עיבוד שפות טבעיות

תרגיל בית 2

שם: רנים אבראהים (212920896) , אסיל נחאס (212245096)

מבוא:

בתרגיל הזה אנחנו מנסים לאמן מודלים כדי לסווג משפטים לדוברים שונים דרך למידת מאפיינים שכל דובר מאפיינים אותו בשיטה שהוא מרכיב את המשפטים שלו.

שלב 1: הגדרת המחלקות

בשלב הראשון זיהינו את שני הדוברים בעלי מספר המשפטים הגדול ביותר בקורפוס הכנסת, ושייכנו אותם למחלקות "first" ו ."second"-כל שאר הדוברים הוגדרו במחלקה "אחר."

איתור הדוברים הדומיננטיים:

הפעלנו סקריפט שסורק את הקובץ ה `jsonl`-ומונה את כמות המשפטים לכל דובר.

מתוך הספירה, בחרנו את שני הדוברים המובילים:

ראובן ריבלין (הדובר בעל מספר המשפטים הגדול ביותר)

אברהם בורג (הדובר השני)

התחשבות בהערה על וריאציות בשם הדובר:

אחדנו שמות אפשריים עבור אותו דובר באמצעות מילון מיפוי (למשל" :ר' ריבלין" ,"רובי ריבלין ,"וכדומה – כולם שייכים לראובן ריבלין)

הסקריפט עובר על כל משפט ובודק האם שם הדובר מופיע במיפוי. אם כן, הוא משייך את המשפט למחלקה המתאימה (למשל 'first' לראובן ריבלין).

מחלקת "אחר":

כל שם דובר שלא זוהה כמופיע במיפוי של שני הדוברים הנבחרים הוגדר תחת המחלקה "אחר .(other) "כך אנו עונים למשימת ה multi-class classification-שבה:

מחלקה 1: הדובר עם מספר המשפטים הגבוה ביותר(first)

מחלקה 2: הדובר השני(second)

מחלקה 3: כל השאר (אחר)

תוצאות:

ספירת הדוברים העלתה כי ראובן ריבלין הוא הדובר שהופיע הכי הרבה בקובץ, ואברהם בורג שני לו במספר המשפטים.

לאחר המיפוי והאיחוד של כל וריאציות השם עבור שני הדוברים, כל משפט שמצאנו עבורם שובץ במחלקה הנכונה. השאר הוגדרו כמחלקת "אחר."

כך, עמדנו בדרישה להגדיר שתי מחלקות עיקריות (לצורך סיווג בינארי) ולהרחיבן למחלקה שלישית ("אחר") עבור הסיווג הרב-מחלקתי, תוך התמודדות עם שמות שונים של אותו דובר.

שלב 2: איזון המחלקות

במשימה זו בדקנו את מספר המשפטים בכל מחלקה לפני ביצוע ה:down-sampling-

other: 103173 משפטים

first: 3156 משפטים

second: 2448 משפטים

לאחר מכן, כדי לאזן את גודל המחלקות, ביצענו down-sampling רנדומלי למחלקות הגדולות כך שכל המחלקות יכילו כמות נתונים השווה לגודל הקטן ביותר:(2448)

other: 2448 משפטים

first: 2448 משפטים

second: 2448 משפטים

כעת שלושת המחלקות מאוזנות ומוכנות לשלבי הסיווג הבאים.

שלב 3: יצירת וקטור מאפיינים(Feature Vector)

וקטור BoW / TF-IDF
בהתאם להנחיות, היינו צריכים לבחור בין שימוש ב CountVectorizer-או ב-
TfidfVectorizer לשם המרת הטקסטים (משפטי הפרוטוקולים) לוקטור מאפיינים.

העדפת TfidfVectorizer :

הקצאת משקל למונחים נדירים :כאשר מסמך מכיל מילים נדירות (אך חשובות) ,
CountVectorizer מעניק להן משקל זהה למילים אחרות, בעוד ש TF-IDF-מעלה את משקלן היחסי ומאפשר למודל להבחין טוב יותר בין דוברים.

איפיון סמנטי מדויק יותר :בפרוטוקולים פרלמנטריים, עשוי להיות שימוש נרחב במילים כלליות כמו "חברי", "כנסת", "דיון TF-IDF ."מדכא מונחים נפוצים מדי ומבליט יותר מונחים ייחודיים.

גודל דאטה מצומצם :כאשר כמות המידע אינה עצומה TF-IDF ,תורם לייצוג "עשיר" יותר של הטקסט ומסייע להשיג ביצועים טובים.

פרמטרים נבחרים ב TfidfVectorizer:

:sublinear_tf=True מסייע להקטין את השפעתן של מילים נפוצות מאוד (Scaling ל-Frequencies).

:max_df=0.4 מסנן מילים המופיעות ביותר מ-40% מהמשפטים (למשל "אני" או "גם"), כיוון שהן פחות מועילות לסיווג.

וקטור מאפיינים מותאם אישית (סגנון/תוכן שאינו BoW):

מכיוון שאסור להשתמש בטקסט עצמו (BoW) בווקטור זה, הסתמכנו על מאפיינים סגנוניים שבאים לידי ביטוי במבנה המשפט:

אורך המשפט (מספר מילים)

מספר הפרוטוקול, סוג הפרוטוקול, מספר הכנסת

סימני פיסוק (שאלות, גרשיים, נקודות, מקפים ועוד)

ספירת פסיקים וכדומה

מטרתנו הייתה לייצר מאפייני סגנון שעשויים להבדיל בין דרכי ביטוי של דוברים שונים, בלי להסתמך ישירות על המילים.

באופן זה, עמדנו בדרישה ליצור שני סוגי וקטורי מאפיינים נפרדים:

וקטור טקסטואלי (TF-IDF) המבוסס על התוכן הלשוני

וקטור סגנוני (ללא BoW) המבוסס על תכונות מבנה/פיסוק/אורך המשפט

שלב 4: אימון המסווגים

מטרה :לסווג (1) וקטור BoW/TF-IDF ו-(2) וקטור סגנוני/מותאם אישית בעזרת שני מסווגים:

LogisticRegression

KNeighborsClassifier

Cross Validation: לשם הערכת הביצועים, השתמשנו ב.5-fold cross-validation-

בחירת פרמטרים:

Logistic Regression:

לכל וקטר בחרנו הפרמטרים שנותנים לנו את התוצאות הכי טובות ועשינו את זה בעזרת Grid Search (לא השארנו את ה-Grid Search בקוד, רק השתמשנו בו למצוא את הפרמטרים), הפרמטרים הכי טובים שמצאנו היו באופן הבא:

logistic regression in binary classification:

TF-IDF Vector:

Solver = liblinear, C = 10, random_state = 42, max_iter = 1000

Custom Vector:

Solver = liblinear, C = 0.1, random_state = 42, max_iter = 1000

תוצאות:

TF-IDF Vector report in binary classification using logistic regression:

Custom Vector report in binary classification using logistic regression:

```
mean accuracy: 0.8826
classification report for LogisticRegression:
              precision    recall  f1-score   support

       first       0.84      0.95      0.89      2448
      second       0.94      0.82      0.87      2448

    accuracy                           0.88      4896
   macro avg       0.89      0.88      0.88      4896
weighted avg       0.89      0.88      0.88      4896
```

מצאנו שבמקרה של ה TF-IDF ערכים גדולים של C עושים רגולריזציה טובה יותר מערכים קטנים כי וקטורים ה- TF-IDF בדרך כלל הם דלילים ולכן המודל יהיה לו ערך גדול של C יותר טוב (זה מפחית את עוצמת הרגולריזציה ככל שערך ה C גדל) כי דלילות הדאטה יכולה

```
mean accuracy: 0.8913
classification report for LogisticRegression:
              precision    recall  f1-score   support

       first       0.92      0.85      0.89      2448
      second       0.86      0.93      0.90      2448

    accuracy                           0.89      4896
   macro avg       0.89      0.89      0.89      4896
weighted avg       0.89      0.89      0.89      4896
```

למנוע באופן טבעי את ה – overfitting. אבל במקרה של הווקטור שאנחנו בנינו, ערך יותר קטן של C נתן תוצאה יותר טובה כי יותר רגולריזציה הייתה נדרשת כדי שהמודל יהיה יותר כללי.

KNN in binary classification:

TF-IDF Vector:

n_neighbors = 7, weights = distance, metric = Euclidean

Custom Vector:

n_neighbors = 11, weights = distance, metric = Manhattan

תוצאות:

TF-IDF Vector report in binary classification using KNN:

```
mean accuracy: 0.8037
classification report for KNN:
              precision    recall  f1-score   support

       first       0.81      0.79      0.80      2448
      second       0.80      0.82      0.81      2448

    accuracy                           0.80      4896
   macro avg       0.80      0.80      0.80      4896
weighted avg       0.80      0.80      0.80      4896
```

Custom Vector report in binary classification using KNN:

מצאנו שבשני המקרים, הפרמטר של המשקל נתן יותר משקל לשכנים הקרובים ביותר, לגבי מספר השכנים, במקרה של TF-IDF Vector יש לנו יותר ממדים בווקטור ולכן מספר בינוני כמו 7 מאפשר למודל למוד את הדאטה בלי להיות מושפע מאוד מהרעש, וגם מאזן ה tradeoff בין ה bias לבין ה variance. אך בווקטור שבנינו כאשר יש לנו פחות ממדים,

```
mean accuracy: 0.9109
classification report for KNN:
              precision    recall  f1-score   support

       first       0.89      0.94      0.91      2448
      second       0.94      0.88      0.91      2448

    accuracy                           0.91      4896
   macro avg       0.91      0.91      0.91      4896
weighted avg       0.91      0.91      0.91      4896
```

מספר יותר גדול כמו 11 עוזר יותר כי הוא מאפשר למודל להיות כללי יותר וגם כאשר נקודות הדאטה רחוקות אחת מהשנייה. בחרנו את המטריקות מאותן סיבות כמו הממדים השונים והדלילות של הדאטה.

בקלסיפיקציה מרבית בחרנו אותם ערכים כי שום דבר לא השתנה עם הוספת מחלקה חדשה חוץ ממספר הדאטה והמיקום שלהם אבל האנלוגיה לבחירת הפרמטרים נשארה כמו שהיא.

תוצאות הקלסיפיקציה המרבית:

TF-IDF Vector report in multiclass classification using logistic regression:

```
mean accuracy: 0.7432
classification report for LogisticRegression:
              precision    recall  f1-score   support

       first       0.71      0.66      0.68      2448
       other       0.65      0.78      0.71      2448
      second       0.91      0.80      0.85      2448

    accuracy                           0.74      7344
   macro avg       0.75      0.74      0.75      7344
weighted avg       0.75      0.74      0.75      7344
```

Custom Vector report in multiclass classification using logistic regression:

TF-IDF Vector report in binary classification using KNN:

```
mean accuracy: 0.6193
classification report for LogisticRegression:
              precision    recall  f1-score   support

       first       0.54      0.63      0.58      2448
       other       0.75      0.45      0.56      2448
      second       0.63      0.78      0.70      2448

    accuracy                           0.62      7344
   macro avg       0.64      0.62      0.61      7344
weighted avg       0.64      0.62      0.61      7344
```

```
mean accuracy: 0.6595
classification report for KNN:
              precision    recall  f1-score   support

       first       0.58      0.60      0.59      2448
       other       0.60      0.63      0.61      2448
      second       0.82      0.74      0.78      2448

    accuracy                           0.66      7344
   macro avg       0.67      0.66      0.66      7344
weighted avg       0.67      0.66      0.66      7344
```

Custom Vector report in binary classification using KNN:

```
mean accuracy: 0.7278
classification report for KNN:
              precision    recall  f1-score   support

       first       0.71      0.84      0.77      2448
       other       0.67      0.69      0.68      2448
      second       0.84      0.66      0.74      2448

    accuracy                           0.73      7344
   macro avg       0.74      0.73      0.73      7344
weighted avg       0.74      0.73      0.73      7344
```

ניתוח:

סיווג בינארי: גם KNN וגם רגרסיה לוגיסטית עשו ביצועים טובים, כאשר KNN משתמש בווקטורים מותאמים אישית שהשיגו את הדיוק הגבוה ביותר.

סיווג רב-מחלקות: רגרסיה לוגיסטית עם וקטורים TF-IDF הציג את הביצועים הטובים ביותר, בעוד KNN עם וקטורים מותאמים אישית הראו גם ביצועים טובים.

תוצאות אלו מדגישות את החשיבות של בחירת השילוב הנכון של תכונות ומסווגים בהתבסס על בעיית הסיווג הספציפית. הייצוג הווקטורי של TF-IDF עבד טוב יותר עם

רגרסיה לוגיסטית עבור סיווג רב-מחלקות, בעוד וקטורים מותאמים אישית היו יעילים יותר עם KNN לסיווג בינארי.

שלב 5: סיווג המשפטים מקובץ knesset_sentences.txt

לאחר שהשווינו בין התוצאות השונות משני המסווגים ושני הווקטורים, בחרנו להשתמש ב- Logistic Regression שהתבסס על TF-IDF כמודל הסופי, כיוון שהוא הפגין את הביצועים הטובים ביותר.
הפעלנו את המודל הזה על קובץ knesset_sentences.txt, כך שלכל משפט נוצר וקטור TF-IDF, והמודל ביצע חיזוי באחת משלוש המחלקות:

first: הדובר שהופיע הכי הרבה

Second: הדובר השני מבחינת כמות משפטים

Other: כל שאר הדוברים

עם זאת, בתוצאות בפועל ראינו שכמעט כל המשפטים בקובץ knesset_sentences.txt סווגו למחלקת "other". חוץ משני משפטים, אחת הסיבות האפשריות לכך היא שמחלקת "other" מכילה מגוון סגנונות וטקסטים רחב במיוחד, כך שהמודל מתקשה למצוא מאפיינים מובהקים כדי לשייך משפטים למחלקות "first" או "second"

לבסוף, רשמנו את הסיווגים עבור כל משפט בקובץ classification_results.txt, כך שבכל שורה מופיעה התווית "second", "first" או "other", בהתאם לחיזוי המודל.

שאלות:

שאלה 1:

קושי בהגדרת שייכות :לעיתים קרובות, המחלקה "אחר" הופכת ל"סל אשפה" עבור דוגמאות שלא מתאימות למחלקות הראשיות, מה שעלול ליצור חוסר עקביות בשיוך דוגמאות.

חוסר אחידות בתוך "אחר :"הסיווג "אחר" עשוי להכיל קבוצה מגוונת מאוד של משפטים (או דוברים), בלי מאפיינים סגנוניים ברורים המשותפים לכולם, דבר המקשה על המודל לזהות מאפיינים מובהקים.

שאלה 2:

מדד רצוי :במקרה כזה נרצה למקסם קודם כול את ה *Recall*-(רגישות) של הדובר הראשון, כי חשוב לא לפספס אף דוגמה שלו (כל פספוס = קנס גבוה). לחילופין, אפשר גם לשקול

Precision גבוה אם העלות של זיהוי-יתר (False Positive) אינה בעייתית. אבל באופן קלאסי, נעדיף מקסימום *Recall* לדובר הראשון.

בהתבסס על ה Recall-עבור הדובר הראשון, המודל המתאים ביותר למטרה זו הוא Logistic Regression עם .TF-IDF Vector מודל זה השיג Recall של 0.95 עבור הדובר הראשון, מה שמבטיח כי 95% מהדוגמאות של הדובר הראשון זוהו נכון. זהו המודל עם ה Recall-הגבוה ביותר עבור הדובר הראשון, ולכן הוא המודל המומלץ למטרה זו.

שאלה 3:

כעת אנו רוצים לא לפספס *שום* דוגמה של שני הדוברים הראשיים. המשמעות היא שהמודל צריך להיות *מאוד זהיר* (גבוה בכל הרגישויות) – כלומר Recall גבוה גם לדובר הראשון וגם לדובר השני.

למעשה, אנו רוצים ש-*כל* הסיווגים של שני הדוברים יהיו נכונים, ולכן נעדיף מודל שמראה ביצועים מאוזנים וגבוהים בשניהם) למשל F1 ,גבוה לשניהם, או Recall גבוה לשניהם.(

אם נאלץ לבחור מדד אחד – כנראה שנבקש מקסימום *Accuracy*(כי אפילו טעות אחת היא בעייתית) או שנחפש *minimize error rate* מוחלט. בכל מקרה, הדגש פה הוא אפס טעויות על שני הדוברים.

שאלה 4:

חלוקה פשוטה:(Train/Test)

יתרון: קלה ליישום ומאפשרת לבדוק ביצועים על סט שלא נראה באימון.

חיסרון: התוצאות תלויות מאוד באופן חלוקת המדגם (עשוי להיות לא יציג). ייתכן שהסט הקטן (הבדיקה) לא מייצג היטב את המגוון הכולל.

Cross Validation:

יתרון: מנצל טוב יותר את כלל הנתונים (כל דוגמה נמצאת גם באימון וגם בבדיקה במעברים שונים). מספק הערכה יציבה יותר של הביצועים.

חיסרון: חישוב יותר יקר (דורש לאמן מודל מספר פעמים).

אמינות :לרוב Cross Validation ,נחשבת לאמינה יותר) בייחוד עם k-fold גדול (כי התוצאה איננה תלויה בחלוקה אחת מקרית ל.Train/Test-

שאלה 5:

KNN:

יתרונות: לא דורש הנחות על צורת הנתונים, קל ליישם, יכול לקלוט דוגמאות חדשות בקלות.

חסרונות: איטי בניבוי (דורש חיפוש שכנים), רגיש לרעש, מצריך הגדרת פרמטר k ומטריקה מתאימה.

Logistic Regression:

יתרונות: מהיר בחיזוי, מודל ליניארי יציב, נותן הסתברויות של הסיווג, קל לפרש (משקלות לכל פיצ'ר).

חסרונות: לרוב מניח יחסים ליניאריים; עלול לא להתאים לבעיות מורכבות לא-ליניאריות.

בתרגיל שלנו (סיווג משפטים), לרוב Logistic Regression נותן תוצאות טובות ומהירות יותר, אבל הכול תלוי בנתונים: אם המבנה מורכב ואנו יכולים לייצג היטב את השכנויות, ייתכן ש KNN-יעבוד היטב. בפועל Logistic Regression ,נוטה להיות עדיף עבור בעיות טקסט גדולות.

שאלה 6:

יתרונות:

ייתכן יותר הקשר טקסטואלי/סגנוני: כאשר מאחדים משפטים לאוגד (Batch), קל יותר למודל לזהות תבניות מורחבות בדיבור/סגנון.

עשוי להפחית רעשים: אם משפט אחד קצר או גנרי, צרוף של מספר משפטים מאותו דובר עשוי לתת תמונה ברורה יותר על הדובר.

חסרונות:

פחות דוגמאות בסך הכול (כל דוגמה "כבדה" יותר) – המודל יכול לאבד גמישות ולסבול ממדגם מצומצם.

לא תמיד כל המשפטים קשורים בדיוק לאותו נושא/סגנון, כך שהאיחוד יכול לערבב סגנונות שונים של הדובר עצמו.

שאלה 7:

אין תשובה מוחלטת, זה תלוי בסוג הטקסט וגודל הדאטה:

יחידה קטנה (1–2 משפטים) :יותר דוגמאות, המודל לומד דקויות על כל משפט, אך עלול לפספס הקשר רחב.

יחידה בינונית (5–10 משפטים) :איזון – המודל מקבל מספיק הקשר בלי לאבד יותר מדי דוגמאות.

יחידה גדולה (100 משפטים) :מעט דוגמאות, אבל המון מידע בכל דוגמה. עשוי להיות מועיל רק אם יש דאטה מספקת, ומובחן סגנונית.

בתרגיל הנוכחי, כנראה 5–10 משפטים ליחידה היו עשויים לשפר הקשר, אך לא להקטין יתר על המידה את מספר הדוגמאות. זאת לעומת 1 משפט שפשוט מקטין הקשר, או 100 משפטים שקשה לאסוף בצורה אחידה.