**אחזור מידע**

**404 brain not found**

**The Final Project Answers Link That we presented :** [**404\_Final\_Project\_answers\_site**](https://information-retrieval-course-kdnz-final-presentation.vercel.app/)

The Final Project GitHub Link: [404\_Final\_GitHub\_Presentation](https://github.com/razimograbi/InformationRetrievalCourse/tree/main/FinalPresentation)

שאלה 1:

בחרנו להתמקד באתר Urban Dictionary שהוא מילון אינטרנטי שמסביר מילים, סלנגים וביטויים שאנשים משתמשים בהם בחיי היומיום ובאינטרנט. אפשר למצוא שם מונחים שאינם נמצאים במילונים רגילים, כמו מילים מצחיקות, כינויים, או ביטויים נפוצים באינטרנט. אנשים יכולים להוסיף הסברים משלהם למילים, לכתוב דוגמאות איך משתמשים בהן, וגם להצביע אם הם אוהבים את ההגדרה או לא. האתר מתאים במיוחד לאנשים שרוצים להבין ביטויים או מונחים פופולריים בתרבות היום-יום.

הקישור לאתר: <https://www.urbandictionary.com/>

שאלה 2:

Query 1: lmao "or" cant  
תוצאה: מצטערים, לא מצאנו: lmao OR cant  
מידע נדרש:  
הגדרה של "lmao” או “can't" כפי שמשתמשים בהם בשיחות אונליין או ביום יום.  
חיפושים מתקדמים הם בעיה באתר הזה.

Query2:  What slang words from the early 2000s are still commonly used today?  
תוצאה: מצטערים, לא מצאנו: אילו מילים סלנג משנות ה-2000 עדיין בשימוש נפוץ היום?  
מידע נדרש:  
רשימה של מילות סלנג פופולריות משנות ה-2000 עם סטטיסטיקות שימוש עדכניות.  
דוגמאות למילים אלו בתקשורת מודרנית או בשיחות.  
תובנות לגבי למה מילים מסוימות נמשכו בעוד אחרות נעלמו.

Query 3: newest slang  
תוצאה: הגדרה מילולית של “סלנג החדש ביותר”  
מידע נדרש:  
רשימה של מילים סלנג שהיו יצירתיות לאחרונה (בשנה האחרונה).  
הגדרות ודוגמאות כיצד הן משמשות בשיחות.

שאלה 3:

<https://colab.research.google.com/drive/1TeyT44cEsJmIm4XTSvmia1N69QX6pKg8?usp=sharing>

שאלה 4:

**Urban Dictionary API:**  
אנו שולפים נתונים מ-API של Urban Dictionary, ואז מפענחים את התוצאות ב-JSON על מנת להוסיף אותן ל- cache המקומי שלנו.

**JSON Caching:**  
אנו שומרים את ה-definitions שנשלפו בקובץ urban\_data.json על מנת למנוע קריאות חוזרות ל- API - ולזרז חיפושים עתידיים.

**splitting queries by "or":**  
הוספנו תכונה שבה המשתמש יכול לקבל definitions מרובות בשאילתה אחת על ידי הוספת "או" בין כל שאילתה, לדוגמה: harry potter "or" john snow .  
אנו נבצע חיפוש עבור "harry potter" ולאחר מכן נבצע חיפוש עבור "john snow". אם המשתמש לא הוסיף "או", נבצע חיפוש על כל השאילתה.

**Custom Ranking Logic:**  
במקום להשתמש בשיטות דירוג מסורתיות כמו TF-IDF, יישמנו נוסחת דירוג מותאמת אישית:  
score=(likes−dislikes)−0.1×(current\_year−year\_of\_post)

**Search for New Slangs:**  
אנו יכולים להשתמש ב-API על מנת להחזיר 7 random definitions, ואז נשתמש ב-API עשרה פעמים כדי לקבל סך הכל 70 הגדרות, ולאחר מכן נוכל להחזיר את המונחים החדשים (שתי שנים אחרונות) עם הדירוג הגבוה ביותר למשתמש.

**Search using words.txt:**  
אם אנו רוצים לבצע חיפוש בכמות גדולה, אנו יוצרים קובץ בשם words.txt ואז בכל שורה כותבים query. - לאחר מכן הקוד יעבור על כל שורה ויביא definitions עבור כל query בקובץ.

שאלה 5:

זמן נמדד (25 שניות) לעיבוד כל השאילתות בקובץ words.txt.  
על מה זה תלוי: מספר מונחים: אם קובץ words.txt מכיל שורות רבות או תתי-שאילתות מופרדות ב-"or", כל אחת מהן חייבת להוציא מה-Urban Dictionary. הזמן גם תלוי בחיבור הרשת, כי כל בקשה ל-API של Urban Dictionary תלויה במהירות הרשת ובזמן התגובה של ה-API. לבסוף, זה גם תלוי בזיכרון מטמון: ברגע שמונח נמצא בזיכרון מטמון בקובץ urban\_data.json, בפעם הבאה שמישהו מחפש את המונח הזה, זה יהיה מיידי. אם אין נתונים במטמון, יותר קריאות יתקיימו ל-API.

האם אפשר לשפר את זה?  
כן, עם ריבוי תהליכים ב-Python ניתן להאיץ את הבקשות ל-API.  
בנוסף, שפה מהירה יותר כמו C++ עשויה לשפר את עיכובי עיבוד הטקסט, אם כי למשימות רשת, מקביליות בדרך כלל עוזרת יותר מאשר שינוי שפות.  
ולבסוף, הפחתת קריאות ה-API גם עוזרת (למשל, אם ה-API היה תומך בשאילתות מרובות מונחים בבקשה אחת), אך אנו זקוקים לבקשה מיוחדת מהאתר כי כרגע זה לא קיים.

שאלה 6:

use: appears in documents {'doc\_indices': [1, 6, 16, 26, 27, 29, 34, 37, 38, 39, 46, 48, 51, 53, 61, 65, 70, 74, 83, 91], 'original\_forms': ['using', 'uses', 'used', 'use'], 'frequency': 29}

hash: appears in documents {'doc\_indices': [31, 33, 34, 36, 37, 38, 39], 'original\_forms': ['hash', 'hashes'], 'frequency': 18}

someth: appears in documents {'doc\_indices': [0, 3, 4, 6, 7, 8, 26, 28, 39, 44, 46, 56, 89], 'original\_forms': ['something'], 'frequency': 17}

word: appears in documents {'doc\_indices': [1, 6, 25, 39, 44, 46, 85, 91, 94, 95], 'original\_forms': ['words', 'word'], 'frequency': 17}

one: appears in documents {'doc\_indices': [4, 5, 21, 24, 27, 34, 42, 74, 75, 81, 82, 85, 94], 'original\_forms': ['one', 'ones'], 'frequency': 16}

cannabi: appears in documents {'doc\_indices': [33, 36, 37], 'original\_forms': ['cannabis'], 'frequency': 15}

like: appears in documents {'doc\_indices': [1, 18, 25, 36, 52, 56, 57, 58, 91], 'original\_forms': ['like'], 'frequency': 14}

edg: appears in documents {'doc\_indices': [20, 26, 27, 29, 44], 'original\_forms': ['edge', 'edging', 'edges'], 'frequency': 13}

also: appears in documents {'doc\_indices': [8, 29, 33, 36, 56, 57, 91], 'original\_forms': ['also'], 'frequency': 12}

usual: appears in documents {'doc\_indices': [14, 29, 33, 34, 36, 75], 'original\_forms': ['usually'], 'frequency': 12}

mean: appears in documents {'doc\_indices': [8, 11, 22, 33, 39, 46, 55, 63, 79, 91], 'original\_forms': ['meanness', 'means', 'mean'], 'frequency': 11}

smoke: appears in documents {'doc\_indices': [27, 31, 32, 33, 36, 37], 'original\_forms': ['smoked', 'smoke', 'smokes', 'smoking'], 'frequency': 11}

plant: appears in documents {'doc\_indices': [30, 33, 34, 36, 37], 'original\_forms': ['plants', 'plant'], 'frequency': 11}

data: appears in documents {'doc\_indices': [34, 78, 79, 83, 85], 'original\_forms': ['dataful', 'data'], 'frequency': 11}

graph: appears in documents {'doc\_indices': [40, 41, 44, 45, 46, 48], 'original\_forms': ['graphing', 'graph', 'graphs'], 'frequency': 11}

שאלה 7:

=== Calculating TF-IDF for query: 'quantum' ===

TF-IDF Calculations:

term doc\_id tf df idf tf\_idf

0 quantum 0 1.000000 3 3.188417 3.188417

1 quantum 1 0.000000 3 3.188417 0.000000

2 quantum 2 2.098612 3 3.188417 6.691250

3 quantum 3 0.000000 3 3.188417 0.000000

4 quantum 4 0.000000 3 3.188417 0.000000

.. ... ... ... .. ... ...

92 quantum 92 0.000000 3 3.188417 0.000000

93 quantum 93 0.000000 3 3.188417 0.000000

94 quantum 94 0.000000 3 3.188417 0.000000

95 quantum 95 0.000000 3 3.188417 0.000000

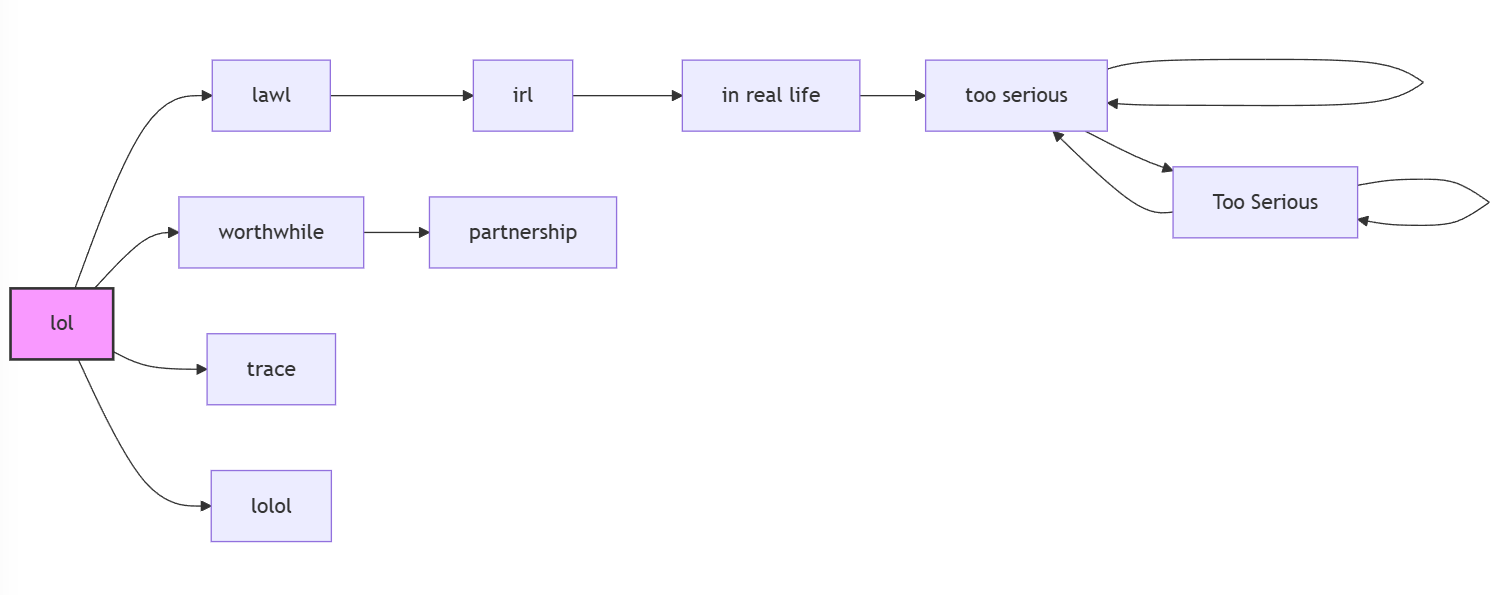
96 quantum 96 0.000000 3 3.188417 0.000000

שאלה 8:

האם יש Hubs / Authorities כאן?   
כן, בעיקרון, כל טקסט הגדרה יכול להכיל מספר הפניות [linked\_term] לדפים אחרים.  
אם הגדרה מסוימת מתייחסת להרבה מונחים אחרים, ההגדרה הזאת יכולה להיחשב כחיבור פוטנציאלי.  
הגדרה שאליה הרבה דפים אחרים מקשרים (מזכירים עם הפניות [סוגריים מרובעים]) יכולה להיחשב כרשות.  
דוגמה:  
אם “הארי פוטר (הגדרה הכי ישנה)” מתייחסת ל-[מכשף], [קסם], [הוגוורטס], זה hub פוטנציאלי.  
אם מספר הגדרות מ-“מכשף,” “קסם,” “הוגוורטס,” “וולדמורט” מקשרות חזרה לדף “הארי פוטר,” אז “הארי פוטר” עשוי להיות authority חזקה.

שאלה 9:

הערה חשובה היא שלא השתמשנו בהגדרות שהוחזרו לצורך חישוב ה-PageRank כי זה בלתי אפשרי ולא מתאים לדרישות. במקום זאת, השתמשנו במילה שהמשתמש ביקש לחפש כדי להרחיב למילים נוספות המקשרות בתוך התיאור של המילה שנחפשה.  
זו המבנה של המילים:

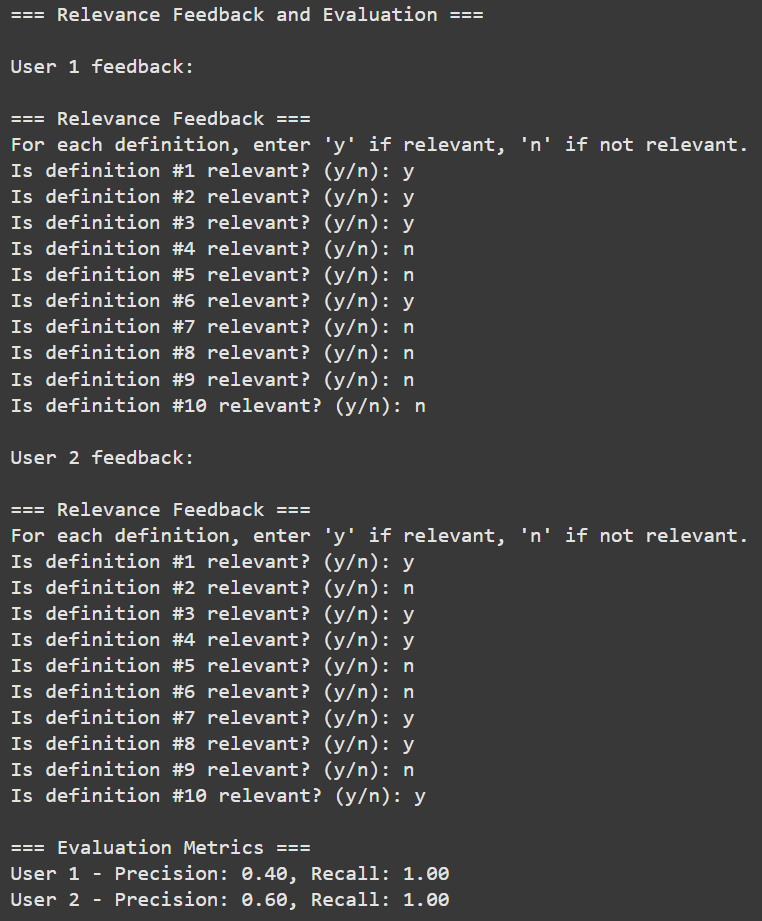


הסיבה שהמבנה נראה כך היא כי לקחנו מילה מהקלט של המשתמש שהיא “lol” והרחבנו אותה להגדרות שונות, לכן זה נראה כמו קו ישר. בנוסף, הגבלנו את הקוד לחפש רק 10 מילים שמקשרות אחת לשנייה, והנה ה-PageRank לאחר 3 איטרציות:

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

התיאור נוצר באופן אוטומטי

שאלה 10:



אנחנו לא יכולים לחשב Recall כי אנחנו משתמשים ב-API שמחזיר את כל ההגדרות הרלוונטיות, ולכן אנחנו מחזירים את כל ההגדרות, אך מציגים רק את 10 ההגדרות המובילות באמצעות מערכת הניקוד שלנו. למרות שאנחנו מציגים רק את 10 המובילות, בפועל אנחנו מחזירים את כל ההגדרות, ולכן זה 100%. בנוסף, אנחנו לא יכולים להציע שאילתה מתוקנת בגלל שהאתר שלנו הוא אתר מילון, ולכן אנחנו מניחים שהמשתמש הזין את השאילתה הנכונה ולא יכולים לשנות אותה.

החישובים נעשים באמצעות שתי המשוואות הבאות:  
Precision = relevant\_retrieved / retrieved\_doc  
Recall = relevant\_retrieved / relevant\_doc

כבר הסברנו שבחישוב ה-Recall, הערך של relevant\_retrieved שווה ל-relevant\_doc, ולכן אנחנו תמיד מקבלים 100%.  
לעומת זאת, בחישוב ה-Precision, אנו מחשיבים מסמכים רלוונטיים ככאלה שסומנו ב-'y' כאשר מקבלים משוב מהמשתמש – כפי שמוצג בתמונה. ה-retrieved\_doc הם המסמכים שהתקבלו כתוצאה מה-API שלנו, ולכן מתקבלים התוצאות שמופיעות למעלה.

שאלה 11:

[**ללחוץ כאן**](https://information-retrieval-course-kdnz-final-presentation.vercel.app/)

שאלה 12:

השתמשנו בהצעות שנכתבו תחת השאלה: "מה לדעתי אפשר לשפר באחזור" (כתבנו רק הצעות נכונות, כי היו תשובות של "אין הצעה")

|  |  |
| --- | --- |
| **איזה שינוי הוצע?** | **האם לדעתכם יש מקום לשיפור זה? אם כן - הסבירו כיצד ניתן לשפר. אם לא, נמקו** |
| האם זה בסדר שכל פעם אחרי הקאשינג נחזיר למשתמש אותם תשובות? | בעיקרון לא, כי אם המשתמש יישאר מחובר מלא זמן אז התשובות לא עדכניות, אך בגלל שאנחנו יודעים שהמשתמש לא יישאר מלא זמן אז זה לא פוגע באיכות התשובות. |
| אופציה לחיפוש אפשרויות לוגיות נוספות | לא צריך לעשות, כי AND זה ממומש, ו- NOT זה לא עוזר לנו באתר שלנו, אף אחד לא יחפש מושג מהסוג "not lmao". |
| הייתי אוילי מסה לשלב את שיטת חישוב הסקור שהם יצרו בנוסף לTFIDF | TFIDF פשוט לא עובד אי אפשר לשלב בכלל בפרויקט שלנו. |
| התמודדות עם שגיאות | קיים, אם מישהו מחפש כלום, או אם אין תשובה מהאתר. |
| גופן יותר ברור בפלט, היה קשה לראות | זו בעיה מה COLAB אין לנו מלא אופציות. |
| מנגנון ה CACHING נראה שמנוהל בקובץ טקסט, אפשר לנהל אותו באופן יעיל יותר | כן אפשר לשפר, אפשר לעשות FIREBASE ולשמור את הנתונים שם, אך יש בעיה של לדעת מתי למחוק את ה CACHE בשביל שזה לא יפגע בתוצאות. |
| אולי לראות איך כן ניתן ליצור מצב שבו חיפוש water או h2o יתקבלו ביחד שכן h2o זה סה"כ הסימול הכימי למים | פה מדובר ב TFIDF, להראות דוגמא, ולא שיפור לקוד. |
| שימוש בעוד כלים | הצעה לא מדויקת, אין תגובה. |
| מספר המילים מצומצם מדי | אנחנו מניחים שמדובר במספר המילים ששולחים ל API, זה לפי מה שהמשתמש הכניס, אפשר בקובץ words.txt לשים מאות מילים שאפשר לחפש. |
| לא היה שימוש בTF-IDF | TFIDF פוגע בצורה ישירה באיכות המידע שמוחזר, כי בהסבר של מונח הופעת המונח עצמו מראה שההסבר לא טוב. |
| לפי דעתי כדאי להוסיף ממשק משתמש שמאפשר להזין מספר שאילתות שירוצו במקום לעשות זאת באמצעות קובץ טקסט | יש דרך להזין שאילתות אך אצלנו אפשר רק שאילתה אחת להזין, אך אפשר לשפר על ידי הפיכת ההזנה ללולאה אין סופית עד שהמשתמש מחפש כלום. |
| לא להשתמש בAPI | רק באמצעות API אנחנו יכולים לקבל את מספר הלייקים והדיסלייקים, חייבים להשתמש בAPI. |
| לתת לאתר לבצע חיפוש בשאילתות יותר ארוכות | זה אפשרי בקוד שלנו, פשוט הדוגמאות שכתבנו היו קצרות למען דוגמא בלבד, אפשר להזין שאילתה:  " Collision Courses: The Juvenile Release" ולקבל תשובה יחידה: " What I call homo-sapiens who are addicted to perianal abscesses.". |
| שיפור אפשרי באחזור יכול להיות על ידי אופטימיזציה של השאילתות למנוע תוצאות יותר מדויקות. | אי אפשר לשחק בשאילתות של המשתמש כי אנחנו חייבים להניח שהוא מחפש את מה שהוא רשם בדיוק. |
| שיפור של שאילתות שלא נשאלו בעבר אולי ניתן לשפר את זמן הריצה של השאילתה . | ההצעה לא ברורה ולא מדויקת אז אין תגובה. |
| ניתן להוסיף לדירוג התחשבות בנתונים הקשורים למבנה קישורים בין הדפים. | פה מדובר על שימוש ב PAGERANK שזה גם לא כל כך עוזר, כי זה לא מעניין אם מונח מקושר למונח אחר בזמן שהמשתמש מחפש מונחים, המשתמש רוצה לראות רק מונחים שהוא רוצה. |
| אפשרות לחפש יותר שאילתה משאילתה אחת בכל פעם | אפשרי דרך words.txt. |
| שיפור הערכת מערכת האחזור | ההצעה לא ברורה ולא מדויקת אז אין תגובה. |
| להשתמש במדדים להערכת שאילתות ובמדד TFidf. | TFIDF לא עובד עם האתר שלנו, אז לא נשתמש ב TFIDF. |
| דיוק באחזור מידע ומענה למקרים שוליים. | אנחנו משתמשים ב API אז התוצאות מדויקות לרמת דיוק האתר עצמו, ומקרים שוליים מכוסים. |

שאלה 13:

אנחנו יכולים לכתבות את האתגרים בכמה נקודות:

1. קושי עם העבודה התכנית (קוד).
2. עבודה עם אתר שיש לו מלא בעיות, כמו קושי עם TFIDF, PAGERANK.
3. מציאת זמן להיפגש, כל אחד היה לו לוז שונה מהאחר.
4. מציאת דרך לקבל LIKES, DISLIKES וזה נפתר על ידי API.
5. המושגים של הקורס היה קשוחים אך הפעילות בכיתה עזרו.
6. אי הסכמה על תשובה לשאלה, כל אחד היה לו רעיון אחר.

אילו היו האתגרים המרכזיים בזמן העבודה בקורס, אך התגברנו על הכול ואנחנו גאים בזה שלמדנו מלא מהקורס הזה.