Amit Sindani 206902694, Raz Olewsky 315341396

Natural Language Processing – The Final Project

חלק א' - אלגוריתם ה-Clustering:

א) האלגוריתם שבו עשינו שימוש לצורך ביצוע ה-clustering הוא האלגוריתם שצורף בהנחיות הפרויקט.

תחילה, נפרט על מבני הנתונים שבהם השתמשנו במימוש האלגוריתם.

1) **dataset** – רשימת הבקשות. כל בקשה מאופיינת ע"י רשימה בעלת 3 איברים:

[request text (string), BERT embedding vector (numpy array), cluster id the request belongs (-1 if not belongs to any cluster)]

2) **clusters** – מילון המכיל את כלל הקלאסטרים:

Key: cluster id, Value: details of specific cluster (sum of vectors, list of indexes of requests)

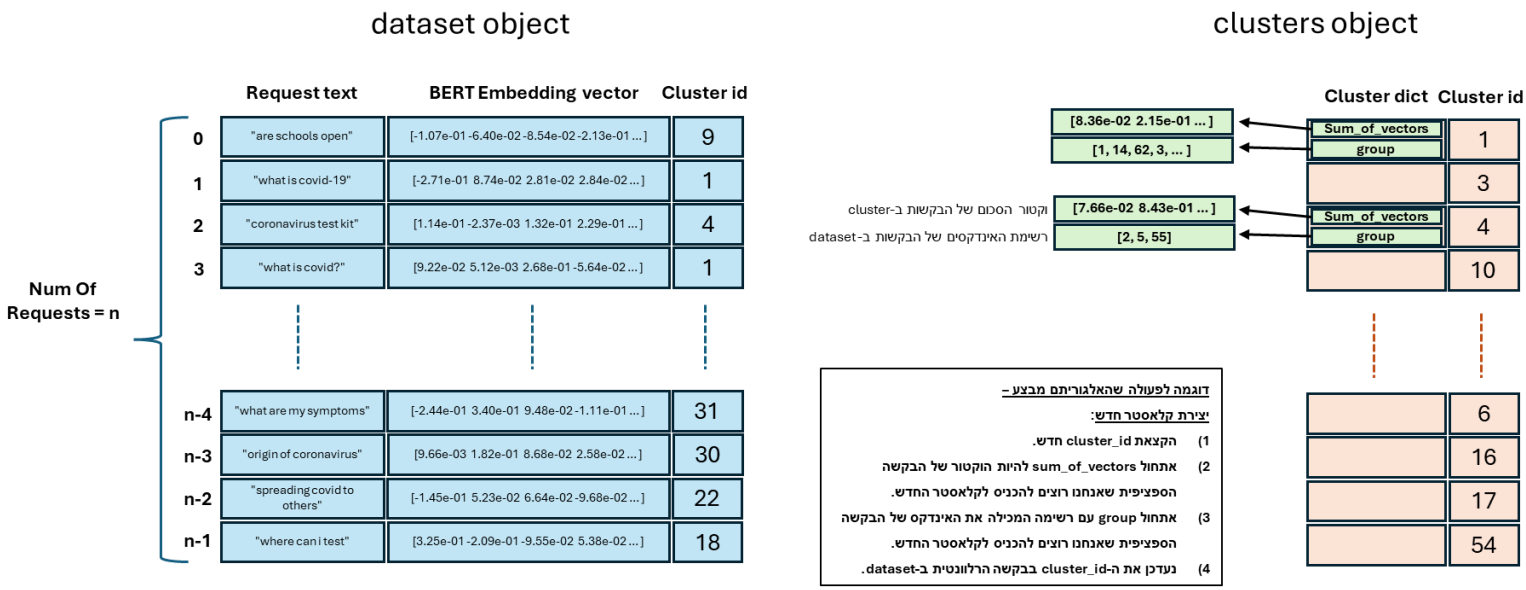
כל קלאסטר הינו מילון של 2 item-ים – sum_of_vectors ו-group:

- sum_of_vectors - מייצג את וקטור הסכום של כלל הבקשות השייכות לאותו הקלאסטר (הוא ישמש אותנו לחישוב הצנטרואיד בכל שלב).
- group - מייצג רשימה של אינדקסים מה-dataset של כלל הבקשות השייכות לאותו הקלאסטר.

החלטנו לבחור במילון כמבנה הנתונים לשמירת הקלאסטרים כיוון שאם היינו בוחרים מבנה נתונים אחר כמו רשימה (וזה מה שעשינו בהתחלה) והאלגוריתם היה מבצע מחיקות לקלאסטרים מסוימים (לדוגמה כאשר הוא מעביר בקשה מקלאסטר המכיל בקשה אחת בלבד לקלאסטר אחר) אז לבקשות מסוימות ב-dataset היינו צריכים לשנות את ה-cluster id כי ברשימה של קלאסטרים ה-cluster id הוא האינדקס של הקלאסטר ברשימה, ובמחיקת הקלאסטר ישנם קלאסטרים שמשנים את האינדקס שלהם. בגדול – שימוש ברשימה במקום מילון עלולה לדרוש עבודה רבה לעדכון הבקשות והיה יוצר קוד מסורבל ופחות מובן.

cluster id הינו מספר רץ, וגדל ב-1 כאשר נוצר קלאסטר חדש. נשים לב שכאשר נמחוק קלאסטר, לא נחזור על cluster id שכבר השתמשנו – הדבר לא משנה העיקר שיהיה מזהה ייחודי שידע להפריד בין קלאסטרים שונים.

להלן תרשים לדוגמה הממחיש את מבנה הנתונים שתיארנו לעיל:



ב) <u>חישוב קרבה בין בקשה לצנטרואיד של כל קלאסטר:</u>

אלגוריתם זה יכניס או לא יכניס בקשה לקלאסטר מסוים בהתחשב ל"קרבה" בין אותה הבקשה לצנטרואיד המוגדר ע"י אותו הקלאסטר.

ישנם סוגי קרבות שונים. נפרט על התהליך שביצענו לבחירת סוג הקרבה שבו עשינו שימוש במימוש האלגוריתם.

1) בהתחלה היה לנו קל לחשוב על קרבה מסוג מרחק אוקלידי.
נכניס לקלאסטר מסוים את הבקשה אם ורק אם המרחק האוקלידי בין הבקשה לצנטרואיד של אותו הקלאסטר קטן מרף (threshold) מסוים <u>וגם</u> המרחק האוקלידי קטן מכל מרחק אוקלידי בין הבקשה לשאר הצנטרואידים של הקלאסטרים הקיימים כרגע.

שימוש במרחק אוקלידי הביא פתרון סביר אך לא מספק.

להלן התוצאות שקיבלנו בשימוש של קרבה מסוג מרחק אוקלידי (optimal threshold = 0.8):

Covid19 requests:

```
clusters in 1st and 2nd solution: 36 and 24
unclustered requests in 1st and 2nd solution: 590 and 595
rand score: 0.9100369433742369
adjusted rand score: 0.7327652167996643
```

Banking requests:

```
clusters in 1st and 2nd solution: 65 and 46
unclustered requests in 1st and 2nd solution: 262 and 210
rand score: 0.9711133921880108
adjusted rand score: 0.5733430158500896
```

2) כדי לשפר את התוצאות ולאחר קריאת המאמר - https://medium.com/@ahmedmellit/text-similarity-implementation-using-bert-embedding-in-python-1efdb5194e65

החלטנו לעבור לבדיקת קרבה מסוג cosine similarity.

לחישוב הקרבה עשינו שימוש בפונקציית ספרייה בשם cosine_similarity השייכת למודול sklearn.metrics.pairwise.

הפונקציה הנ"ל מחשבת את ה-cosine similarity על סמך הנוסחה הבאה :

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

לאחר שימוש בסוג קרבה זה, כמות הקלאסטרים גדלה והתקרבה לכמות הנדרשת וגם המדדים של ה-RI וה-ARI עלו (optimal threshold = 0.675):

Covid19 requests:

```
clusters in 1st and 2nd solution: 36 and 29
unclustered requests in 1st and 2nd solution: 590 and 598
rand score: 0.9437679978727858
adjusted rand score: 0.83152873192005
```

Banking requests:

```
clusters in 1st and 2nd solution: 65 and 53
unclustered requests in 1st and 2nd solution: 262 and 265
rand score: 0.9764310020052257
adjusted rand score: 0.6441006981495596
```

3) התוצאות שקיבלנו עדיין לא מספקות.

נרצה למצוא סוג קרבה אחר או לשנות סוג קרבה קיים שיביא לשיפור התוצאות.

לאחר ניתוח חישוב הקרבה cosine similarity גילינו שהחישוב שעשינו אינו מיטבי.

המכנה בנוסחה של ה-cosine similarity הוא מכפלת אורכי הווקטורים. חישוב כל אורך וקטור כחלק מהנוסחה מהווה בעיה – בגלל שהווקטורים ש-BERT ייצר כבר מנורמלים אז האורכים של הווקטורים הם כבר 1 – כשנחשב אותם בתוך הפונקציה נקבל מספרים שהם קירוב ל-1.

```
BERT vector length: 0.99999994
BERT vector length: 1.0
BERT vector length: 1.0000001
```

הערכים הלא מדויקים משפיעים על טיב התוצאה. לכן החלטנו לחשב את המרחק רק על סמך המונה (המכפלה הפנימית) בין 2 הווקטורים בהנחה שהמכנה הוא 1 ואינו רלוונטי. לאחר שינוי זה – קיבלנו תוצאות טובות הרבה יותר (optimal threshold = 0.6):

Covid19 requests:

```
clusters in 1st and 2nd solution: 36 and 35
unclustered requests in 1st and 2nd solution: 590 and 596
rand score: 0.9791521693624581
adjusted rand score: 0.9370548613158843
```

Banking requests:

```
clusters in 1st and 2nd solution: 65 and 65
unclustered requests in 1st and 2nd solution: 262 and 252
rand score: 0.9915631501430857
adjusted rand score: 0.8586110604054348
```

`

חלק ב' - Cluster Naming/Labeling:

בחלק זה נפרט על כיווני חשיבה שונים ליצירת לייבלים עבור הקלאסטרים ועל מספר דרכים שמימשנו עד הדרך שבחרנו.

לפני שהתחלנו להציע רעיונות סימנו מס' עקרונות מנחים שאנו מצפים מהלייבלים שניתן:

- לייבלים קצרים (2-6 מילים)
- לייבלים נכונים תחבירית
- לייבלים אינפורמטיביים ותמציתיים

דרך א':

המחשבה הראשונה שעברה לנו בראש הוא להיעזר בצנטרואיד של כל קלאסטר. וזאת מהסיבה כי הצנטרואיד של קלאסטר הוא ייצוג שמתייחס לכלל הבקשות בקלאסטר ודומה להן (מבחינת "קרבה"). הדרך הנאיבית ביותר לשימוש בצנטרואיד הייתה לחפש את הבקשה הקרובה ביותר לצנטרואיד ולהשתמש בה כ-label. לשיטה זו מספר חסרונות: ה-label יכול להיות ארוך מאוד (בהתאם לאורך הבקשה), ה-label עלול להיות לא תמציתי ולהכיל פרטים רבים שאינם רלוונטיים.

דרך ב':

דרך נוספת שחשבנו עליה, הוא להשתמש במודלי שפה קיימים כדי לייצר label חדש על סמך הצנטרואיד. כלומר בהינתן הצנטרואיד נפיק label – טקסט המבוסס על הצנטרואיד. חיפשנו שיטות לעשות את זה, לחפש קופסאות שחורות (מודלי שפה מוכנים - מפענחים) שיעשו לנו את העבודה – אך לא מצאנו מודל מספק לצרכינו.

דרך ג':

הדרך הבאה היא שימוש ב-ngrams.

בחרנו ללכת בכיוון זה לאחר התבוננות בקבצי ה-output שאלה צירפה לפרויקט וזיהינו כי הלייבלים הם בעצם ngrams שקיימים בתוך אותו הקלאסטרים.

1) הדרך הפשוטה להפקת לייבל עבור קלאסטר ע"י שימוש ב-ngrams הוא לעבור על ngrams בחלונות מסוימים שנגדיר מראש (בין 2 ל-6 מילים שזהו אורך label סביר) מהבקשות של אותו הקלאסטר ולספור את הכמויות של כל אחד ואחד מהם. לבסוף ניקח את ה-ngram השכיח ביותר להיות הלייבל. שיטה זו פותרת את הבעיה של לייבל ארוך. אך בעיה חדשה צצה – השיטה תעדיף ngrams בגדלי חלונות קטנים (2-3) על פני ngrams ארוכים יותר (5-6) מהסיבה שלרוב ה-ngrams בעלי החלונות הקטנים נספרים יותר מה-ngrams בעלי החלונות הגדולים (הסיכוי להתקל באותו ngram בעל חלון קטן גבוה יותר מהסיכוי להתקל ב-ngram בעל חלון גדול).
2) שיפור לשיטה הקודמת הוא להכיל פונקציית משקל על הכמויות של כל ה-ngrams.
בשיטה זו, נוסיף משקל לכמות של כל ngram בהתחשב לאורכו – ככה נאזן ואף נגדיל במקרים מסוימים את העדיפות ל-ngrams באורכים גדולים יותר. הסיבה שנרצה לעשות זאת כי בתיאוריה לייבלים שמכילים יותר מילים מספקים יותר אינפורמציה על תוכן הקלאסטר.
האתגר האמיתי בשיטה זו הוא מציאת פונקציית המשקל הרלוונטית. פונקציית המשקל צריכה להיות מספיק טובה כדי שתביא תוצאות טובות לכל דאטה סט של בקשות שהוא.

דרך ד':

בדרך זו עדיין נרצה שהלייבל יוגדר ע"י ngram שלקוח מתוך הבקשות באותו הקלאסטר.

אך בשונה מדרך ג', לא נשייך לכל ngram את כמותו אלא נשייך לכל ngram משקל (מספר) שיהיה סכום הערכים המתאימים לאותו ngram בוקטורים הנוצרים ע"י הפעלת TFIDF vectorizer.

המחשבה היא שככל שסכום זה מקבל ערך גבוה יותר - המשמעות של ה-ngram עבור אותו הקלאסטר גדל,

זה הגיוני כי הערכים שמהם מורכב הסכום הוא התוצאה של הנוסחה הבאה (של tfidf):

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

בשיטה זו עדיין השתמשנו בפונקציית משקל.

הגישה הנ"ל הניבה תוצאות לא רעות אך עדיין ניתן להשתפר.

`

דרך ה':

דרך נוספת שניסינו (ולצערנו נכשלה) היא שימוש ב-TFIDF vectorizer בשילוב עם SelectKBest.
בתרחיש זה רצינו למנף את היכולת למצוא את ה-ngrams בעלי הכוח התיאורי הגדול ביותר הקיימים באוסף הבקשות בקלאסטר.
לכן, כפי שעשינו במטלות קודמות, לקחנו את כל הבקשות בקלאסטר והמרנו אותם לווקטורים של Tfidf.
ניזכר שבתהליך זה יכלנו מראש להמנע מ-stop words ובאמת זה מה שעשינו (המחשבה הייתה ששימוש ב- stop words עלול יהיה להפיק לייבלים פחות אינפורמטיביים).
כעת נוכל להריץ את הפונקציה SelectKBest כדי למשוך מספר ngrams בעלי הכוח התיאורי הרב ביותר המייצגים את הבקשות.
לצערנו דרך זו נכשלה, שכן המון מהלייבלים שנוצרו היו שייכים לבקשות בודדות ולא ייצגו נאמנה את כלל הקלאסטר.

לדוגמה –

```
"cluster_name": "remain open",
"requests": [
    "are the national parks closed?",
    "what city parks are open?",
    "are the local parks open?",
    "are the local parks open?",
    "what parks are open?",
    "are the national parks closed?",
    "are the national parks closed?",
    "what city parks are open?",
    "what parks are open?",
    "will the parks and bike trails remain open?",
    "are the local parks open?"
]
```

(המילה remain שייכת לבקשה בודדת בתוך הקלאסטר)

השערה ללמה ניסיון זה נכשל היא מהסיבה כי SelectKBest מחפש את הפיצ'רים (ngrams) שבזכותם ניתן יהיה להפריד את הבקשות אחת מהשנייה. כלומר ngram יבחר בתור לייבל במקרה הזה דווקא עבור בקשות חריגות בנוף בתוך הקלאסטר – וזה בדיוק הפוך מהדרישה שלנו למצוא לייבל שישקף נכון את כלל הבקשות בקלאסטר.

הדרך שבה בחרנו:

לצערנו נגמר לנו הזמן להכנת הפרויקט ולא הגענו לתוצאות המספקות אותנו.

אי לכך ובהתאם לזאת נבחר את הדרך שבה אנו חושבים שהגענו לתוצאות הטובות ביותר.

החלטנו לבחור בדרך ג' עם השיפור של הכלת פונקציית משקל על הכמות של כל ngram.

נבחן בקצרה האם עמדנו ביעדים ליצירת לייבלים שהצבנו לעצמנו בתחילת התהליך –

- לייבלים קצרים (2-6 מילים) - ✓ (כל ngram מראש מוגדר להיות באורך לפחות 2 ולכל היותר 6)
- לייבלים נכונים תחבירית – עמדנו חלקית. מכל הדרכים שניסינו, שימוש ב-ngrams כן מאפשר לייבלים נכונים תחבירית באופן יחסי – זאת כי ngram הוא רצף חלקי ממשפט שמראש היה נכון תחבירית (ברוב המקרים). אבל עדיין כשאנו מדברים על ngram מדובר על משפט חלקי ולכן יתכן שהלייבל שיבחר יהיה שבר של משפט שלם ולכן לא יהיה בהכרח תקין תחבירית עד הסוף. בנוסף החלטנו כן ללכת בגישה אשר מורידה stop words, בגישה זאת אנו מוותרים על מילים מסוימות שתרומתן פחותה, אבל ויתור עליהן יכול לגרוע מהנכונות התחבירית של הלייבל.
- לייבלים אינפורמטיביים ותמציתיים – עמדנו חלקית. מצד אחד, דווקא הורדת stop words מאפשרת להתרכז ב-ngrams המכילים יותר מילים משמעותיות להגדרת הקלאסטר ולכן הלייבלים יהיו יותר אינפורמטיביים. מצד שני, המנגנון שיצרנו אינו מושלם ואינו "יודע" לבחור דווקא את המילים שאכן ישקפו באופן הנכון ביותר את הקלאסטר.

`