל יקבל אותם.

לבסוף, עשינו סטנדרטיזציה לנתונים בטבלה. יש שתי סטנדרטיזציות מוכרות ביותר ששניהם היו כאופציונליות לכל אחד מהמודלים הראשונה MinMaxScaler והשנייה StandardScaler. השארנו אופציה כמו כן לא לעשות סטנדרטיזציה בכלל לנתונים למשל אם משתמשים בוקטוריזציה.

אם צריך גם עושים פיצול לנתונים כך שיהיו נתונים בtrain ובvalidation על מנת לבחון את ביצועי המודלים של כל אחד מהמודלים (20 אחוז לvalidation).

מודלים: בעבודה זו נעשה שימוש במספר מודלים, המודלים - SVM , Logistic Regression (ליניארי ולא ליניארי), XGBoost, Feed Forward NN, RNN-1.

Hyperparameters: ההיפר-פרמטרים נבחרו עבור כל מודל בעזרת שיטת Grid Search Cross Validation ונקבעו על ידי תוצאות המודל הטובות ביותר.

מדדים ותוצאות: הערכנו את התוצאות בעזרת מדד Accuracy.

Insights and Conclusions:

Data Preprocessing and Data Usage : עבור שלושת המודלים הראשונים הכוללים Logistic Regression, SVM linear and SVM non-linear ביצענו סטנדרטיזציה עם StandardScaler משום שהוא יצא הכי טוב בכל אחד מהמודלים הללו ולא ביצענו וקטורזציה של הטקסט ולא כללנו את הטקסט עצמו בטבלה שהעברנו למודלים הללו כי יש את הפיצ'רים שחושבו לפי מחיקת הטקסט ואפיינו את הטבלה בצורה טובה כדי לדעת מי כתב את הטקסט. עבור מודל FFNN הכנסנו לו גם את הוקטורזציה של הטקסט אחרי הניקוי של הטקסט כלומר Embedding של הטקסט בלי סטנדרטיזציה של הנתונים וגם האפיונים של הטקסט בעמודות השונות עם פיצול הTRAIN לTRAIN ול-VALIDATION. עבור מודל XGBOOST הכנסנו לו גם את הוקטורזציה של הטקסט אחרי הניקוי של הטקסט כלומר Embedding של הטקסט בלי סטנדרטיזציה של הנתונים וגם האפיונים של הטקסט בעמודות השונות. עבור מודל הRNN, הכנסנו לו בPreprocess רק את הוקטוריזציה של הטקסט אחרי הניקוי שלו כלומר הRNN לומד מהEmbedding של הטקסט בלי האפיונים המיוחדים שעשינו לטקסט.

Representation Data and Feature: חילוץ הפיצ'רים במשימה היו הפיצ'רים עסקו בהרגליי כתיבת וסגנון של טרמפ, לשם כל חילצנו פיצ'רים העוסקים בסגנון בתיבה (מעין meta data על הטוויט). בין היתר, היו אורך הטוויט ומספר סימני שאלה, סימני קריאה ועוד דברים על הטקסט שחילצנו שכבר הרחבנו לפי כן שהשתמשנו בהם בכל מודלי הML וגם במודל הFFNN. בנוסף, עבור טקסט בשפה האנגלית קיימים כלים משוכללים יותר כגון ייצוג המילים ב embedding באמצעות מודלי Embedding כמו glove שמאומנים מראש בהם השתמשנו על מנת לאמן את המודלים על הטקסט עצמו לאחר ניקוי הטקסט השתמשנו בEmbedding הנ"ל במודל FFNN,RNN ו-XGBoost.

תוצאות וההיפר-פרמטרים של האלגוריתמים:

	Logistic Regression	SVC Linear	SVC Non-Linear	FFNN	XGBoost	RNN
Pre-Processing	standartization='standard' vectorize=False split=False	standartization='standard' vectorize=False split=False	standartization='standard' vectorize=False split=False	standartization='None' vectorize=True split=True	standartization='None' vectorize=True split=False	features = only vectorized text split=True
Parameters	C=0.2 max_iter=10000	C=0.2 degree=1 max_iter=800	C=0.2 degree=1 kernel="sigmoid" max_iter=800	Eight hidden layers from input size to output size epochs=100 batch_size=64 optimizer="adam" learning_rate=1*e-5 loss=BCE loss	booster = 'gbtree' learning_rate = 0.1 n_estimators = 200	Two hidden layers of size 100 epochs=40 batch_size=64 sequence_length=10 learning_rate=1*e-4 dropout=0.5 loss=BCE with Logits loss
Accuracy	0.8231	0.8183	0.8044	0.7904	0.8599	0.7

Comparison Results: המודלים הקלאסיים הניבו את התוצאות הטובות ביותר, כמעט ברוב המקרים רשתות נוירונים ומודלי למידה עמוקה כמו BERT ו-RNN דורשות המון נתונים בשביל להגיע ללמידה מעמיקה של הטקסט או כל דבר אחר כמו וידיאו. בנוסף, בגלל כמות קטנה של דאטה במודלי DL יכולים להגיע OVERFIT על הנתונים ולכן צריך להכניס כל מיני טכניקות על מנת למנוע זאת כמו הורדת REGULARIZATION,EPOCHS, DROPOUT ועוד וככל הנראה זה גרם לתוצאות שהתקבלו. ההיפר פרמטרים נבחרו מ- GridSearchCV לכן זה ההיפר פרמטרים הכי טובים עבור מודלי הML עבור המשימה הנוכחית.

Significant Performance: מודל ה-XGBoost נתן את התוצאות הטובות ביותר עבור המשימה הזו, אחריו רגרסיה לוגיסטית, SVM ליניארי, SVM לא ליניארי, FFNN, ולבסוף RNN בסדר הזה היו גם ביצועיהם מהטוב ביותר לגרוע ביותר. זאת בגלל גודל הנתונים וההיפר פרמטרים ששמנו עבור כל אחד מהמודלים (הכי טוב עבור כל מודל בהתחשב לpreprocess שלנו). כמו כן, לא להכניס את מודלי הלמידה העמוקה לOVERFIT עבור הדאטה הקטן הזה.