דוח מנוע חיפוש חלק ב'

1. פירוט השיטות:

1. Advanced Parser:

בשיטה זו החלטנו לשפר את הביצועים של הפרסר על מנת לתפוס את המילים החשובות ביותר והרלוונטיות ביותר של המסמך. שלב הParser- מבצע בדיקה ועמידה בחוקים הדרושים, פירוק הטקסט לterms כנדרש כאשר ביצוע הפעולות הנ"ל מתבצעות בעת קריאה לפונקציה המתאימה במחלקה, המחלקה יוצרת ומחזירה אובייקט מסוג document לכל מסמך שעבר תהליך parse.

המחלקה מתחלקת לשלושה חלקים עיקריים –

* ביצוע Parse למסמך כולו, הפונקציה הראשית שאחראית לקריאת לפונקציות המבצעות את הParse לחלקים ספציפיים במסמך. *(Parse\_doc())*
* ביצוע Parse למשפט יחיד.*(Parse\_setence())*
* ביצוע Parse לurl.*(Parse\_url())*

נתונים שהרצת הפרסר יוצרת שעוזרים לנו בהמשך:

*Dict\_upper\_word* – מילון שמכיל את כל המילים שהם באותיות גדולות וכמות הפעמים שהופיעו בטקסט.

*Entity\_dict* – מילון המכיל את כל הישויות העשויות להיות בטקסט וכמות הפעמים שהופיעו בטקסט.

*Retweet\_dict* – מילון שמכיל את כל הציוצים שעשו להם שיתוף.

Tweet\_Date - שומר את התאריך של יצירת הציוץ.

נסביר על שתי שיטות עיקריות במחלקה ועל הכללים והחוקים התבצעים בהם:

*Parse\_setence(text)* – השיטה מקבלת את כל הטקסט שעליה לבצע parse.

הפונקציה מחזירה את הנתונים הבאים:

*Final\_term* – מילון מונחים סופי לאחר שעברנו על כל המילים בטקסט וביצענו עליהם את כלל החוקים וכמות הפעמים שהופיע המילה בטקסט.

*Entity\_dict* – מילון מונחים סופיים של כלל המונחים העשויים להיות ישויות וכמות הפעמים שהופיעו בטקסט.

אופן הפעולה של הפונקציה:

הפונקציה מבצעת מחיקה של התווים הבאים [...,\n,-,,] ואז מבצעת split לפי רווחים ומחזירה מערך עם המחרוזות המופרדות.

לאחר מכן הפונקציה עוברת בלולאה על כל המחרוזות במערך ומטפלת בכל אחת לפי חלוקה למקרים וביצוע מספר בדיקות:

תחילה נבדוק האם המילה מכילה אותיות ותווים מיוחדים שעלולים לגרום לנו לבעיות בהמשך התהליך.

לאחר מכן נוודא המילה אינה ריקה ונתחיל בתהליך החוקים.

1. אם המילה מכילה את הסימן '-', '/', '\_' :

נבצע הפרדה של המילים לפי התו הנ"ל ונכניס אותן לסוף רשימת המילים המיועדות לבדיקה.

2. אם המילה מכילה את הסימן '%':

נשלח את המילה לפונקציית עזר שתבדוק האם זהו מספר שהוא אחוז ותנקה אותו מסימנים מיותרים ותכניס אותו לרשימת המילים הסופיות.

3. אם המילה מתחילה או מסתיימת עם הסימן $:

נשלח לפונקציית עזר שתוריד סימנים מיותרים ותבדוק האם זה באמת מספר, על מנת שנוכל להתייחס למונח זה כסכום כסף.

4. האם המילה היא מספר:

בפונקציה זו אנחנו מטפלים בכמה אפשרויות:

* מספר רגיל שאין לו יחידות – נכניס אותו לפונקציית עזר הבודקת את היחידות המתאימות לפי גודל המספר ומבצעת את ההתאמות לפי שכתוב בהנחיות העבודה.

במידה ונקבל 1000000 נחזיר 1M.

* מספר עם מילת יחידות אחריו – נבדוק האם המילה שמגיעה אחרי המספר היא מילת יחידת ונחזיר אותם מחוברים לדוגמא: 1milliom נחזיר 1M.
* מילה עם מילה שמסמנת לנו אחוז- נבדוק אם המילה היא חלק ממילים שיכולות להיות אחוז ונחזיר את המספר עם הסימון %.

5. האם המילה היא url – נבדוק האם המילה מתחילה עם http ובמידה וכן נשמור אותה על מנת לבצע עליה parse בהתאם בחוקים של url.

במידה והמונח לא נכנס לאף קטגוריה למעלה נכניס אותו לסט בדיקות של מונחים שבטוח מכילים מילים.

תחילה נבצע נקיון על המילה שמכילה סימנים שמוגדרים ברשימת סימנים למחיקה.

6. האם המילה מתחילה ב@:

נשמור את המונח כמו שהוא ונכניס אותו אל הרשימה.

7. האם המילה מתחילה ב#:

נשלח לפונקציה עזר שמחלקת את ההאשטאג למילים לפי מילים אותיות קטנות גדולות, לפי סימנים מיוחדים, ולפי הכתוב בהוראות במידה וקיים '\_', בנוסף, לכל מילה שהפרדנו נהפוך את כולה לאותיות קטנות ונבצע עליה stemmer במידת הצורך.

לאחר מכן נבצע הפרדה האם המילה מורכבת ממספר אותיות או שמכילה רק שתי אותיות, במידה והיא אות יחידה אנחנו נסיר אותה מכיוון שזה יכול להיות טעות או מילים שפחות יהוו לנו משמעות בשאילתה.

בסט בדיקות זה לפני שנכניס את המילה למילון נבדוק שהיא אינה מופיעה ברשימת המילים stop words ונבצע עליה stemmer במידת הצורך.

8. נבדוק האות ההתחלתית היא אות גדולה:

נכניס את המילה למילון בצורה כך שכל המילה תהיה באותיות גדולות. (את הבדיקה של אותיות קטנות/גדולות נבצע ב-indexer על מנת לחסוך זמן ריצה וכדי שנוכל לבדוק את המילים מול הקורפוס כולו)

9. בדיקת ישויות:

כאשר מתגלה מילה עם אות גדולה נכניס אותה לרשימת ישויות אופציונלית, ונשמור את מיקום המילה במסמך. כאשר נמצא מילה נוספת שהיא עם אות גדולה נבדוק האם המיקומים עוקבים. במידה וכן, נתייחס למונח כישות ונכניס אותו לרשימת הישויות האופציונלית גם כן.

במידה והמילה העוקבת אינה אות גדולה, נבדוק כמה מילים יש לנו ברשימת הישויות האופציונלית. במידה וזה יותר ממילה אחת נחבר בין המילים לישות אחת ונכניס אותה למילות ישויות של המסמך שבו נספור גם את כמות המופעים של הישויות במסמך.

במידה וזו מילה יחידה עם אות גדולה לא נתייחס אליה כישות.

(בדומה לאותיות גדולות קטנות נבצע בדיקת ישויות נוספות מול הקורפוס כולו בשלב הindexer).

כל מילה אחרת תיכנס למילון כמילה באותיות קטנות.

את שלב זה נבצע על שני ערכים שקיימים לנו בנתונים על המידע:

1. הציוץ המלא – כל התוכן המילולי שקיבלנו מהציוץ.

2. Retweet – במידה והציוץ משותף נבדוק האם יש טקסט שנוסף לשיתוף ונבצע עליו את שלב הפרסור על מנת שנוכל להרחיב את המידע על תוכן הציוץ.

Parse\_url:

תחילה נבצע בדיקה האם קיים לנו url בתא המתאים שקיבלנו מכל המידע שקראנו על

המסמך. במידה וקיים לנו url מלא, נבצע עליו את ריצת הparse. אחרת, אם קיים לנו נתיב

מקוצר נשלח אותו. במידה ולא קיים לנו נתיב בכלל לא נבצע תהליך parse זה למסמך.

url\_handler(val,tweet\_id) – פונקציה זו מקבלת נתיב ואת המספר הייחודי של הציוץ.

תחילה היא מבצעת split לפי הסימנים [/@!#$)(,~?%\*&:;]. לאחר מכן נעבור על כל מונח שהתקבל מהפירוק ונכניס אותו לעץ אפשריות על מנת לבדוק אותו לפי הכללים והחוקים שנגדיר.

1. במידה המילה מכילה '-':

נבצע split לפי התו הנ"ל ונכניס כל אחד מהמילים שהתקבלו מהפירוק את רשימת המילים הממתינים לבדיקה.

2. נבדוק שאורכו של המונח הוא גדול מתו אחד:

בדומה בריצת הparse שנעשה על הטקסט מילים שהם בגודל של תו אחד אינם מביאים לנו משמעות במסמך.

3. במידה והמונח מתחיל ב'www':

נפריד את המונח לשני חלקים ונשמור רק את החלק של הדומיין. את המונח www החלטנו כי אין לנו צורך לשמור מכיוון שהוא לא מהווה לנו מידע רלוונטי בעת החיפוש.

4. במידה והמונח שווה למספר הייחודי של הציוץ:

החלטנו כי לשמור את המספר הייחודי של הציוץ אינו מהווה לנו מידע שימושי, והנתיב היא רק קישור אל הציוץ עצמו ברשת ולכן החלטנו שלא להוסיף אותו לרשימת המונחים שלנו.

5. רשימת stop words:

החלטנו שעל מנת לייעל את תהליך הparse לא נוסיף מילים אל רשימת הstop word הקיימת אלא נבנה אחת משלנו שתכיל את כל המילים שאנחנו לא רוצים לשמור ברשימת המונחים הסופית.

נבדוק האם המילה לא נמצאת ברשימה הנ"ל ונוסיף אותה לרשימת המונחים הסופית של המסמך.

פונקציות נוספות שמהוות חשיבות רבה במחלקה זו:

Popularity\_retweet(url\_list,id\_tweet) – ציוצים רבים עושים שיתוף לציוצים אחרים וראינו שניתן להשתמש במידע זה ולקבל ממנו ערך חשוב שמייצג את פופולריות המסמך.

הגדרת הפופולריות – ככל שיותר מסמכים עשו למסמך i שיתוף ככה הפופולריות של מסמך i עולה.

פונקציה זו עוברת על המידע שקיבלנו על המסמך ובודקת האם קיימים הנתיבים שמייצגים את הציוצים להם הם עשו שיתוף.

ניקח את הנתיבים הנ"ל ונחלץ מהם את המספר הייחודי של הציוץ שאותו שיתפו. נשמור את רשימת הציוצים ובהמשך בתהליך הindexer נבצע בדיקה כמה פופולרי כל מסמך.

Tweet\_date\_parse(tweet\_date) – ראינו לנכון לשמור את תאריך הפרסום של הציוץ על מנת שיוכל לעזור לנו בתהליכים מתקדמים. לכן לקחנו את המידע שקיבלנו כאשר קראנו את המסמך, בודדנו את תאריך הפרסום ורשמנו אותו בתבנית בה יהיה לנו נוח להשתמש. (xx/xx/xxxx)

1. WordNet-

שיטה זו מבצעת הרחבה של השאילתה ומוסיפה מילים בעלות משמעות דומה למילים בנמצאות בשאילתה. שיטה זו עוזרת לנו לאפשר למצוא מסמכים שיהיו רלוונטים בצורה הגבוה ביותר בתוכנם ובמשמעותם.

על מנת לממש שיטה זו יצרנו מחלקת WordNet שתהיה אחראית על כך, השתמשנו בספריית WordNet של NLTK.

אופן הפעולה של השיטה:

השיטה מקבלת רשימה של term שמייצגים את השאילתה לאחר ביצוע Pasre.

נעבור על כל מילה ברשימה ונבצע עליה את הפונקציה Synsets שמביאה לנו משמעויות שונות של המילה אותה הכנסו. לפי בדיקה מקיפה של המילים המוחזרות ראינו כי המילים הראשונות המוחזרות הן המילים שמייצגות את המונח בצורה הטובה ביותר והכי דומות למילה שהוכנסה.

בנוסף, ערכנו מספר בדיקות לגבי כמות המילים שניקח מכל מילה שנבצע עליה הרחבה והוחלט כי המספר שעוזר לקבל את השאילתות הרלוונטיות ביותר הוא 2.

לאחר שנרחיב את השאילתה, נמשיך בביצוע החיפוש והדירוג הרגיל שלנו.

2. שינויים והוספות ל-Parser:

הוספות –

תוכן מציוצים משותפים- ראינו כי במידה והציוץ ביצע שיתוף ניראה כי קיים תוכן נוסף במידע שאנחנו מקבלים. לכן בדקנו האם קיים תוכן מילולי שמייצג לנו את הציוץ ששותף ולאחר ביצוע פרסור הוספנו את ה-terms שהתקבלו לרשימת המילים שמוחזרת. ראינו כי הוספה זו מאפשר לנו לקבל קשת רחבה יותר של מילים שמתארות את המסמך וכך נוכל לדייק במשמעות תוכן המסמכים לשאילתה.

מילה בעלת 2 אותיות- ראינו כי הרבה מילים בעלות 2 מילים אינן מספקות לנו מידע רלוונטי על המסמך ולכן החלטנו כי מילים בעלות 2 אותיות לא נכניס למילון שלנו על מנת שהמילון יכיל את המילים המשמעותיות ביותר.

הורדות-

Stemmer- החלטנו כי לא נבצע Stemmer על הביטויים המתקבלים. פעולה זו לוקחת הרבה זמן ומאטה את קצב הביצועים ובנוסף, ראינו כי לפי המדדים שנבדקו המסמכים המוחזרים לשאילתה לאחר ביצוע ה-Stemmer פחות רלוונטיים לשאילתה מאשר אם כן נבצע Stemmer.

Lemmatize- החלטנו לאחר כמה ניסיונות כי על כל מונח שהוא מסוג מילה נבצע עליו lemmatizing וכך נדאג שמילים שנכנסות למיון יהיו בצורתן השורשית ככל שאפשר. כבר זה מדייק את תוצאות השאילתות.

3. כמות מילים:

כמות המילים שקיבלנו לאחר שהרצנו את הקורפוס benchmark\_data – 15,535

4. קבצי Index:

בחלק זה החלטנו להיעזר בארבעה קבצי אינדקס שמכילים את כל המידע שאנחנו נעזרים על מנת לאחר את המידע הטוב ביותר.

קבצי Invertad-index: מילון המונחים. במילון נמצאים כל המילים וכמות המופעים שלהם במילון.

במנה של מילון זה מאפשר לנו גישה מהירה אל כל מילה בקורפוס. מידע זה עוזר לנו בדירוג המסמכים.



term

frequency

קבצי הposting:

נשמור מילון שהמפתח שלו הוא הterm עצמו והערך שלו הוא List שמכיל Nodes שבנוי מאובייקט מסוג tuple ובו שתי שדות tweet ID וכמות הפעמים שמילה הופיעה במסמך. מבנה זה השתנה מהחלק הקודם מכיוון שהוגדר שכל המילון נשמר בזיכרון החלטנו לשמור מילון שהמפתח שלו הוא ה-term והערך הוא רשימה של מספרי המסמכים שבהם המילה מופיעה וכמות המופעים שלה במסמך. מימוש זה מאפשר גישה מהירה למידע ואחידות בקוד.

קובץ posting ניראה כך:



Tweet ID

Term

Frequency

קבצי ה-docs:

עבור כל ציוץ נשמור מילון שהמפתח שלו הוא tweet\_ID והערך שלו הוא tuple שמכיל את השדות הבאים:

§       תאריך הציוץ (Date) - במידה ונרצה לאחזר מסמכים לפי תאריך הפרסום שלהם.

§       מספר המילים הייחודיות במסמך (Unique) – לפי בקשת ההוראות.

§       מספר המופעים של המונח הנפוץ ביותר במסמך (Max tf) - לחישוב ה Inner product.

נשתמש בקבצים אלו על מנת לחשב את דירוג המסמך בשיטת Cosimularity, ובנוסף על מנת שנוכל לאפשר למשתמש לבחור למיין את המסמכים הרלוונטיים לפי תאריך פרסום הציוץ.

קבצי ה-docs נראים כך:



Tweet ID

Max tf

Unique

Date

Date

:Popularity dict ציוצים רבים עושים שיתוף לציוצים אחרים וראינו שניתן להשתמש במידע זה ולקבל ממנו ערך חשוב שמייצג את פופולריות המסמך מידע זה נשמר Popularity dict כאשר הKey הוא tweed id וה value הוא ערך מספרי שמייצג לנו כמה טוויטים שיתפו את הטוויט הנוכחי .

הגדרת הפופולריות – ככל שיותר מסמכים עשו למסמך i שיתוף ככה הפופולריות של מסמך i עולה.

פונקציה זו עוברת על המידע שקיבלנו על המסמך ובודקת האם קיימים הנתיבים שמייצגים את הציוצים להם הם עשו שיתוף.

ניקח את הנתיבים הנ"ל ונחלץ מהם את המספר הייחודי של הציוץ שאותו שיתפו. נשמור את רשימת הציוצים ובהמשך בתהליך הindexer נבצע בדיקה כמה פופולרי כל מסמך.

קבצי Popularity ניראה כך:



Tweet ID

Popularity

5. ייעול זמני ריצה:

אותיות גדולות קטנות – על מנת לייעל את זמני הריצה שלנו החלטנו להוריד את הבדיקה של אותיות גדולות קטנות מהחלק של הפרסר ולבצע אותו רק באינדקסר. דבר זה בודק את המילים מול הקורפוס כולו ואנחנו נמנעים מכפילות של בדיקה.

צמצום מילים בקורפוס - מכיוון שהוגבלנו בחלק זה לכמות מילים במילון (כ-100,000 מילים) ביצענו מספר התאמות של ה-Indexer על מנת לבצע זאת בכל קורפוס שנקבל. אחד הדברים שלקחנו בחשבון זה התיאוריה של Zip low שאומר מילים נפוצות יותר יהוו חלק גדול יותר מהקורפוס.

במהלך אינדוקס המסמכים עקבנו אחרי כמות המילים שקיימים לנו במילון, במידה והגענו לכמות של 125,000 מילים. בחרנו את כמות מילים זו מכיוון שראינו שכך ה-Indexer עובד בצורה היעילה ביותר וכך אנחנו דואגים לא למחוק מילים רלוונטיות מהמילון. בחרנו k=4 התחלתי שמילים בעלות כמות מופעים קטנה ממנה אנחנו מוחקים מהמילים. מכיוון שבקורפוס גדול כמות המילים שמתווספות גדלה בכל פעם ב-N ולכן ניראה להגדיל את k בצורה אקספוננציאלית על מנת ליעל את המנוע ולא לבצע כמות רבה של מחיקות תוך כדי ריצה. הדבר המייעל את זמני הריצה שלנו משמעותית בעבודה.

לאחר שסיימנו לעבור על כל המסמכים נרצה להקטין את המילון ל100,000 ולכן נבצע עוד פעולת מחיקה נוספת.

6. שיפור איכות התוצאות:

על מנת ליצור שיפור בתוצאות החלטנו כי השיטה הטובה ביותר היא שיפור הפרסר יחד עם השיטה של wordnet. נרחיב את השאילתה למילים בעלות משמעות דומה וככה נוכל לקבל קשת רחבה יותר של מסמכים שהמשמעות שלהם תהיה מתאימה לשאילתה שקיבלנו.

Retweet- במידה וניראה כי מתקיימת שיתוף לציוץ החלטנו לבדוק את המידע על השיתוף ולהוסיף אותו אל רשימת ה-terms הסופית. וכך אם קיימים מילים נוספות שיכולות לתאר את המסמך הוא קיים לנו במידע על המסמך

7. איכות השיטות:

על מנת להחליט איזה מהשיטות יהיו הטובות ביותר להחזרת השאילתה יצרנו לכל שיטה מחלקת search\_engin משלה ומחלקת searcher אחת שתבצע הרחבה לשאילתה והשנייה לא תבצע. את כל אחת מהשיטות הרצנו בנפרד והסתכלנו על הממדים שבדקנו. (בהמשך נרחיב עליהם) לפי התוצאות שקיבלנו בכל אחד מהממדים החלטנו כי השיטה הטובה ביותר היא חיבור של שתי השיטות ביחד. על מנת לוודא זאת נכנסנו אל התוצאות ובדקנו את המסמכים שהוחזרו.

8. קוד פתוח:

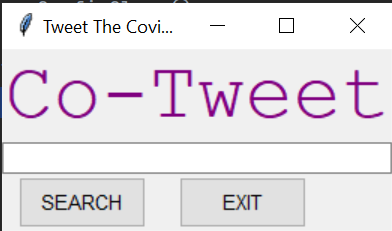
השתמשנו בספריית WordNet של NLTK על מנת לבצע את הרחבת השאילתה.

השתמשנו בספריית Lemmatizer של NLTK על מנת להפוך את המילה לצורתה השורשית.

9. מידע נוסף:

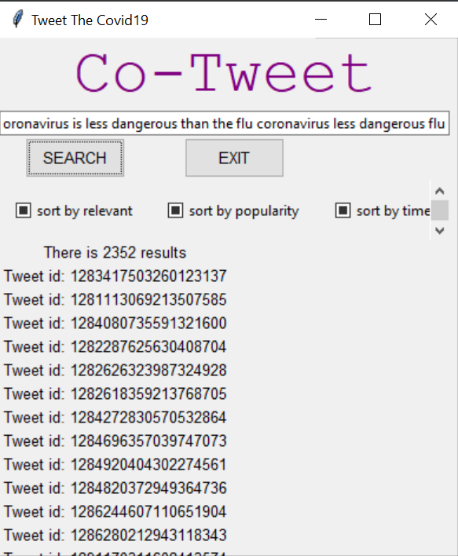
החלטנו בחלק זה להוסיף חווית משתמש ולכן החלטנו ליצור ממשק לתוכנית שבה המשתמש יוכל לבצע חיפוש לשאילתה שלו ולקבל את מספרי המסמכים הרלוונטיים בצורה נוחה ומהירה.

להלן תמונה להמחשה של הממשק לפני הכנסת השאילתה.



לאחר שהמשתמש מבצע חיפוש כמות המסמכים הרלוונטיים ומספרי המסמכים יוצגו למסמכים. המסמכים יוצגו בצורה ממוינת לפי הרלוונטיות שנתנה למסמך (לפי שיטת דירוג Inner Proudct)

בנוסף, המשתמש יוכל לבצע מיון לפי המדדים הבאים: זמן, פופולריות, לחזור למצב הרגיל (רלוונטיות)

להלן תמונה להמחשה של הצגת התוצאות למשתמש.

במידע שאספנו במהלך פרסור המסמך עזר לנו להביא למשתמש מידע נוסף שבעזרתו הוא יוכל למיין את תוצאות השאילתה שקיבל. נעזרנו במידע הפופולריות שניתחנו כל מסמך ועל תאריך יצירת הציוץ.

10. תשובות לשאילתות:

1. השאילתה : **Dr. Anthony Fauci wrote in a 2005 paper published in Virology Journal that hydroxychloroquine was effective in treating SARS. fauci paper hydroxychloroquine sars**

5 הציוצים הראשונים שהתקבלו :

1. (מספר מסמך : 1285169749333356544 ) הציוץ מדבר על אותו נושא כמו בשאילתה עם הרבה מילים משותפות ובנוסף יש דמיון בין משמעות השאילתה למסמך. בשניהם יש התייחסות על המחקר של פרופסור פוזי שפורסם במגזין virology שאומר שהטיפול של סארס יכול להיות מתאים גם לקורונה.
2. (מספר מסמך: 1284229606179053568) הנושאים דומים ויש הרבה מילים משותפות. אותו מדען שטוען שיש לו את הטיפול לקורונה ולא הקשיבו לו. כמו בשאילתה נאמר שהטיפול של הסארס מתאים גם לקורונה.
3. (מספר המסמך: 1291138937017307136 ) ניתן לראות שהציוץ הוא שאלה שהשאילתה עונה עליה. ולכן המשמעות היא דומה אך בצורה של שאלה.
4. (מספר מסמך: 1291767378896920577) יש התייחסות לאותו נושא, המדען. ולכן יש מילים דומות בין בשאילתה למסמך.שניהם מדברים על אותו מדען שפרסם במגזין virology .
5. (מספר מסמך: 1288845290926215174) מדבר על אותו נושא, על המדען ועל החומר שהוא חוקר.

2. השאילתה : **The seasonal flu kills more people every year in the U.S. than COVID-19 has to date. flu kills more than covid**

5 הציוצים הראשונים שהתקבלו :

1. (מספר מסמך : 1291791311998029824) שני הציוצים מדברים על נושא השפעת וכמה היא הורגת אנשים לעומת הקורונה. קיימות מילים דומות רבות בין המסמך לשאילתה
2. (מספר מסמך: 1291802650980343809) שני הציוצים מדברים על זה נושא השפעת וכמה היא הורגת אנשים לעומת הקורונה. קיימות מילים דומות רבות בין המסמך לשאילתה
3. (מספר המסמך: 1291786635097657344) שני הציוצים מדברים על זה נושא השפעת וכמה היא הורגת אנשים לעומת הקורונה. קיימות מילים דומות רבות בין המסמך לשאילתה
4. (מספר מסמך: 1291742415158571009) חלק מהשאילתה מופיעה בתוך המסמך ולכן יש התאמה ביניהם.
5. (מספר מסמך: 1283530163033124870) חלק מהשאילתה נמצאת בתוך במסמך עצמו, ויש משמעות בין המשמעות של השאילתה למשמעות המסמך.

4. השאילתה : **The coronavirus pandemic is a cover for a plan to implant trackable microchips and that the Microsoft co-founder Bill Gates is behind it gates implant microchips**

5 הציוצים הראשונים שהתקבלו :

1. (מספר מסמך : 1283662288654917632 ) הציוץ אינו מכיל תוכן טקסט נוסף אלא משתף ציוץ אחר, המקרה המתואר בציוץ שהוא שיתף מתאים לשאילתה בצורה מדויקת. נאמר שם כי מגפת הקורונה היא תוכנית כיסוי כדי "לשתול" צ'יפ שיאפשרו מעקב על בבני אדם. הציוץ והשאילתה מכילים הרבה מאוד מונחים משותפים.

2. (מספר מסמך : 1283579019154399232) המסמך שהוחזר הוא דומה למסמך הראשון.

3. (מספר מסמך : 1283609352755007489 ) המסמך שהוחזר הוא דומה למסמך הראשון.

4. (מספר מסמך : 1283586103983910912 ) המסמך שהוחזר הוא דומה למסמך הראשון.

5. (מספר מסמך : 1287340654813741056) קיימת התייחסות ספציפית ל Bill Gates - מדובר על זה שחברה דרום קוראנית שמגובית על ידי bil gates עלולה להצליח לייצר 200 מליון חיסונים עד יוני.

7. השאילתה : **Herd immunity has been reached. Herd immunity reached**

5 הציוצים הראשונים שהתקבלו :

1. (מספר מסמך : 1281134802419306502) המסמך מכיל את המילים שיש בשאילתה, המסמך מתייחס לנושא.
2. (מספר מסמך: 1281526339313577985) המסמך מכיל את השאילתה ומראה שאלות רלוונטיות שאנשים שאלו עם נושא השאילתה הזו.
3. (מספר המסמך: 1282076498271363075) המסמך מכיל את השאילתה ומראה שאלות רלוונטיות שאנשים שאלו עם נושא השאילתה הזו.
4. (מספר מסמך: 1282357711984889856) המסמך מכיל את נושא השאילתה ומראה ציוץ שעונה לגבי הנושא של השאילתה.
5. (מספר מסמך: 1284600321549967361) המסמך מכיל את נושא השאילתה, ומדבר עליה בצורה חיובית.

8. השאילתה : **Children are “almost immune from this disease.” children immune to coronavirus**

5 הציוצים הראשונים שהתקבלו :

1. (מספר מסמך : 1291151835554811906 ) הציוץ אינו מכיל תוכן טקסט נוסף אלא משתף ציוץ אחר, המקרה המתואר בציוץ שהוא שיתף מתאים לשאילתה בצורה מדויקת. בציוץ המשותף מדובר על trump ועל טענתו ש"הילדים" כמעט חסינים מהקורונה וירוס.

2. (מספר מסמך : 1291157420518051842) המסמך שהוחזר הוא דומה למסמך הראשון.

3. (מספר מסמך : 1291146248779698177) המסמך שהוחזר הוא דומה למסמך הראשון.

4. (מספר מסמך : 1291161086922817544) המסמך שהוחזר הוא דומה למסמך הראשון.

5. (מספר מסמך : 1291146405667434498) המסמך שהוחזר הוא דומה למסמך הראשון.

11. תשובות למדדים:

**WordNet with Advance Parser**

ערך מינימאלי – 0.096525

ערך מקסימאלי – 0.9853

ממוצע – 0.6053

חציון – 0.6384

**Advanced Parser**

ערך מינימאלי – 0.096525

ערך מקסימאלי – 0.9853

ממוצע – 0.6053

חציון – 0.6384

**WordNet with Advance Parser**

ערך מינימאלי – 0.2

ערך מקסימאלי – 1

ממוצע – 0.822

חציון – 0.8

**Advanced Parser**

ערך מינימאלי – 0.2

ערך מקסימאלי – 1

ממוצע – 0.822

חציון – 0.8

**Advanced Parser**

ערך מינימאלי – 0.1

ערך מקסימאלי – 1.0

ממוצע – 0.8

חציון – 0.9

**WordNet with Advance Parser**

ערך מינימאלי – 0.1

ערך מקסימאלי – 1.0

ממוצע – 0.797

חציון – 0.9

**WordNet with Advance Parser**

ערך מינימאלי – 0.16

ערך מקסימאלי – 1.0

ממוצע – 0.752

חציון – 0.84

**Advanced Parser**

ערך מינימאלי – 0.20

ערך מקסימאלי – 1.0

ממוצע – 0.74

חציון – 0.8

**WordNet with Advance Parser**

ערך מינימאלי – 0.8

ערך מקסימאלי – 1

ממוצע – 0.99

חציון – 1

**Advanced Parser**

ערך מינימאלי – 0.8

ערך מקסימאלי – 1

ממוצע – 0.99

חציון – 1

**WordNet with Advance Parser**

ערך מינימאלי – 0.3204

ערך מקסימאלי – 0.9958

ממוצע – 0.73693

חציון – 0.8095

**Advanced Parser**

ערך מינימאלי – 0.3127

ערך מקסימאלי – 0.9952

ממוצע – 0.7319

חציון – 0.8017

12. יתרונות וחסרונות:

יתרונות:

* GUI – החלטנו להקל על המשתמש וליצור ממשק שבו הוא יוכל להכניס את השאילתה ולקבל את מספרי הציוצים בצורה נוחה ומהירה. לאחר מכן המשתמש יוכל להחליט את סדר התצוגה שלו לפי אפשרויות מיון שונות. (פופולאריות, זמן יצירת הציוץ, רלוונטיות הציוץ)
* Advance Parser – החלטנו בחלק זה לקחת את הפרסר שלב קדימה והחלטנו לבצע פירסור על מידע רב שאנחנו מקבלים בכל מסמך. ככה אפשרנו לביצועים שלנו לעלות. דבר נוסף זה שהחלטנו להסיר את פעולה ה-Stemmer ולהוסיף את פעולת Lemmatize וכך להגיע לצורתו בשורשית של המילה וראינו שהתוצאות שמוחזרות מדויקות יותר.
* קיצור מילים בכמה פעמים – על מנת שלא ניצור עומס על זיכרון המחשב החלטנו שלאחר כמות מסוימת של מילים אנחנו נבצע מחיקה של מילים שמופיעות בכמות נמוכה. דבר זה עוזר לנו לשמור על מילון שהזיכרון יכול להחזיק. ולהוריד מילים שאין נפוצות במסמכים.

חסרונות:

* על מנת לחסוך בזמני ריצה החלטנו לבנות את רוב הפונקציונליות בעצמנו ולכן יכולים להיות דברים שהם פחות מדויקים מאשר אם היינו משתמשים בספריות מבחוץ שמביאות תוצאות יותר טובות. (לדוגמא החלטת ישויות במסמך)
* 2.מימשנו את שתי שיטות הדירוג inner\_product cos\_similarity מבדיקות שעשינו inner product הניבה תוצאות טובות יותר גם בקורפוס קטן וגם בקורפוס גדול. אנחנו מניחים ששיטה זאת טובה יותר בגלל שהשונות של אורכי המסמכים היא נמוכה. ולכן אם המידע היה שונה ולא טוויטים סביר להניח ש cos\_similarity הייתה טובה יותר.
* 3.המנוע שלנו לא מתייחס למקרים של שאילתות או טוויטים שנרשמו עם טעויות סמנטיות. אם היה לנו spelling correction כנראה שהייתי יכולים לטפל במקרה כזה.