תיעוד הפרויקט

מיקומים של קבצים:

- Datasets סופיים – "./code/ 3. Classifier/ datasets"
בתיקייה זו יש תיקיות עבור sms , email והפירוק המורפולוגי שלהם, ובתוך כל תיקייה יש קובץ עבור spam וקובץ עבור non_spam.
- בתוך התיקייה "./code / 4. analyze" יש את כל קבצי הפלט ממוספרים, שבמאמר אנחנו מכנים אותם "results files".

1. build dataset

להרצת הסקריפט:
להפעלה עבור ייצוא קובץ וואטסאפ:
[python clean_data.py [w | w2] [2]

להפעלה עבור ייצוא קובץ אימייל:
python clean_data.py [e | e2] 2

קובץ איסוף ההודעות המקורי נקרא _chat.txt , קובץ זה הוא ייצוא צ'אט מהוואטסאפ.
קובץ איסוף המיילים המקורי הוא מסוג mbox, קובץ זה הוא ייצוא מהג'ימייל בעזרת הקישור https://takeout.google.com/settings/takeout

<u>באופן כללי תהליך הרצת הסקריפט הוא בצורה כזו:</u>
1. קבצים עם השמות "email.txt","email2.txt","whatsapp.txt","whatsapp2.txt" שנמצאים בתיקייה original_data שהם קבצי הייצוא המקוריים מהוואטסאפ והמייל יהוו קלט לשלב הראשון של הסקריפט (כרגע נמצא שם רק קובץ וואטסאפ אחד של ספאם, כי אלה קבצים מקוריים ולא מצונזרים, קובץ mbox לא נמצא שם).
לאחר שמריצים את השלב הראשון של הסקריפט (ללא הפרמטר "2") עוברים לתיקייה cleaned_data, שם נמצאות ההודעות ללא סימנים מסביב, מופרדות על ידי מפרידים שיפורטו בהמשך באופן ספציפי.
2. נכנסים לתיקייה cleaned_data ושם מנקים את ההודעות ידנית באופן שיתואר בהמשך באופן ספציפי עבור קובץ וואטסאפ וקובץ mbox.
3. מעבירים את הקבצים לתיקייה manual_clean, מריצים את השלב השני של הסקריפט (עם הפרמטר "2") והקבצים הסופיים המוכנים למודל עוברים לתיקייה processed_data.

<u>להרצה על קובץ וואטסאפ:</u>
1. סקריפט שלב 1 - ניקוי הקובץ:
א. ניקיון הודעות המערכת של וואטסאפ, כמו "יצרת קבוצה זו" , "מחקת הודעה זו" , "כעת ההודעות הנשלחות בקבוצה זו מאובטחות עם הצפנה מקצה-לקצה." .
ב. ניקיון את פרטי המשתמש השולח - "[18.7.2019, 19:14:51] ,18.7.2019]" , "Guy Twito:" 23:01:18] יובל זיידל:" ודומים בעזרת regex.
ג. קו מפריד בין כל הודעה. בגלל שהרבה פעמים כמה הודעות הספאם נאספו להודעת וואטסאפ אחת, הצבנו קו מפריד שונה מהראשון במקומות שבהם יש שורה אחת ריקה, לצורך שלב ביניים של מעבר ידני לפני המעבר לשלב השני.
ד. המשתנה הגלובלי "use_separator2" מגדיר האם יש הודעות שנאספו להודעה אחת, אם הוא False לא יוצב המפריד וההודעות יופרדו ללא צורך בשלב הביניים.

2. שלב ביניים:
אם use_separator2=True יש לעבור באופן ידני על ההודעות בתיקייה cleaned_data ומחקנו את הקו המפריד בהודעות שנחצו ליותר מחלק אחד.
הודעות נחצו מכיוון שהייתה בהם שורה ריקה בודדת נוצר הקו מפריד הקטן שם, לכן מחקנו את המפריד הקטן ("++++++++++++") במקומות שבהם הוא לא מתאים.
בכל מקרה בלי קשר לערך של use_separator2 - את הקובץ שנשאר ב-cleaned_data יש לשים בתיקייה manual_clean.

3. סקריפט שלב 2 - עבודה על נתונים:
א. בכל מקום שיש בו קו מפריד כלשהו יש מעבר שורה במקום.
ב. הסרת הודעות כפולות.
ג. החלפת תווים מסויימים בקבועים (מעברי שורה , אמוג'ים-מכיל גם ספירה שלו או את האימוג'י עצמו) וכך ביטויים כמו אימיילים, לינקים, וטלפונים הוחלפו בקבועים (המכילים את אורכם במקרה של לינקים ואת סוגם במקרה של טלפונים),
בנוסף החלפנו את השמות המופיעים בהודעות המקוריות בשמות רנדומליים בעברית, שנלקחו מהאתר -https://www.itim.org.il/%D7%9E%D7%90%D7%92%D7%A8 /%D7%A9%D7%9E%D7%95%D7%AA
ד. ערבוב ההודעות על מנת שלא יהיה רצף של שיחה (דיסקרטיות).

להרצה על קובץ מיילים:

סקריפט שלב 1 - ניקוי הקובץ:

בייצוא מהמייל אין צורך בשלב הראשון של הסקריפט, מכיוון שהפונקציה שקוראת מקובץ מסוג mbox כבר דואגת לקרוא בכל מייל את הכותרת ואת תוכן המייל באופן נקי.

סקריפט שלב 2 - עבודה על נתונים:

1. העבודה על מיילים מאוד דומה לזו של הוואטסאפ, מלבד הצורך בקריאת הקובץ והפרדת הודעות בעזרת הקווים המפרידים מכיוון שההודעות כבר נמצאות במערך משלב קריאת הקובץ.
2. הסרת הודעות כפולות, החלפת תווים בקבועים, ערבוב הודעות (לא קריטי)
3. לא מבצעים את השלב הראשון של הסקריפט אבל עדיין צריך לבדוק את הדאטה שיצא כפי שעוברים על הדאטה של ההודעות.
לצורך כך הוספנו משתנה גלובלי "seperate_stage2_messages", שאם הוא True הוא מפריד בשורת רווח בין ההודעות לאחר השלב השני, כך ניתן לקרוא בצורה ברורה את ההודעות.

YAP.2

הכלי כתוב בשפת go, התיעוד שלהם מכיל את כל ההוראות להתקנה ולשימוש.
הקמנו שרת מינימלי על מנת להתקין את יאפ ולהריץ אותו בתור api (7.5 GB). מהר מאוד התגלה שצריך יותר מזה כדי להריץ את כל ההודעות, לכן הרמנו שרת של 30 GB.

*** עבור 1,2 אפשר פשוט להעתיק את הפקודות ב installation.txt.**
1. התקנת go (https://tecadmin.net/install-go-on-ubuntu/)
2. לעקוב אחרי ההוראות של yap .
3. להעתיק את הקבצים מתוך BASH FILES, ואת הdatasets ולתת להם הרשאות: chmod 777 FILENAME.
4. לעקוב אחרי ההסבר לגבי BASH FILES להרצה.

הוראות עבור bash scripts:

מתחברים לשרת בעזרת SSH של GCP.

על מנת להריץ את Yap:

1. start_yap/.
זה מפעיל את יאפ בתור api ברקע ואת הפלט שלו זורק ל home/guyy18/yapproj/trash/
2. נקרא את trash ונחכה שרשום בסוף Loaded model
3. עכשיו אפשר להריץ את הפקודה
& python3 seperate_chars.py
התוכנית לוקחת את כל הdatasets של המיילים והסמסים , מפרידה את הסימנים . , ! ? ומטפלת בסימנים " \ ( ) [ ] { } ככה שהיא יכולה להריץ את run_yap עבור כל משפט ולהכניס את כל הפלט לתיקיית output בקובץ מתאים.

*יש לשים לב בtrash אם יש memory error (זה מפיל את השירות yap ולא ניתן להשתמש בו).
במקרה כזה יש לקחת את קבצי הoutput , לבדוק מה הקובץ האחרון שיצא ומה המשפט האחרון בתוכו, אם יש קובץ שכל המשפטים בו עברו yap יש לקחת אותו ולרוקן אותו ב datasets (על מנת לא להפעיל עליו שוב את יאפ).
לאחר מכן יש להריץ את התהליך מההתחלה (מ start_yap) ואפילו כדאי לעשות קודם ריסטרט לשרת.

3. classifier

*יש תיעוד עבור הפונקציות בתוך הקוד.

להרצת הסקריפט:

[Run command - python go.py [0-5
different models = 1-5

מודלים אפשריים:

1 Naive Bayes classifier
2 Maximum Entropy Model
3 Memory Based Learning
4 Support Vector Machines
5 AdaBoost

אפשרויות לפרמטרים:

- test_driver.py - פרמטר print_wrong_decisions אומר אם להציג בפלט את ההודעות שסווגו באופן שגוי, נרצה אותו פעיל כדי שיכתוב לקבצי הפלט, לכבות אותו בזמן בדיקות הרצה.
- model.py - יש את models_names שמגדירים את שמות המודלים לפי סדר, ואת models שתואם לשמות ומגדיר את המודלים עצמם. מתחת בהערה יש עוד מודלים שאפשר לנסות להריץ.
- go.py:

send_whatsapp - האם לשלוח לנו הודעה בסיום, בהרצות אמת נרצה אותו פועל כדי לדעת להריץ עוד פעם כשמסתיימת הרצה, בבדיקות סתמיות אפשר לכבות.
k_fold_number - אותו k של cross validation. אם משתמשים בk=1 זה גם סבבה לבדיקה מהירה פשוט בלי cross validation.
include_parse_tree_feature - קובע אם לכלול את הפיצ'ר שבודק parse trees על המשפטים.
spam_max_length - הגבלה על כמות הודעות הספאם, נרצה את זה כדי לשנות יחס בין ספאם ללא ספאם.

non_spam_max_length - הגבלה על כמות הודעות הלא ספאם, נרצה את זה כדי לשנות יחס בין ספאם ללא ספאם, וגם כדי להגביל את כמות ההודעות (יש קובץ עם 60 אלף הודעות) .
block_all_models_if_k_big - קובע אם לאפשר להריץ cross validation על כל המודלים יחד או לא (5 מודלים * 4~6 דאטה סטים * k הרצות) – עלול ליצור בעיות זכרון. אם הוא True אז אפשר להריץ על מודל 0 (שזה כל המודלים) רק אם k=1 . עם שרת עם 30 ג'יגה ראם אפשר להשאיר את הפרמטר כבוי.

תיקיית whatsapp:

מכילה קובץ whatsapp.py שברגע שמסתיימת ריצה בשרת הוא שולח הודעה לוואטסאפ שהסתיימה הריצה ומצרף את הפרמטרים של הריצה. הוא עוזר לנו לדעת שאפשר לבצע עוד ריצה.
כרגע אין בסקריפט את המידע הדרוש של twilio משום שהוא אישי וצריך חשבון בtwilio.

Bash scripts

- x] aaa_python_run_model_live]- מריץ את מודל מספר 0-5) x) עם פלט למסך וגם לקובץ בתיקייה output עם תאריך ושעת הרצה. לא רץ אם מתנתקים מהשרת. בנוסף מבטל הרצות python קודמות (הורג תהליך).
- x] aaa_python_run_model]- כמו live רק שהריצה ברקע, כך שהוא רץ גם בעת התנתקות מהשרת ולא מציג פלט למסך (רק לקובץ).
- current_state - מציג את הפלט של הריצה האחרונה. נרצה להריץ אותו גם בזמן הרצת aaa_python_run_model (לא live) כדי לראות בקלות את הפלט של הריצה כרגע.
- .bashrc - להוסיף את השורה שיש שם לסוף ה .bashrc בשרת על מנת שהשרת ישלח הודעת וואטסאפ ברגע שהשרת דלוק ללא הרצה.

4. analyze

בתיקייה הזאת קבצי פלט.

קבצי הפלט מכילים מידע על:

- פרמטרי ריצה.
- זמן ריצה.
- מדדי הצלחה של כל אחד מהמודלים בכל איטרציה על K.
- סיכום מדדי ההצלחה לכל מודל לפי ממוצע ולפי המדד הכי נמוך מבין כל האיטרציות.
- משפטים שסווגו לא נכון, איזה קלאס קיבלו, איזה מודלים עשו את הטעות וכמה פעמים.

מדדי הצלחה:

- baseline
- positive margin
- recall
- precision
- F1 score
- confusion matrix
- WAcc
- TCR

הסבר לכל קובץ:

- מיילים וסמסים רגילים, מוגבל ל6000, שם קובץ - 1 [2019-09-11 02_42_30] output
- מיילים וסמסים רגילים, מוגבל ל3000, שם קובץ - 2 [2019-09-11 02_44_08] output
- סמסים רגילים, רק MultinominalNB, מוגבל ל10000 (נכשל על כל המודלים וגם נכשל בלי אדה בוסט), שם קובץ - 3 [2019-09-09 18_35_20] output
- מיילים מורפולוגים, מוגבל ל6000, בלי עצים או POS , שם קובץ -4 [2019-09-09 01_19_24] output
- סמסים מורפולוגים, מוגבל ל6000, בלי עצים או POS , שם קובץ - 5 [2019-09-09 18_30_46] output
- מיילים וסמסים מורפולוגים, מוגבל ל6000, בלי עצים, עם POS, שם קובץ - 6 [2019-09-10 17_53_04] output
- מיילים וסמסים מורפולוגים, מוגבל ל6000, עם עצים, בלי POS, שם קובץ - 7 [2019-09-11 18_42_05] output
- מיילים וסמסים מורפולוגים, מוגבל ל6000, עם עצים, עם POS, שם קובץ - 8 [2019-09-11 18_44_09] output
- שילוב מיילים וסמסים מורפולוגים, מוגבל ל3000 (6000 יחד), בלי עצים, בלי POS, שם קובץ - 9 [2019-09-14 22_34_43] output
- מיילים וסמסים מורפולוגים, מוגבל ל6000, עם עצים, בלי POS, מודל AdaBoost, שם קובץ - 10 [2019-09-16 18_08_58] output