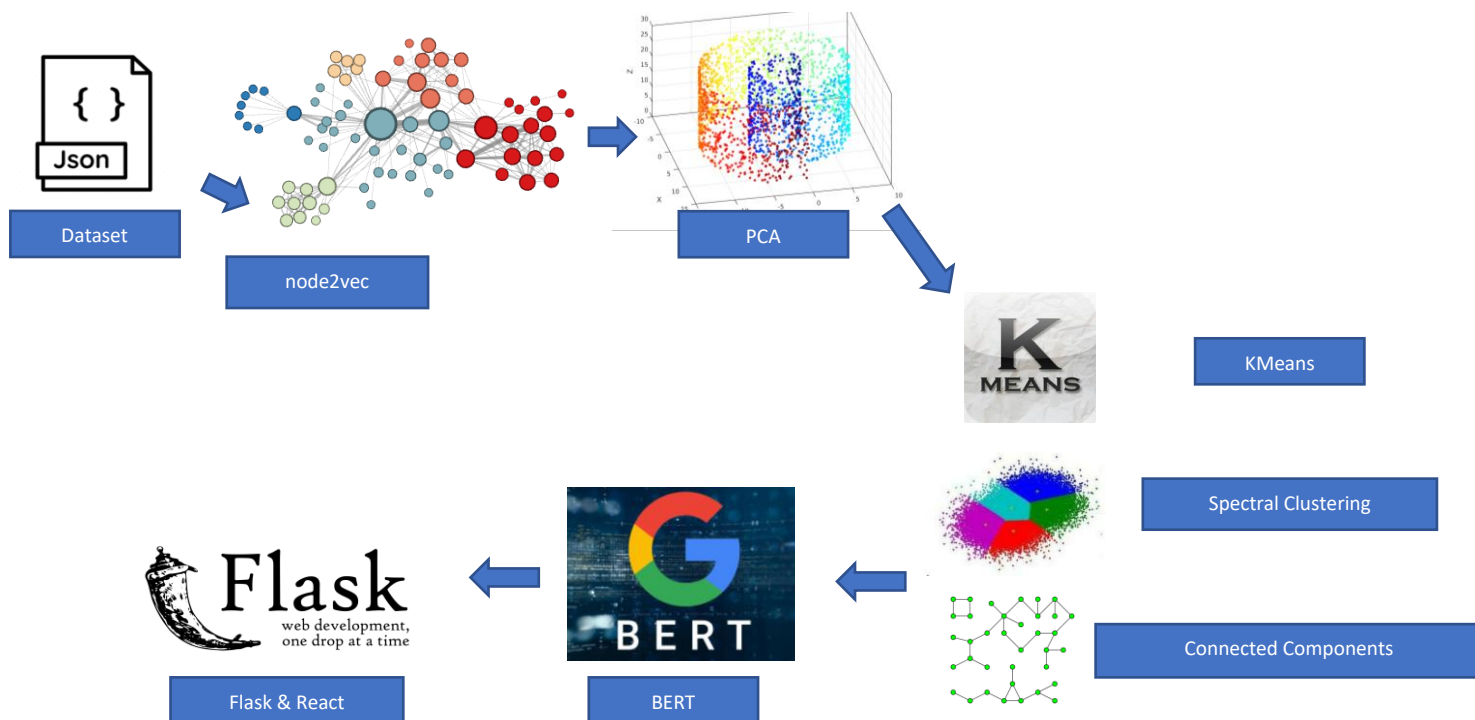


מבט על:

מטרה: זיהוי קהילות חשדות בעלות אופי פדופילי בהינתן שיחות וקיטלוג לקבוצות

תרשים זרימה:



תיאור שלבי הפרויקט בקצרה

הרעיון צמח לאחר שיחה עם המנחה שלנו. אור חימאגר שעובד באלביט. וסיפר לנו על מספר פרויקטים שרצים כרגע בחברה שלהם. אחד מהם היה פרויקט שעשו בשביל המשטרה שבקשה כלי שיאפשר לה ע"ב צ'אטים ברשת לזהות אילו צ'אטים חשודים בשיחות עם אופי פדופילי. אהבנו את הרעיון והחלטנו לנסות לייצר כלי כזה תוך כדי מחקר על הנושא.

מה עשינו בפרויקט עצמו:

המידע המקורי נשמר ב-XML, החלטנו להעביר אותו לפרויקט JSON שהוא נוח יותר לעבודה. בהתחלה אנחנו מקבלים בקובץ JSON גדול מאוד של שיחות. כל שיחה היא בין 2 אנשים. אשר נתון ID מזהה שלהם וההתכתבות בנויה. הוצאנו מתוך קובץ זה את כל הנתונים לתוך אובייקט שיצרנו בפייטון. לאחר מכן יצרנו בעזרת networkx גרף כשאר כל קודקוד בו הוא ID של User מסוים וכל קשת מעידה על שיחה אחת לפחות שהתקיימה בין 2 Users. את הגרף הזה רצינו לקודד לגרף חדש מבוסס מיקום. את זה ביצענו בעזרת האלגוריתם node2vec שהוא אלגוריתם שלוקח דגימות של מסלולים שבהם הוא טייל על הגרף. ולפי זה הוא בעצם מייצר גרף חדש שמייצג את הגרף המקורי כאשר כל קודקוד הוא בעל 64 קודינאטות.

המאמר: <https://towardsdatascience.com/node2vec-embeddings-for-graph-data-32a866340fef>

קוד המקור: <https://github.com/eliorc/node2vec>

<https://github.com/ZviMints/Final-Project> 

ע"י אילון צדוק וצבי מינץ

כעת השתמשנו באלגוריתם שנקרא **Principal component analysis PCA** אשר לוקח את הגרף וממיר אותו לגרף בעל 3 מימדים, ביצענו את זה כדי שיהיה ניתן להציג את הגרף במציאות.

כעת רצינו למצוא באילו שיחות יש סיכוי שיש תוכן עם אופי פדופילי. לשם כך רצינו למצוא clusters שונים בגרף כלומר קבוצת קודקודים שכנראה יש להם נושאי שיחה משותפים. בשביל זה השתמשנו ב-3 אלגוריתמים שונים למציאת clusters בגרף: KMeans, Spectral Clustering, Connected Component. כל אלגוריתם עובד בצורה שונה לניתוח ומציאה של Clusters. לאחר מציאת הclusters השונים סימנו אותם על הגרף ולכל Cluster נתנו שם.

KMeans Clustering: <https://theory.stanford.edu/~sergei/papers/vldb12-kmpar.pdf>

Spectral Clustering: <https://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm.pdf>

כעת בשלב האחרון מה שעניין אותנו זה לנסות להבין על מה כל Cluster מדבר. ובנוסף כאשר זיהינו חיתוכי clusters שם עוד יותר עניין אותנו מה נושאי השיחה המרכזיים כי זו קבוצה יותר קטנה ש-2 או 3 אלגוריתמים שונים מצביעים עליה כ-Cluster כך שכנראה באמת יש להם נושא שיחה מובהק ולכן 3 או 2 אלגוריתמים שונים זיהו אותו.

לשם כך השתמשנו בכלי של גוגל שנקרא BERT היא מערכת לעיבוד שפה טבעית (NLP) המבוססת על בינה מלאכותית. בלי להיכנס להסברים מסובכים, המערכת עוזרת למחשבים להבין שפה אנושית כפי שבני אדם מבינים אותה.

BERT: <https://arxiv.org/pdf/1810.04805.pdf>

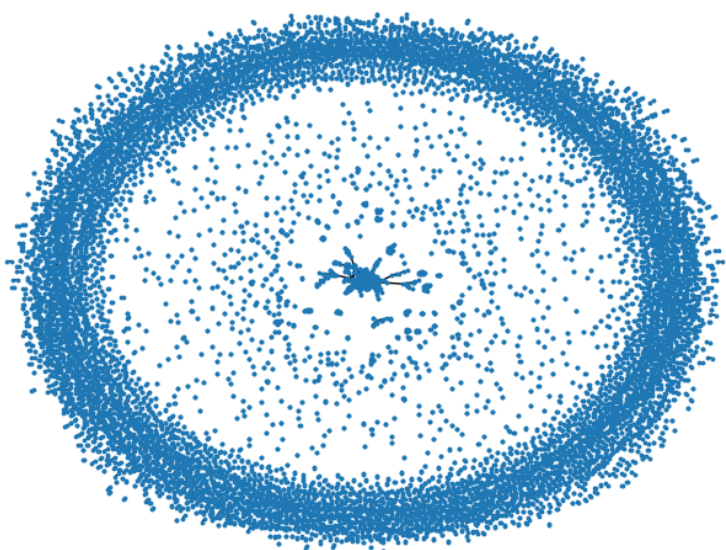
השתמשנו במודל מאומן של Bert הכנסנו אליו את השיחות הרלוונטיות לדוגמא Cluster אחד או חיתוך של clusters ולבסוף ע"י שימוש במודל קיבלנו וקטור יחיד שמאפיין את כל השיחות ב-Cluster

מתוך מילון שיצרנו לכל המילים באנגלית וקטור לכל מילה. מצאנו את חמשת הוקטורים הקרובים ביותר לוקטור שקיבלנו. ובתור פלט החזרנו את חמשת המילים שבעצם הם נושאי השיחה המרכזיים ב-cluster

חלק ראשון Dataset Representation:

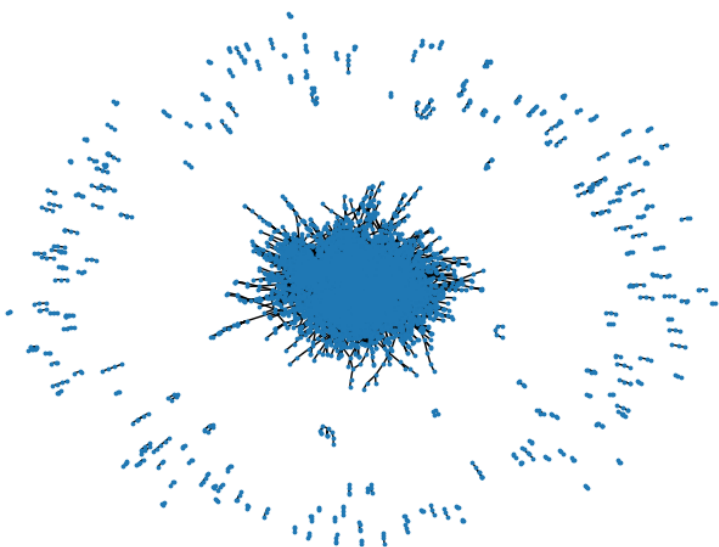
בתחילת הפרויקט קיבלנו מהמנחה שלנו אור חימאגר קישור לקובץ גדול מאוד של json אשר מכיל בתוכו אלפי שיחות אשר חלקן חשודות בפדופיליה.

בשלב הראשון חקרנו את הקובץ. הבנו מה המבנה של כל שיחה. וכדי שיהיה לנו יותר קל יעיל ומהיר לעבוד עם הנתונים של קובץ השיחות הזה עברנו עליו והמרנו אותו לאובייקטים מתאימים שיצרנו. כל הודעה המרנו לאובייקט Message אשר מכיל את ה id של מחבר ההודעה, את הזמן שנכתבה ואת הטקסט עצמו. כל שיחה לאובייקט Conversation אשר מכיל את ה id הייחודי של השיחה, מכיל את ה id של שני המשתתפים בשיחה, ורשימה של אובייקטים של Message. כך בעצם יצרנו רשימה של Conversation אשר מייצגת את קובץ json כולו.



בשלב השני המטרה שלנו הייתה ליצור גרף אשר ייצג את קובץ השיחות בצורה הבאה. כל קודקוד הינו user בעל id ייחודי וכל קשת בין 2 קודקודים מייצגת שיחה / שיחות בין 2 users. בכדי ליצור גרף זה השתמשנו בספרייה networkX אשר עברנו על כל רשימת ה-conversation. מכל אובייקט Conversation כל user מבין 2 ה-users שהשתתפו בשיחה הוסף לגרף בתור קודקוד לפי ה id שלו (בהנחה שהוא לא היה נמצא כבר לפני כן בתוך הגרף) בנוסף מכל אובייקט Conversation כזה הוספה לגרף קשת אשר מייצגת שיחה בין שני ה-users בשיחה. במקרה שכבר נצפתה שיחה אחרת בין 2 users לא הוספה עוד קשת.

בשלב השלישי מתוך הבנו ביחד עם המנחה כי שיחה בין 2 users אשר דיברו רק בניהם ולא עם עוד users אחרים לא מספיק מעניינת אותנו. ושלא נרצה להתעניין בהן ולכן נוריד אותם מהגרף. לשם כך עברנו על כל הרכיבי קשירות בגרף ואת כל הרכיבי קשירות אשר קטנים או שווים ל2 הורדנו מהגרף.



חלק שני Knowledge Graph Embedding:

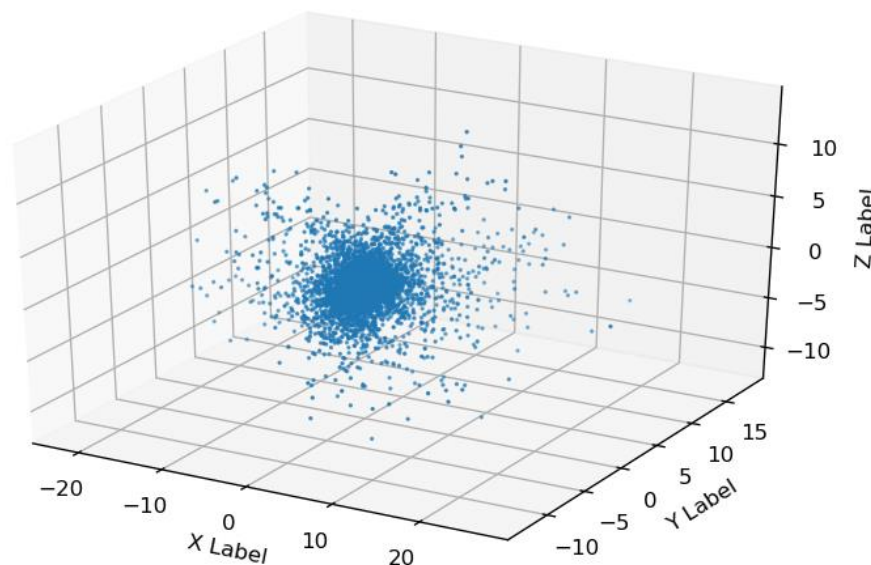
בשלב הראשון קראנו מאמרים אשר מסבירים לעומק על התהליך של embedding בעזרת random walk והבנו בצורה כללית את האלגוריתם שצריך לבצע ואיך זה משתלב בפרויקט שלנו. והחלטנו להשתמש בספרייה שמיועדת לכך node2vec.

בשלב השני למדנו איך להשתמש בספרייה זאת כדי שתשרת בדיוק את המטרה שלנו - קידוד של כל הקשרים (קשר בגרף = התקיימו שיחות) בין users כאשר כל קודקוד בגרף מייצג קידוד של user ואת הקשרים שלו בגרף. לשם כך הגדרנו לה שתקודד את כל הקודקודים בגרף networkX שיצרנו בחלק הקודם. כך שכל קודקוד יקודד ל-64 ממדים, הגדרנו שלכל קודקוד יתבצעו 10 הליכות על הגרף ושכל רצף כזה של הליכה יהיה על 25 קודקודים. בסופו של דבר קידוד כל קודקוד יתבצע בעזרת 10 הרצפים של קודקודים אשר נדגמו עבורו.

בשלב השלישי ביצענו את הרצפים של ההליכות על כל קודקוד לפי ההגדרה וייצרנו קובץ txt עצום אשר מייצג את הרצפים של ההליכות אשר התבצעו על כל קודקוד. קובץ זה מכיל המון שורות כך שכל שורה מכילה id 25 שונים אשר ביחד מהווים מסלול הליכה.

בשלב הרביעי יצרנו את הקידוד עצמו וכך יצרנו בעצם מודל אשר מכיל את אותה כמות קודקודים שהייתה לנו בגרף networkX (לאחר הורדת הקודקודים הלא רלוונטיים עבורנו) רק שכל קודקוד קודד ל-64 ממדים לפי מסלולי ההליכה שנדגמו עבורו.

בשלב החמישי השתמשנו באלגוריתם שנקרא **Principal component analysis PCA** אשר לוקח את הגרף וממיר אותו לגרף בעל 3 ממדים. ביצענו את זה כדי שיהיה ניתן להציג את הגרף במציאות.



חלק שלישי Community Detection:

בשלב זה של הפרויקט הגענו לכך שיש לנו גרף ידע ב-3 ממדים שמציג את הקידוד של כל הקשרים (קשר בגרף = התיקומו שיחות) בין users כאשר כל קודקוד בגרף מייצג קידוד של user ואת הקשרים שלו בגרף.

כעת בעזרת שלושה אלגוריתמים שונים: k-means, spectral clustering, connected component נרצה למצוא קהילות (clusters) של קודקודים. כאשר קודקודים נמצאים באותו Cluster הם עם סבירות גבוהה יותר להיות בעלי מאפיינים משותפים לפי אותו אלגוריתם שקיבץ אותם תחת Cluster זה. וכך בהמשך המטרה שלנו תהיה לנסות לנתח מהו הנושאים העיקריים עליהם מדברים users אשר נמצאים באותו Cluster מתוך הבנה שנושאי השיחה העיקריים משותפים לרוב users ב-cluster. לאחר מכן גם ננסה לנתח חיתוכי clusters מאלגוריתמים שונים. מתוך הבנה שלקודקודים שנמצאים בחיתוך בין clusters יש אפילו סבירות גבוהה עוד יותר להיות בעלי מאפיינים משותפים. ולכן נושאי השיחה העיקריים יהיו אפילו יותר דומים בניהם.

האלגוריתמים השונים בהם השתמשנו למציאת ה-Clusters:

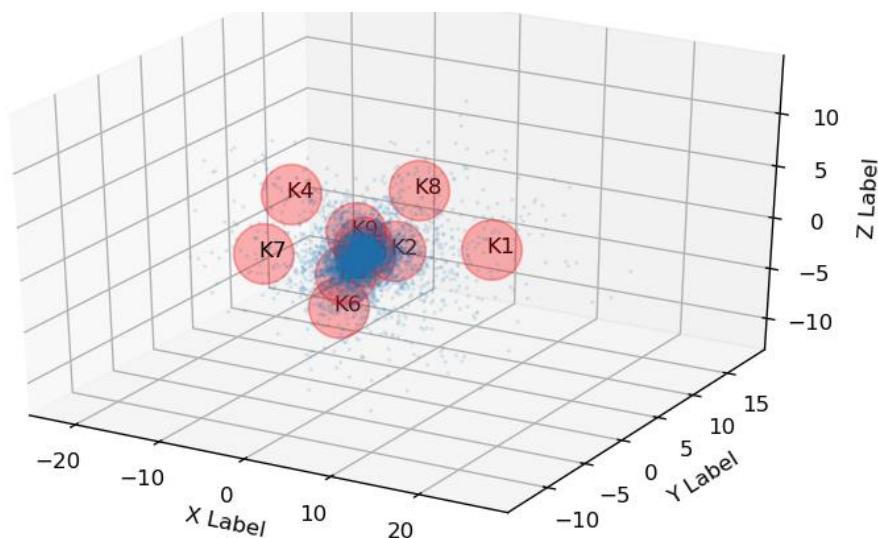
KMeans: נבחר את כמות הclusters לפי שיטת המרפק (יפורט בהרחבה בהמשך). לכל Cluster נבחר בצורה רנדומלית מיקום על שלושת הצירים. בשלב הראשון נעבור על כל הקודקודים בגרף ולכל קודקוד נמצא את הCluster הקרוב ביותר אליו ונשייך אותו אליו. לאחר שנסיים לעבור על כל הקודקודים, בשלב השני עבור כל Cluster נעבור על כל הקודקודים ששייכו אליו ונחשב את המיקום הממוצע שלהם. ולשם נעביר את הCluster שלהם.

כעת שוב נחזור לשלב הראשון ונעבור על כל הקודקודים ונמצא את הCluster הקרוב ביותר אליהם אם שיינו את השיוך של אחד הקודקודים נעבור לשלב השני וכן הלאה לשלב הראשון תהליך זה ייעצר כאשר בשלב הראשון כל הקודקודים יישארו משויכים לאותו Cluster שהיו משויכים אליו מקודם.

ניתן לקרוא בהרחבה בלינק הבא: https://en.wikipedia.org/wiki/K-means_clustering

בקוד שלנו השתמשנו בספרייה sklearn בשביל לבצע את האלגוריתם של k-means ע"י כך שהבאנו לה את הגרף מהשלב הקודם בו כל קודקוד מקודד ל-3 קורדינאטות וגם הבאנו לה את מספר הCluster שאנחנו רוצים ע"י בניית פונקציית `find_elbow` אשר מוצאת את מספר הclusters האופטימלי (יפורט בהרחבה בהמשך).

בסוף התהליך נקבל רשימה של clusters כשאר כל Cluster יוגדר כנקודה ב-3 ממדים אשר מייצגת את מרכז ה-cluster. כדי לדעת מי הם הקודקודים אשר שייכים לאותו Cluster נגדיר רדיוס מסוים כך שכל הקודקודים אשר נמצאים בתוך רדיוס זה שייכים לאותו cluster.

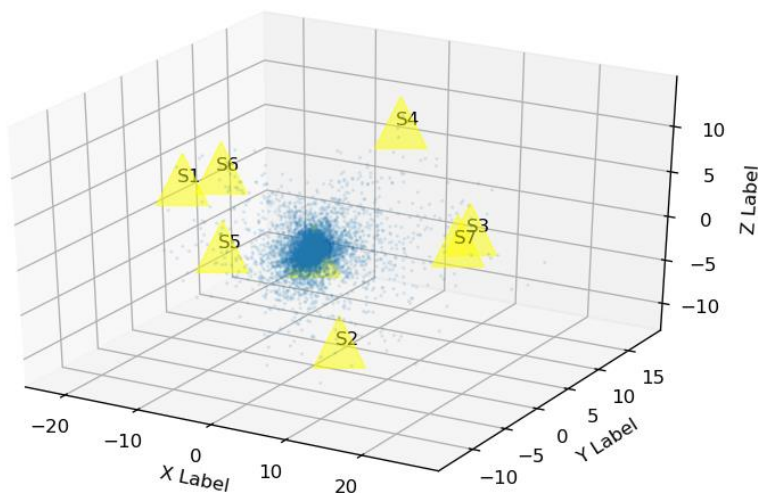


Spectral Clustering

בקוד שלנו השתמשנו בספרייה sklearn בשביל לבצע את האלגוריתם של spectral clustering ע"י כך שהבאנו לה את הגרף מהשלב הקודם בו כל קודקוד מקודד ל 3 קורדינאטות וגם הבאנו לה את מספר clusters שאנחנו רוצים ע"י בניית פונקצייה find_elbow אשר מוצאת את מספר clusters האופטימלי (יפורט בהרחבה בהמשך) בכדי לאפשר הרצה יותר מהירה הגדרנו את כמות clusters להיות מספר קבוע (שמונה clusters) אולם ניתן גם להשתמש פשוט בפונקצייה find_elbow אך יש לקחת בחשבון שזמן ההרצה יתארך משמעותית.

בסוף התהליך נקבל רשימה של clusters כשאר כל cluster יוגדר בנקודה ב 3 ממדים אשר מייצגת את מרכז ה cluster. כדי לדעת מי הם הקודקודים אשר שייכים לאותו cluster נגדיר רדיוס מסוים כך שכל הקודקודים אשר נמצאים בתוך רדיוס זה שייכים לאותו cluster.

כדי להבין לעומק את האלגוריתם של Spectral Clustering ואיך הוא ממיינ את הקודקודים ל clusters שונים ניתן לקרוא בהרחבה בלינק הבא: https://en.wikipedia.org/wiki/Spectral_clustering



Connected Component

הרעיון באלגוריתם זה הוא יחסית הכי פשוט. כל רכיב קשירות יהיה בעצם cluster. בקוד שלנו בעזרת הגרף networkX מהשלב הראשון מצאנו את כל הרכיבי קשירות ודרכם מצאנו את הרכיבי קשירות בגרף הסופי שהגענו אליו ב-3 ממדים. כאשר מרכז Cluster חושב ע"י ממוצע כל מיקומי הקודקודים אשר שייכים לאותו cluster.

בסוף התהליך נקבל רשימה של clusters כאשר כל Cluster יוגדר בנקודה ב-3 ממדים אשר מייצגת את מרכז ה cluster. כדי לדעת מי הם הקודקודים אשר שייכים לאותו Cluster נגדיר רדיוס מסוים כך שכל הקודקודים אשר נמצאים בתוך רדיוס זה שייכים לאותו cluster.

כדי להבין יותר בפירוט את הרעיון של אלגוריתם זה ניתן לקרוא בהרחבה בלינק הבא:

[https://en.wikipedia.org/wiki/Component_\(graph_theory\)](https://en.wikipedia.org/wiki/Component_(graph_theory))

שיטת המרפק:

לאלגוריתמים k-means ו spectral clustering מעבר לקלט של גרף הידע בעל ה-3 ממדים לפיו הם יעבדו היה צורך גם להכניס קלט שהוא כמות הקהילות (clusters) שהם יעבדו לפיו. כדי לחשב את כמות clusters בדקנו לפי שיטת המרפק.

1. עבור על הגדלים הרלוונטים של ה clusters:

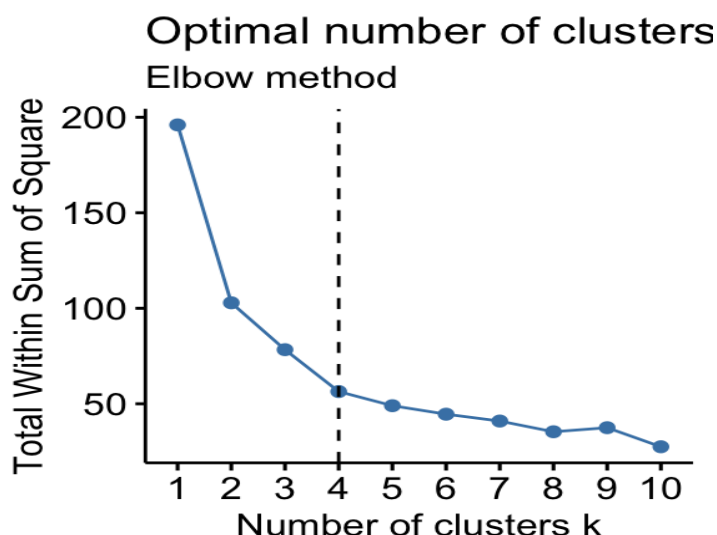
1.1 נעבור על כל הנקודות בכל Cluster ונסכום את המרחק בריבוע של כל הנקודות באותו

Cluster מהנקודה שמייצגת את מרכז Cluster

1.2 נבצע סכום של הסכומים שקיבלנו בכל clusters

2. נבנה גרף כאשר ציר ה x הוא כמות clusters וציר ה y הוא הסכום שחישבנו בסעיף 1.2 עבור כל כמות clusters שונה.

3. נקבל גרף שנראה כך:



נבחר את הנקודה שנראית כמו הקצה של המרפק. בדוגמה כאן ניתן לראות שזה אומר שנבחר כמות clusters של 4. הסיבה לבחירת מספר זה היא כי אחריו הירידה בסכומים שחישבנו בסעיף 1.2 יורדת בצורה הרבה יותר איטית. כלומר התרומה של הגדלת מספר ה clusters כבר לא יהיה מאוד משמעותי.

חלק רביעי Topic Extraction Per Cluster

כעת הגענו לשלב הקריטי והמעניין ביותר בפרויקט. בשלב זה אנחנו בעצם מתמקדים בקהילות (clusters) שנמצאו מתוך הבנה ששם לפי האלגוריתמים ישנה פעילות שיכולה להיות מעניינת או רלוונטית עבורנו, מכיוון שusers שנמצאים באותו Cluster יש להם לפי האלגוריתם מאפיינים דומים ולכן סיכוי גדול שנושאי השיחה העיקריים שלהם דומים.

כעת ברצוננו להצליח לתמצת את כל השיחות שמתבצעות בכל אחד מהקהילות האלו לכדי נושאים עיקריים עליהם מדובר שם. בנוסף במקרה שקיימים חיתוכים בין הקהילות הנ"ל נרצה גם להצליח לתמצת את החיתוכים האלו לכדי נושאים עיקריים. וזאת מתוך ההבנה כי חיתוכי קהילות הינם קבוצות users אשר סיכוי גבוה יותר שיש להם נושאי שיחה עיקריים משותפים מאחר ומספר אלגוריתמים שונים סווגו אותם כבעלי מאפיינים דומים.

לצורך כך בחרנו להשתמש בכלי שנקרא BERT.

BERT ראשי תיבות של Bidirectional Encoder Representations from Transformers היא מערכת לעיבוד שפה טבעית (NLP) המבוססת על בינה מלאכותית. בלי להיכנס להסברים מסובכים, המערכת עוזרת למחשבים להבין שפה אנושית כפי שבני אדם מבינים אותה.

השימוש בBERT כולל 2 שלבים:

שלב ראשון הוא הכנת המידע אשר יהווה input עבור BERT כלומר כל השיחות בקהילה מסוימת או כל השיחות בחיתוך של קהילות מסוימות.

שלב שני קידוד הנתונים בעזרת BERT לוקטור יחיד אשר מייצג את כל השיחות. לאחר מכן מציאת הנושא (במילים) אשר ווקטור זה מייצג.

נפרט כעת את השלבים לעומק:

השלב הראשון:

✓ המרנו את הקובץ json עם כל השיחות לרשימה של אובייקטי Conversation לאחר סינון השיחות הלא רלוונטיות דבר אשר ייעל מאוד את זמן הריצה. השיחות הלא רלוונטיות הן השיחות אשר התקיימו בין 2 users. אשר אותם 2 חוץ מלדבר אחד עם השני לא ניהלו עוד שיחות בכלל.

✓ מתוך הגרף ידע ב3 ממדים שבנינו שמציג את הקידוד של כל הקשרים (קשר = התקיימו שיחות) בין users כאשר כל קודקוד בגרף מייצג קידוד של user ואת הקשרים שלו בגרף נחלץ את id של כל ה users בקהילה הנבחרת / בחיתוך הקהילות הנבחרות. (כדי לחלץ את id נעזרנו בגרף של networkX כך שהתאמנו כל user ממנו אל user המתאים בגרף ידע ב3 מימדים)

✓ נעבור על הרשימה של אובייקטי Conversation שיצרנו ונסנן אותם ע"י כך שנחלץ משם את כל השיחות אשר שני ה users אשר מנהלים את השיחה נמצאים בקהילה הנבחרת או בחיתוך הנבחר.

השלב השני:

- ✓ הורדנו מהאינטרנט מסמך TXT אשר מכיל את כל האופציות של מילים באנגלית 28118 מילים. הרצנו עליו script שיצרנו כדי לסנן את כל המילים חסרו המשמעות שהיו בו. כגון: ##?, * & ועוד אחרים. עד שלבסוף נשארו על קובץ TXT מצומצם בעל 8273 מילים. את כל המילים האלו אחת אחרי השנייה הכנסנו לתוך המודל המאומן של BERT ויצרנו dictionary (map) של מילה, ווקטור.
- ✓ כעת נכניס את כל אובייקטי ה Conversation שאספנו (לאחר הסינון בשלב הראשון) לתוך מודל מאומן של BERT. (בכדי שנוכל לקבל תוצאות בזמן סביר יצרנו משתנה קבוע אשר מגביל את כמות השיחות אשר נקודד עבור בקהילה \ חיתוך קהילות הנבחרות אולם בכדי לקבל תוצאות מדוייקות ככל האפשר נכניס למשתנה זה את הערך המקסימלי של integer וכך כל השיחות שמשוייכות לקהילה \ חיתוך קהילות הנבחרות יקודדו).
- ✓ עבור כל שיחה נבצע קידוד של כל משפט. מכל משפט יחזרו מהמודל המאומן כמות וקטורים ככמות המילים במשפט. נבצע ממוצע של וקטורים אלו וכך נקבל וקטור שמייצג משפט. לאחר מכן נבצע ממוצע של כל הוקטורים שנקבל וכך נקבל וקטור שמייצג שיחה. לבסוף נבצע ממוצע של כל וקטורי השיחות לכדי וקטור אחד אשר מייצג את כל השיחות בקהילה \ חיתוך קהילות הנבחרות.
- ✓ מתוך ה dictionary שיצרנו מצאנו את 5 הוקטורים בעלי המרחק האוקלידי הקטן ביותר מהווקטור שמייצג את כל השיחות. ובעזרת ה dictionary קיבלנו את המילים שאותם חמשת ווקטורים אלו מייצגים. אלו בעצם חמשת נושאי השיחה שלנו.

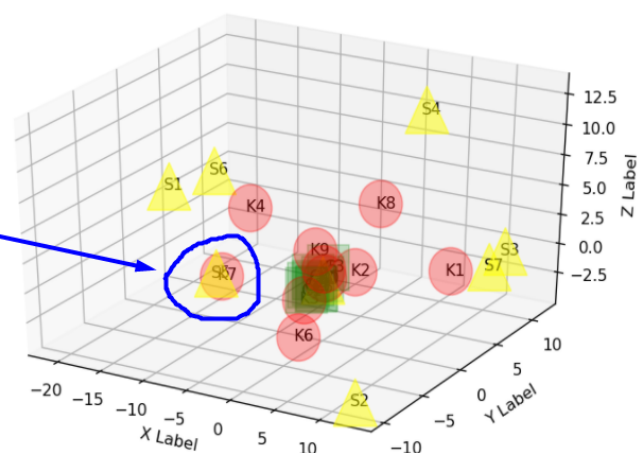
BERT

Find Specific Topic All Clusters

Select Clusters Intersection

[K7, S5]

K0 Topic: ["introductions", "advisors", "gulp", "notification", "siren"]



חלק חמישי GUI Implementation:

על מנת לאפשר למשתמש לראות בצורה נוחה את הקהילות והנושאים שקבלנו לאחר עיבוד המידע, היינו צריכים ליצור ממשק גרפי אשר מציג בצורה נוחה את המידע בכל אחד מהשלבים שצוינו קודם לכן, בנוסף, רצינו ליצור כלי דינמי אשר יכול לבצע את כל הפעולות מחדש על dataset חדש, ובכך נוכל בעצם לשחזר את כל השלבים ולעשות עיבוד נתונים חדשים בהתאם לנתונים חדשים ולקבל נושאים חדשים וקהילות חשודות חדשות, בנוסף היינו צריך לציין פרוטוקול מוסכם על מנת לקבל את הקלט מהמשתמש על מנת שנוכל לעבד את המידע בצורה תקינה.

לשם כך, החלטנו ליצור מערכת מבוססת web-application אשר מתממשת עם האלגוריתמים השונים אשר התבצעו בשלבים הקודמים, כלומר framework אשר תומך בפייטון, ולאחר מכן היינו צריכים למצוא פתרון על מנת להציג את הנתונים בצורה דינמית באתר ולכן היינו צריכים לחפש ספרייה אשר עוטפת את javascript על מנת לאפשר חזות דינמית וחווית משתמש אופטימלית.

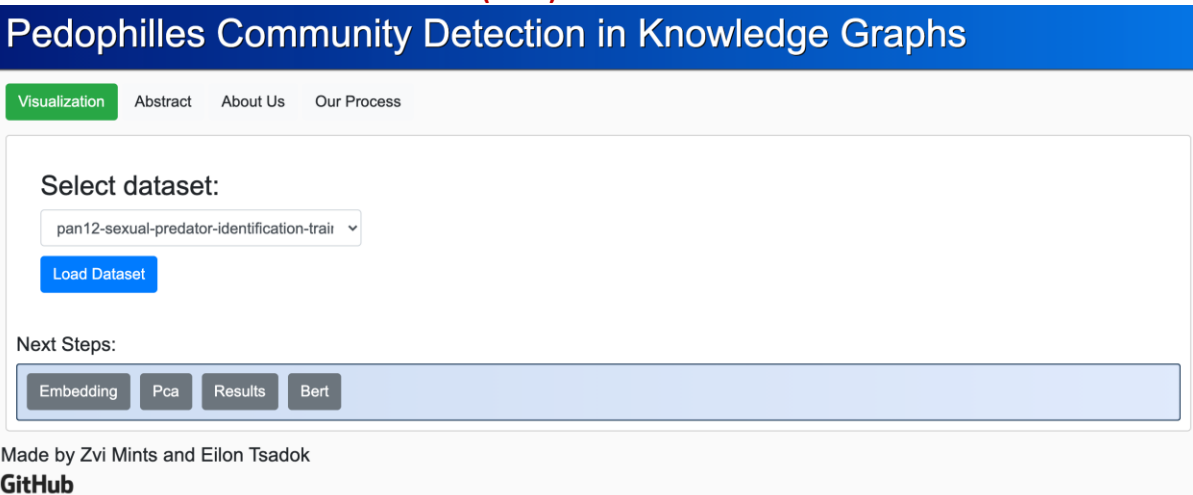
החלטנו להשתמש ב-Framework אשר נקרא Flask אשר נותן מענה לפיתוח אפליקציות מבוססות web בעזרת השפה python, Flask היא סביבה לפיתוח Web בשפת Python. זוהי Micro Framework המבוססת על פלאגינים ולכן ליבת הספרייה מאוד רזה, מה שאומר שקל מאוד ללמוד איך להתחיל להשתמש בה, וזה התאים מעולה לצרכים שלנו.

Flask בין היתר מספקת אפשרות לצד לקוח ולא רק לצד שרת, אך החלטנו ללכת על React אשר הינה ספריית JavaScript הצהרתית, יעילה וגמישה של לבניית ממשקי משתמש אשר נכתבה ע"י פייסבוק.

כדי ליצור חיבור בין צד השרת לצד הלקוח, החלטנו לייצר routes עבור כל אחד מהשלבים אשר בוצעו קודם לכן, ובכך בעצם להתאים את העבודה שבוצעה בשלבים קודם לכן לקריאות אשר יתבצעו ע"י הלקוח בסופו של דבר.

השלבים:

שלב הטעינה: (Load)



שינוי ה-Dataset גורם לשינוי בשם ה-dataset, אשר משפיע על בקשת הלקוח לצד השרת.

בעת לחיצה על הכפתור [Load dataset](#) מתבצעת קריאת REST באופן הבא:

כתובת: /load

סוג בקשה: POST

גוף הבקשה: dataset והאם להשתמש במידע שקיים בשרת או לבצע חישוב מחדש

תשובות:

500: שגיאת שרת, מופיע הודעה עם התוכן Server Error

200: המידע הבא נשמר:

before_path – התמונה של הnetwork לפני ההסרה של צמדי הקודקודים
after_path – התמונה של הnetwork אחרי ההסרה של צמדי הקודקודים
before – הקשתות בגרף לפני ההסרה
after – הקשתות בגרף אחרי ההסרה
graphData – מספר הקודקודים והקשתות בכל גרף

Graph Information:

Before Removing 2-connected components: 202623 Nodes, 111986 links
After Removing 2-connected components: 9387 Nodes, 15368 links

Graph as JSON:

Before remove 2-connected components

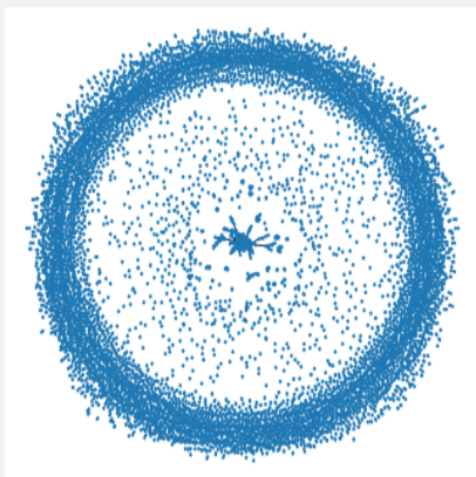
```
[{"source": "0a39f78bcb297ab0ebe8a29c28bfed89", "target": "60659cfd992013e610f285c46692d28"}, {"source": "0a39f78bcb297ab0ebe8a29c28bfed89", "target": "8d5c0dd1af8dbfd69120fd4882cb3a68"}, {"source": "0a39f78bcb297ab0ebe8a29c28bfed89", "target": "f9b36de6c178d98ca2171bb653acb05e"}, {"source": "0a39f78bcb297ab0ebe8a29c28bfed89", "target": "4d389837f8ff4967ac73a3d840c55ece"}, {"source": "0a39f78bcb297ab0ebe8a29c28bfed89", "target": "f139aba52f9fa1394b4034a7954b2220"}]
```

After remove 2-connected components

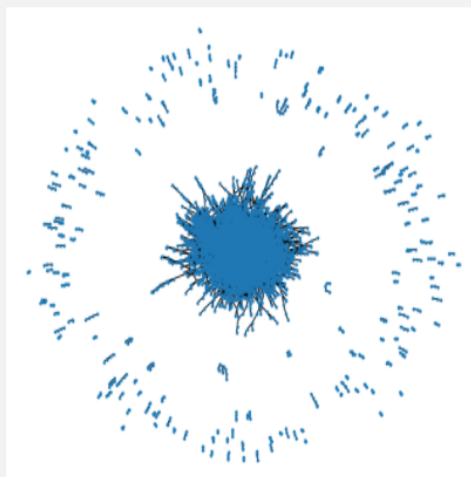
```
[{"source": "0a39f78bcb297ab0ebe8a29c28bfed89", "target": "60659cfd992013e610f285c46692d28"}, {"source": "0a39f78bcb297ab0ebe8a29c28bfed89", "target": "8d5c0dd1af8dbfd69120fd4882cb3a68"}, {"source": "0a39f78bcb297ab0ebe8a29c28bfed89", "target": "f9b36de6c178d98ca2171bb653acb05e"}, {"source": "0a39f78bcb297ab0ebe8a29c28bfed89", "target": "4d389837f8ff4967ac73a3d840c55ece"}, {"source": "0a39f78bcb297ab0ebe8a29c28bfed89", "target": "f139aba52f9fa1394b4034a7954b2220"}]
```

Visualization:

Before remove 2-connected components



After remove 2-connected components



Generate Random Walks & Embedding

Embedding

כתובת: /embedding

סוג בקשה: POST

גוף הבקשה: dataset והאם להשתמש במידע שקיים בשרת או לבצע חישוב מחדש

תשובות:

500: שגיאת שרת, מופיע הודעה עם התוכן Server Error

200: המידע הבא נשמר:

Dataset: pan12-sexual-predator-identification-test-corpus-2012-05-17

Walks Generated:

25 Number of nodes in each walk

10 Number of walks per node

[Checkout Generated Walks](#)

Nodes Embedded **Successfully**

walks – קובץ txt אשר מכיל בתוכו את כל המסלולים אשר נדגמו (כל שורה מכילה user id 25)
num_walks – מספר המסלולים שנבצע עבור קודקוד (user) יחיד.
walk_length – אורך המסלול, כלומר מספר users שנדגמו לאורך המסלול.

```
row 1: 6167d4aac2726f67fcbdb47034e095d... 6167d4aac2726f67fcbdb47034e095d... 6167d4aac2726f67fcbdb47034e095d... 6167d4aac2726f67fcbdb47034e095d... 6167d4aac2726f67fcbdb47034e095d...
row 2: 62b7346cb77f91bbacfb680b0db2b85... 62b7346cb77f91bbacfb680b0db2b85... 62b7346cb77f91bbacfb680b0db2b85... 62b7346cb77f91bbacfb680b0db2b85... 62b7346cb77f91bbacfb680b0db2b85...
row 3: 44c3977bc056745dd9e1d2f9bf631b36... 44c3977bc056745dd9e1d2f9bf631b36... 44c3977bc056745dd9e1d2f9bf631b36... 44c3977bc056745dd9e1d2f9bf631b36... 44c3977bc056745dd9e1d2f9bf631b36...
row 4: 4443b5072efc45f02738999681abb... 4443b5072efc45f02738999681abb... 4443b5072efc45f02738999681abb... 4443b5072efc45f02738999681abb... 4443b5072efc45f02738999681abb...
row 5: 7472440bc9f0bb5b9d4e419cfaaf18... 7472440bc9f0bb5b9d4e419cfaaf18... 7472440bc9f0bb5b9d4e419cfaaf18... 7472440bc9f0bb5b9d4e419cfaaf18... 7472440bc9f0bb5b9d4e419cfaaf18...
row 6: 479da0486c46d6324240b63610a36... 479da0486c46d6324240b63610a36... 479da0486c46d6324240b63610a36... 479da0486c46d6324240b63610a36... 479da0486c46d6324240b63610a36...
row 7: 71908f0f9c7768a05851d0d3c753d4d... 71908f0f9c7768a05851d0d3c753d4d... 71908f0f9c7768a05851d0d3c753d4d... 71908f0f9c7768a05851d0d3c753d4d... 71908f0f9c7768a05851d0d3c753d4d...
row 8: 1e18c22e70bde49ce1775bba579bd6... 1e18c22e70bde49ce1775bba579bd6... 1e18c22e70bde49ce1775bba579bd6... 1e18c22e70bde49ce1775bba579bd6... 1e18c22e70bde49ce1775bba579bd6...
row 9: bbf5d129d54bea20542bdf31d8d5c... bbf5d129d54bea20542bdf31d8d5c... bbf5d129d54bea20542bdf31d8d5c... bbf5d129d54bea20542bdf31d8d5c... bbf5d129d54bea20542bdf31d8d5c...
row 10: f96d123b2141b0d4c8bbf7d31a8d5c... f96d123b2141b0d4c8bbf7d31a8d5c... f96d123b2141b0d4c8bbf7d31a8d5c... f96d123b2141b0d4c8bbf7d31a8d5c... f96d123b2141b0d4c8bbf7d31a8d5c...
row 11: 8abf08fd72cc4a06f48ddc11739f... 8abf08fd72cc4a06f48ddc11739f... 8abf08fd72cc4a06f48ddc11739f... 8abf08fd72cc4a06f48ddc11739f... 8abf08fd72cc4a06f48ddc11739f...
row 12: 984724c86e0fc0a4320d6da9056728... 984724c86e0fc0a4320d6da9056728... 984724c86e0fc0a4320d6da9056728... 984724c86e0fc0a4320d6da9056728... 984724c86e0fc0a4320d6da9056728...
row 13: b1c11f0f93dc6c08a46d7065429142e... b1c11f0f93dc6c08a46d7065429142e... b1c11f0f93dc6c08a46d7065429142e... b1c11f0f93dc6c08a46d7065429142e... b1c11f0f93dc6c08a46d7065429142e...
row 14: b1b775e1d790c050448957a72a488060... b1b775e1d790c050448957a72a488060... b1b775e1d790c050448957a72a488060... b1b775e1d790c050448957a72a488060... b1b775e1d790c050448957a72a488060...
row 15: fc9592bc19249eff078860d90d4282... fc9592bc19249eff078860d90d4282... fc9592bc19249eff078860d90d4282... fc9592bc19249eff078860d90d4282... fc9592bc19249eff078860d90d4282...
row 16: 9a0562444d8a398919e800c0216a... 9a0562444d8a398919e800c0216a... 9a0562444d8a398919e800c0216a... 9a0562444d8a398919e800c0216a... 9a0562444d8a398919e800c0216a...
row 17: d669bc384c420145eb17d64713be5e... d669bc384c420145eb17d64713be5e... d669bc384c420145eb17d64713be5e... d669bc384c420145eb17d64713be5e... d669bc384c420145eb17d64713be5e...
row 18: c77a59844c51ef3d0104474c2c9f5... c77a59844c51ef3d0104474c2c9f5... c77a59844c51ef3d0104474c2c9f5... c77a59844c51ef3d0104474c2c9f5... c77a59844c51ef3d0104474c2c9f5...
row 19: 48a4fbac2491bb8aee694a307dca3... 48a4fbac2491bb8aee694a307dca3... 48a4fbac2491bb8aee694a307dca3... 48a4fbac2491bb8aee694a307dca3... 48a4fbac2491bb8aee694a307dca3...
row 20: 788f039c23906668895d5737792acc... 788f039c23906668895d5737792acc... 788f039c23906668895d5737792acc... 788f039c23906668895d5737792acc... 788f039c23906668895d5737792acc...
row 21: a04b6f6061c51a0ff90056099f6651a... a04b6f6061c51a0ff90056099f6651a... a04b6f6061c51a0ff90056099f6651a... a04b6f6061c51a0ff90056099f6651a... a04b6f6061c51a0ff90056099f6651a...
row 22: 20eb6eb59584bf47c30b2173190c47b... 20eb6eb59584bf47c30b2173190c47b... 20eb6eb59584bf47c30b2173190c47b... 20eb6eb59584bf47c30b2173190c47b... 20eb6eb59584bf47c30b2173190c47b...
row 23: 1b60e95a2c6705cf6331ff9e89826... 1b60e95a2c6705cf6331ff9e89826... 1b60e95a2c6705cf6331ff9e89826... 1b60e95a2c6705cf6331ff9e89826... 1b60e95a2c6705cf6331ff9e89826...
row 24: 0d4157bf6a2c75208003a765646e99fa... 0d4157bf6a2c75208003a765646e99fa... 0d4157bf6a2c75208003a765646e99fa... 0d4157bf6a2c75208003a765646e99fa... 0d4157bf6a2c75208003a765646e99fa...
row 25: f0c9833955f0b0b95c5d606faf2b01... f0c9833955f0b0b95c5d606faf2b01... f0c9833955f0b0b95c5d606faf2b01... f0c9833955f0b0b95c5d606faf2b01... f0c9833955f0b0b95c5d606faf2b01...
row 26: ffd8c8e91ca57a70c3d5f979d21600f6... ffd8c8e91ca57a70c3d5f979d21600f6... ffd8c8e91ca57a70c3d5f979d21600f6... ffd8c8e91ca57a70c3d5f979d21600f6... ffd8c8e91ca57a70c3d5f979d21600f6...
row 27: 36e8f752290832bc90b021c356510f... 36e8f752290832bc90b021c356510f... 36e8f752290832bc90b021c356510f... 36e8f752290832bc90b021c356510f... 36e8f752290832bc90b021c356510f...
row 28: 01f8b471f1bb878cdd2ff7d7a78e... 01f8b471f1bb878cdd2ff7d7a78e... 01f8b471f1bb878cdd2ff7d7a78e... 01f8b471f1bb878cdd2ff7d7a78e... 01f8b471f1bb878cdd2ff7d7a78e...
row 29: 870f8a0e9e2d90a5f9c7e0ed792b075a... 870f8a0e9e2d90a5f9c7e0ed792b075a... 870f8a0e9e2d90a5f9c7e0ed792b075a... 870f8a0e9e2d90a5f9c7e0ed792b075a... 870f8a0e9e2d90a5f9c7e0ed792b075a...
row 30: f2631fb6f5c505c6cc0c7c967683d3d5a... f2631fb6f5c505c6cc0c7c967683d3d5a... f2631fb6f5c505c6cc0c7c967683d3d5a... f2631fb6f5c505c6cc0c7c967683d3d5a... f2631fb6f5c505c6cc0c7c967683d3d5a...
row 31: 45c3e969cd3d7c7304111e1abb872bf... 45c3e969cd3d7c7304111e1abb872bf... 45c3e969cd3d7c7304111e1abb872bf... 45c3e969cd3d7c7304111e1abb872bf... 45c3e969cd3d7c7304111e1abb872bf...
row 32: 270c75902c81288fa50f41d58155a62... 270c75902c81288fa50f41d58155a62... 270c75902c81288fa50f41d58155a62... 270c75902c81288fa50f41d58155a62... 270c75902c81288fa50f41d58155a62...
row 33: 7fc2956c70a886f74542845fc10f199... 7fc2956c70a886f74542845fc10f199... 7fc2956c70a886f74542845fc10f199... 7fc2956c70a886f74542845fc10f199... 7fc2956c70a886f74542845fc10f199...
```

פרויקט גמר

נושא: Pedophile's Community Detection in Knowledge Graphs
מנחים: ד"ר דביר עמית זאב ואור חמיגאר (מנחה חיצוני – חברת אלביט)
רכז פרויקטים: יוסי זגורי

<https://github.com/ZviMints/Final-Project> 

ע"י אילון צדוק וצבי מינץ

שלב המרת הממדים: (pca)

Principal component analysis

PCA

כתובת: /pca

סוג בקשה: POST

גוף הבקשה: dataset והאם להשתמש במידע שקיים בשרת או לבצע חישוב מחדש

תשובות:

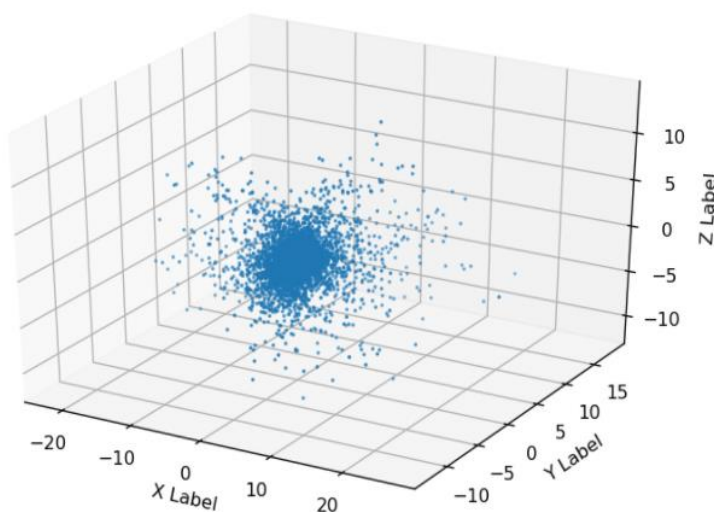
500: שגיאת שרת, מופיע הודעה עם התוכן Server Error

200: המידע הבא נשמר:

base – התמונה של הגרף הבסיסי לאחר ההמרה ל-3 ממדים

Dataset: pan12-sexual-predator-identification-test-corpus-2012-05-17

Base Graph:



פרויקט גמר

נושא: Pedophile's Community Detection in Knowledge Graphs
מנחים: ד"ר דביר עמית זאב ואור חמיגאר (מנחה חיצוני – חברת אלביט)
רכז פרויקטים: יוסי זגורי

<https://github.com/ZviMints/Final-Project> 

ע"י אילון צדוק וצבי מינץ

שלב הצגת הקהילות החשודות: (Results)

Results

Check which algorithm to show:

☒ KMeans

☐ Spectral Clustering

☐ Connected Components

Show Graph

כתובת: /results

סוג בקשה: POST

גוף הבקשה: dataset ורשימה של אלגוריתמים (KMeans, Spectral Clustering, ConnectedComponents)

תשובות:

500: שגיאת שרת, מופיע הודעה עם התוכן Server Error

200: המידע הבא נשמר:

Image – התמונה שמתקבלת כפלט לאחר הסיווג של האלגוריתמים שהתקבלו בקלט.

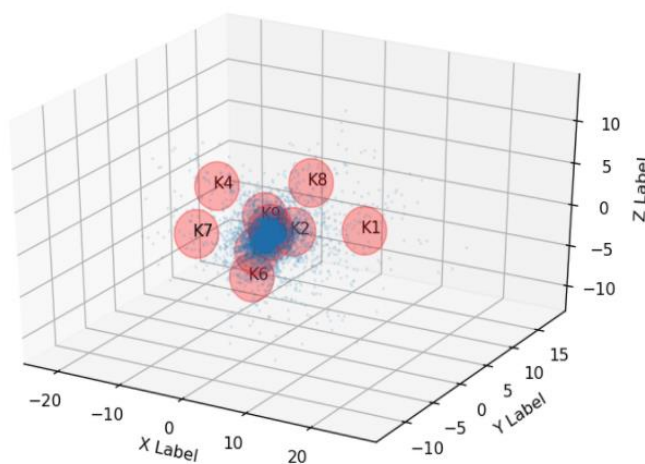
Dataset: pan12-sexual-predator-identification-test-corpus-2012-05-17

Dictionary:

Connected-Components Clustering KMeans Clustering Spectral Clustering Nodes

Algorithms: kmeans

Image: /data/pca/pan12-sexual-predator-identification-test-corpus-2012-05-17/kmeans.png



פרויקט גמר

נושא: Pedophile's Community Detection in Knowledge Graphs
מנחים: ד"ר דביר עמית זאב ואור חמיגאר (מנחה חיצוני – חברת אלביט)
רכז פרויקטים: יוסי זגורי

<https://github.com/ZviMints/Final-Project> 

ע"י אילון צדוק וצבי מינץ

שלב חילוץ נושאי השיחה: (Bert)

Gather insights with Bert

Continue to BERT

כתובת: bert /

סוג בקשה: POST

גוף הבקשה: dataset והאם להשתמש במידע שקיים בשרת או לבצע חישוב מחדש
בנוסף שולחים cluster ספציפי או חיתוך clusters שמתוכם נחלץ את נושאי השיחה העיקריים

תשובות:

500: שגיאת שרת, מופיע הודעה עם התוכן Server Error

200: המידע הבא נשמר:

topic – רשימה של 5 מילים אשר מציינות את נושאי השיחה העיקריים בcluster

BERT

Find Specific Topic

All Clusters

Select Clusters Intersection

[K7, S5]

Submit

Topic: ['glint', 'siren', 'introductions', 'notification', 'gulp']

כתובת: getLabels /

סוג בקשה: POST

גוף הבקשה: dataset והאם להשתמש במידע שקיים בשרת או לבצע חישוב מחדש

תשובות:

500: שגיאת שרת, מופיע הודעה עם התוכן Server Error

200: המידע הבא נשמר:

labels – רשימה של כל האופציות האפשריות לcluster או חיתוכי clusters

BERT

Find Specific Topic

All Clusters

Fetch Data

Clusters	Topic
[K0]	
[K1]	
[K2]	
[K3]	
[K4]	
[K5]	
[K6]	
[K7]	
[K8]	
[K9]	

Lsflask
gunicorn
networkx
matplotlib
flask_session
node2vec
numpy
sklearn
gensim
bert-embedding
pickle-mixin
ijson
npm
react
flask

צד שרת: [/http://localhost:5000](http://localhost:5000)

ב-FinalProject/Final-Project/API/client/server-
הרץ:

```
export FLASK_APP=application.py  
flask run
```

צד לקוח: [/http://localhost:3000](http://localhost:3000)

ב-FinalProject/Final-Project/API/client-
הרץ:

```
npm start
```

קישורים שימושיים:

- **Dataset Source**
 - <https://pan.webis.de/clef12/pan12-web/sexual-predator-identification.html>
 - .json format **training** dataset: <https://f2h.io/8b3xzuy6rnq4>
 - .json format **test** dataset: <https://f2h.io/5n7mcc7cx3b6>
- **Knowledge Graph Vertices Embedding**
 - https://github.com/jimmywangheng/knowledge_representation_pytorch
 - <https://radimrehurek.com/gensim/models/word2vec.html>
- **PCA (Principal Component Analysis)**
 - https://en.wikipedia.org/wiki/Principal_component_analysis
 - <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>
- **Spectral Clustering**
 - https://en.wikipedia.org/wiki/Spectral_clustering
 - <https://scikitlearn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html>
 - [https://en.wikipedia.org/wiki/Elbow_method_\(clustering\)](https://en.wikipedia.org/wiki/Elbow_method_(clustering))
- **Topic Extraction**
 - https://en.wikipedia.org/wiki/Topic_model
 - <https://towardsdatascience.com/the-complete-guide-for-topicextraction-in-python-a6aaa6cedbbc>
- **Google BERT Model Utilization in Python**
 - <https://towardsdatascience.com/word-embedding-using-bert-inpython-dd5a86c003423>