אחזור מידע – Fall 2021

פרויקט מנוע חיפוש

דו"ח חלק א

עיצוב התוכנה:

* ספרית המנוע מורכבת ממספר מחלקות ולכל אחת תפקיד ברצף הפעולת של קריאת, פרסור, אינדוקס הקורפוס ואז תשאול, ודירוג של המסמכים הרלוונטיים לשאילתא.  
  בשורות הבאות נפרט את המחלקות שלוקחות בכך חלק ומה תפקידן באופן מצומצת.
  + Reader – המחלקה האחראית על הגישה לקורפוס, רישום הקבצים הרלוונטיים כקלט, וקריאה שלהם לטוב תהליך הפרסור. המחלקה מטיילת על עץ המבנה של תיקית הקורפוס, ומחלצת את כל קבצי הparquet הרלוונטיים כקלט. לאחר שהרשימה הוכנה, פונקציות הקריאה של קובץ יכולות לקרוא את הקבצים בה ולהעביר הלאה לצרכן הבא.
  + Parser- המחלקה אחראית על פירוק של מסמך \ טוויט מטקסט ומטאדאטא לגורמים הרלוונטיים לטובת האינדקס שניצור בהמשך. המחלקה יודעת לקרוא ולחלק את הטקסט של הציוצים, לחלץ ישויות, האשטג(#), תיוגים(@) מספרים, דולרים ולהוסיף באופן אחיד לרשימת הטרמים שתאוחסן במבנה המסמך שיועבר לאינדקסר.
  + Indexer- המחלקה האחראית על בנית האינדקס לטובת ביצוע השאילתות. מחלקה זו בונה מספר מבני נתונים מבוזרים המכילים מידע רב על הtweets לטובת איחזור המידע בזמן שאילתא. המבנים ממומשים כמילונים והרשימה היא: inverted index, posting (מחולק לקידומת לפי האות הראשונה של הterm), entities index (שומר את כל המידע על הישויות לפני האיחון עם האינדקס), entities posting (מידע הפוסטינג בנוגע לישויות לפני האיחוד) ומילון לכל מסמך המכיל את אורך המסמך ואת התדירות הגבוהה ביותר של מילה במסמך.
  + Searcher- מחלקה זו אחראית לחפש את המסמכים \ טוויטים בקורפוס שיכולים להיות רלוונטים לשאילתא המוצגת. האחזור מתבצע באמצעות קריאה מהדיסק של רשומות הposting הרלוונטיות ל terms שמופיעים בשאילתא, ושימוש ברשומות אלו כי לקבוע באלו מסמכים מופיעים הterms שמקורם בשאילתא.
  + Ranker- מחלקה זו תפקידה לדרג את המסמכים שאוחזרו ע"י מחלקת הSearcher לפי ציון הcosine similarity בין כל וקטור tf-idf של מסמך ווקטור הtf-idf של השאילתא. במחלקה זו מחושבים הtf-idf עבור כל המסמכים הרלוונטיים לשאילתא והטרמים הקיימים בשאילתא, ולאחר מכן מחושב המרחק בינם לבין וקטור הtf-idf של השאילתא.
* מעבר לכך, כדי להתמודד עם מגבלות הזיכון של המחשב תכננות את המערכת כך שתוכל לרוץ בprocessים נפרדים באופן בלתי תלוי בין אחד לשני. וכן בנינו מודול שמאפשר לכל התהליכים לרוץ במקביל. המודול מורכב מprocesses pool, של חמישה תהליכים ונשלחים אל הprocesses pool משימות כמספר קבצי הקלט, ומתבצע תהליך שלם של קריאת הקובץ, פרסינג, ואינדוקס שלו. כאשר הפלט של תהליך כזה הוא כתיבת קובצים זמניים לדיסק, כל אחד מהווה subset של האינדקס הסופי, ומקבל סיומת של המספר הסידורי של המשימה מ0 ועד מס' הקבצים בקלט.

לאחר שכל התהליכים סיימו לעבוד (מוודאים באמצעות join), ניגש לתהליך האיחוד שלהם.  
חלק זה טוען את כל הקבצים הזמניים לכל אחד ממבני הנתונים ומאחד אותם לindex, documents\_dict ואוסף קבצי posting המחולקים לפי הקידומת של המילים המאוחסנות בהם. בתהליך זה מתבצע גם טיפול באותיות גדולות\קטנות ובעצם כל מילה שמופיעה באותיות גדולות נבדק כי אין מילה זהה לה באותיות קטנות. בנוסף בחלק זה מועמדים לישויות המופיעים יותר מפעם אחת נכנסים לאינדקס ולרשימת הterms הסופית.  
ראינו שהעבודה בחמישה תהליכים במקביל גרמה לתהליך הפרסור והאינדקס שלנו לרוץ בזמן טוב של כ40 דק' על החומרה של המחשב הנייד שלנו. מה גם שמנענו גישות רבות לדיסק באמצעות העובדה שכל תהליך יכול לצבור בזיכרון שלו את כל המידע שהוא אוסף ולכתוב לדיסק בסיומו. ולאחר מכן גם בשלב האיחוד, אנו קוראים באופן הדרגתי מהדיסק לזיכרון (בין אם זה את כל קבצי האינדקס, או את קבצי הposting כל פעם לפי אות אחרת), ולבסוף כותבים באופן מרוכז לקובץ בדיסק.

* קבצי הposting- באמצעות תהליך של הערכה וdata exploration הסקנו שכיוון שיש 26 אותיות בABC ועוד קובץ אחד לכל שאר הterms, זה באותו סדר הגודל של קבצי הקלט. כיוון שקובץ קלט אחד נכנס בזיכרון גם יחס עם הפוסטינג הרלוונטי שלו, נראה שחלוקה של כל קבצי הposting לפי אותיות זה דבר שהתוכנה מסוגלת להתמודד איתו בהצלחה. בנוסף זה מאפשר לנו להימנע מלהחזיק עוד מידע באינדקס, של מצביע לקובץ הפוסטינג הרלוונטי. כיוון שיש לנו מיפוי מהיר בין term ובין קובץ הפוסטינג שלו. שמרנו את הקבצים באמצעות מחלקת הutils שסופקה לנו ושומרת קבצים בפורמט pkl ומאפשרת לטעון אותם במבנה המקורי שלהם. מספר קבצי הposting הסופי הוא 27. בנוסף לקובץ inverted index ולקובץ documents dictionary.

מבנה קובץ הposting הוא כזה:  
מילון שבו המפתחות הינם הterms המאוחסנים והערכים הם רשימה של tuples. הtuples מורכבים מdoc\_id ומרשימה שבה האיבר הראשון הוא מס' החזרות של הterm במסמך והאיבר השני הוא הpositional indexing של הterm במסמך.

Term: [(doc\_id, [terms\_freq, [pos\_idx1, pos\_idx2]])]

קובץ האינדקס פשוט יותר, הוא מכיל מילון של הterms ולכל אחד בכמה מסמכים הוא מופיע.

קובץ הdocument\_dict מכיל מיפוי של כל מסמך אל האורך שלו והתדירות של המילה הכי תדירה בו.

* גודל קבוצת הטוויטים החלקית בעצם נבחר להיות קובץ קלט אחד. כיוון שכדי לקרוא את קובץ הקלט עליו להיכנס לזיכרון בתוכנה זו, אנו מניחים שזה קבוצת טוויטים שהמחשב יכול להחזיק בזיכרון בזמן נתון (ובאמצעות ניסוי וטעיה ראינו שגם 5 במקביל מחזיקים באופן יפה). הקבצים נבנים באופן הבא: (קיים גם תיעוד בקוד הרלוונטי במחלקת הindexer)  
  בפונקציה add\_doc במח' הindexer נקלט מסמך חדש, המסמך מכיל בין השאר את מילוני הterms והמועמדים להיות ישויות.(כל מילון כזה מכי לאת מס' המופעים במסמך ואת הpositional index) טיפול בterms בעצם עובר כך  
  מעבר על כל הterms במילון של המסמך הרלוונטי הזה.  
  נעדכן את מספר המסמכים בהם מופיע הterm בindex.

נוסיף את הערך של כמה פעמים מופיע הterm במסמך וכן את רשימת הpositional index שלו למקום הרלוונטי במילון של הposting. (מילון של אותיות שכל ערך שלהן הוא מילון של terms).

באופן דומה נעבור על כל מועמד לישות. ונעדכן באינדקס ובposting.

לבסוף נעדכן את מילון המסמכים שלנו ונוסיף לו את אורך המסמך וכן את הmax\_frequency של המסמך.

לאחר שנעבור את התהליך עבור כל אחד מקבצי הקלט, נאחד ביניהם באופן שתואר מוקדם יותר.

* האלגוריתם בו עשינו שימוש בRanker הינו cossim. בתחילה חישבנו tf לכל מילה במסמך וכן idf לכל מילה במאגר. אחר כך הכפלנו את tf ב-idf לכל מילה בכל מסמך . לאחר מכן הכפלנו את הווקטור של השאילתא בכל ווקטור במסמך וחילקנו את זה במכפלה של –הסכום של ווקטור המסמך ושל הסכום של הווקטור של השאילתא. לאחר שקיבלנו מספר בין 0 ל1 לכל מסמך , מיינו את המסמכים מגבוה לנמוך ועל פי מספר זה. מסמך שיופיע גבוה יותר ברשימה דורג גבוה יותר.
* לחוקי הפרסר הוספנו את:
  + הפירסור של retweet\_url, בלא מעט מהפעמים ניתן למצוא בretweet\_url terms שמצביעים על נושא הציוץ. שימוש בזה יכול להגדיל את הfrequency של הנושא ובכך להגדיל את ציון הדמיון בין השאילתא ובין המסמך.

לדוגמא: בטוויט 1281008388055457798, הטקסט המלא הוא:

RT @JoeBiden: Increased testing isn’t the reason for the surge in coronavirus cases — Donald Trump's failed leadership is. <https://t.co/j7v…>

ושדה הretweet\_url הוא: {"https://t.co/j7vKxnlaj4":"https://www.axios.com/coronavirus-cases-testing-growth-62d6256b-33e2-491d-b94e-91110a74bc85.html"}

שדה הurl עצמו ריק לגמרי. וניתן לראות שאם נחפש שאילה בנוגע לעליה בבדיקות הקורונה, המידע שיפורסר משדה retweet\_url יוכל להעלות את הסיכוי שלנו למצוא testing growth לדוגמא.

* + בשלב הפרסינג אנחנו מחפשים גם מספרים שלאחריהם המילה או הסימון דולר. באמצעות זה אנחנו יכולים לפרמט מספרים בדולרים כך שבאופן אחיד כל ההופעות של מספר עם דולר ישמרו יחד. וזה מאפשר לחפש בשפה חופשית או במספר ולמצוא ערכים שהיו בשפה חופשית או במספר.

לדוגמא: בטוויט 1281675432966991873, שדה הfull\_text הוא: @melissadderosa Have you forgotten the medical ship, the javitis center hospital, the ventilators, the ppe’s, all supplied by the federal government? Have you forgotten the 6 billion dollar deficit that your boss built up prior to the virus hitting New York? Spin it fairly.

שמרנו את הצירוף 6 billion dollar כ6B$ וכך כל פעם שיחפשו את 600,000,000 $ לדוגמא, ימצאו את הtweet שלנו כי זה יפורמט באותו האופן. מה שמחזיר טוויטים רלוונטיים יותר לשאילתא.

* פרטי מידע נוספים ששמרנו:
  + האינדקסים של ה terms בכל טוויט:

הסיבה העיקרית לכך היא, שאם נרצה לבדוק האם יש מילים בשאילתא אשר הופיעו אחד ליד השני בטוויטים , אנחנו יכולים ללכת למילון לחפש את המילה ואז לחפש את המסמכים שהיא הייתה ואת האינדקסים בהם היא הופיעה במסמכים. אותו דבר נעשה במילה השנייה בשאילתא ואז נשווה האם יש מסמכים שווים ואם יש , נבדוק האם יש אינדקסים צמודים.

* + סימנו באופן מיוחד terms שהם ישויות, (positional index=None) וכך בעצם בעת תהליך הדירוג, נוכל לבחור לתת משקל גבוה יותר לישויות, כיוון שבמידה וחיפשת ישות, מסמך שבו מופיעה הישות הזו כנראה יהיה רלוונטי בשבילך.
* ספריות קוד פתוח בהן עשינו שימוש:

nltk.corpus lin\_thesaurus

היה עלינו לממש את גישת הthesaurus בשלב איחזור השאילתות, ולכן, השתמשנו בה כדי לייבא את המילים של ה- thesaurus .

עשינו שימוש בפונקציה synonyms של המודול.

nltk.stem.porter

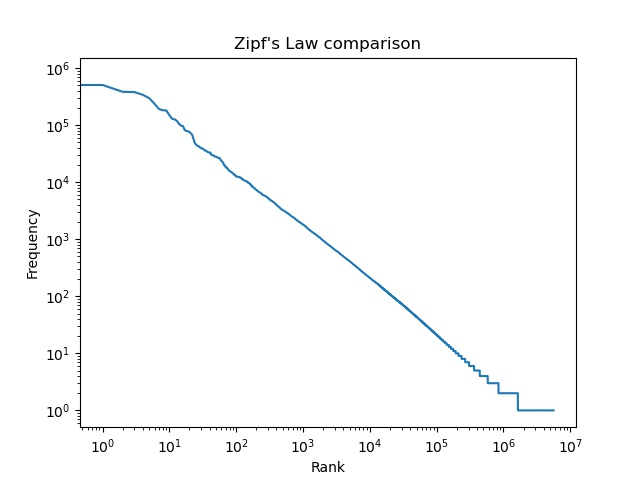
השתמשנו בה כדי לבצע stemming בשיטת porter.

עשינו שימוש בפונקציה stem של המודול.

* הערה שחשוב לדעת, משיקולי זמן ריצה ואיחזור רלוונטי לשאילתא, בחרנו להגביל את מס' הsynonyms לכל term בשאילתא לאחד, כך אנחנו לא מבזבזים זמן ארוך על זה וכן מונעים ממילים שכלל לא קשורות להוות את רוב המשקל בשאילתא.
* יתרונות המנוע שלנו על פני מנועים אחרים:
  + המנוע שלנו מאפשר לעבוד באופן גנרי ורחב על מגוון קורפוסים במגוון נושאים.
  + שיטת האינדוקס שלנו מאפשרת להוסיף מסמכים חדשים \ קבוצת מסמכים חדשה למאגר תוך כדי תנועה באופן נח יחסית. ניתן לבנות קבצים זמניים ולמזג אותם אל האינדקס. אנחנו כבר תומכים באיחוד של מס' תהליכי פרסור ואינדוקס במקביל ומיזוגם בתום התהליך, כך שמדובר בהוספת פונקציונליות מעטה יחסית.
  + השימוש בשיטת thesaurus מאפשר לנו לקחת בחשבון גם מילים נרדפות למה שביקש המשתמש ולהחזיר גם שאילתות שעשויות להיות קשורות לנושא השאילתא גם אם מילות השאילתא לא מופיעות בו באופן מפורש.
* חסרונות המנוע שלנו על פני מנועים אחרים:
  + המאגר שלנו מאוד רגיש לגודל הקורפוס מבחינת אינדוקס וכן מבחינת זמני איחזור. כיוון שאנחנו בונים אותו מ0 ולא תוך כדי תנועה. בנוסף בתהליך האיחזור יש חישובים שאנחנו מבצעים שלוקחים לא מעט זמן, ותלויים ב מספר המסמכים פוטנציאלים המקושרים לשאילתא .
  + מבחינת דיוק של השאילתא, אין לנו תמיכה בשאילתות "מדויקות" שבהן המשתמש ירצה להעדיף מחרוזת מדויקת אם הופיעה כפי שהיא במסמך.
  + באיחזור תלוי קונטקסט, אנו מוגבלים יחסית למילים הנרדפות שהthesaurus מחזיר לנו, אך לא נותנים משקל לדמיון בין מילים במשמעות, לא מבצעים spelling correction וכד', ולכן איכות התוצאות שלנו היא לא אופטימלית.

עיבוד הנתונים במאגר:

* מספר הterms השונים ללא stemming: 5,535,037
* מספר הterms השונים אחרי stemming: 5,402,382
* Terms שהינם מספרים במאגר: 1,970,566
* 10 הterms השכיחים ביותר במאגר:
  + ('twitter.com', 747154)
  + ('...', 508470)
  + ('2.020K', 381786)
  + ('#covid19', 364765)
  + ('https', 342592)
  + ('19', 297919)
  + ('7', 237599)
  + ('1', 192399)
  + ('#twitter.com', 182535)
  + ('i', 182452)
* 10 הterms הכי פחות שכיחים במאגר:
  + ('//t.co/1vuzmb2e4q', 1)
  + ('1291682.553B', 1)
  + ('#donatewithoutspending', 1)
  + ('//t.co/srhgvm7yaf', 1)
  + ('1291683.090B', 1)
  + ('1291684.302B', 1)
  + ('1291685.344B', 1)
  + ('@skreilly', 1)
  + ('//t.co/jbgzs7xc1h', 1)
  + ('1291685.501B', 1)
* גרף שכיחות מילה כנגד דירוגה:



ניתן לראות שבסך הכל הגרף מתנהג כמו חוק זיפף, עם מעט סטיה בקצוות, בדיוק כמו שדנו בהרצאה.

* גודל קבצי הposting עם stemming: kb2660000, (2.66GB)
* גודל קבצי הposting ללא stemming: kb831000 (831MB)
* משך בנית האינדקס כל קבצי הcorpus: 34.7 דקות.
* ניתוח תוצאות הטוויטים:

שאילתא 1:

Dr. Anthony Fauci wrote in a 2005 paper published in Virology Journal that hydroxychloroquine was effective in treating SARS.

טוויט מדורג ראשון :

השאילתא מדברת על hydroxychloroquine , כתרופה אפקטיבית נגד הקורונה.

בטוויט שמדורג ראשון אצלנו מציין בדיוק את אותה טענה ומביא דוח בנושא.

הסיבות שהטוויט דורג אצלנו ראשון הינם:

בטוויט מוזכר פעמיים #SARS. כמו כן המילה מופיעה בכותרת הדוח שמובא בלינק בטוויטר. מבחינת tf המילה SARS הוא ה max term בטוויט זה.

בטוויט מוזכר hydroxychloroquine פעם אחת.

בטוויט מוזכר המילה 2005 .

טוויט מדורג שני :

ישנם כמה סיבות שזהו הטוויט שמדורג שני אצלנו בשאילתא זו.

בטוויט מוזכר פעמיים hydroxychloroquine. מבחינת tf המילה hydroxychloroquine הוא ה max term במטוויט זה, ביחד עם מילה נוספת שלא מוזכרת בשאילתא זו.

בטוויט מוזכר SARS פעם אחת.

בטוויט מוזכר המילה 2005 .

טוויט מדורג שלישי :

המילה paper מופיעה פעמיים. פעם אחת בלינק המופיע בטוויט וכן פעם נוספת בגוף הטוויט.

לפי החישוב tf-idf , בהנחה שהמילה לא מופיעה בכל המסמכים, המסמך יקבל ערך גבוה יותר בגלל הtf .

ייתכן שזו הסיבה שבגללה הטוויט דורג במקום זה.

טוויט מדורג רביעי :

בטוויט מופיעה המילה published, וייתכן שעקב כך שמימשנו thesarus , חזרה המילה בשאילתא שכן היא דומה לwrote .

טוויט מדורג חמישי :

בטוויט זה לא מוזכר מילים בשאילתא.

טוויט מדורג חמישי :

בטוויט זה לא מוזכר מילים בשאילתא. לא ברור למה חזר טוויט זה במקום זה.

שאילתא 2:

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

טוויט מדורג ראשון :

השאילתא מדברת על שפעת עונתית שהורגת מדי שנה יותר ממחלקת הקורונה.

הטוויט מדבר סינון אוויר לחסל מזהמים.

בטוויט שמדורג ראשון אצלנו מוזכרים המילים ,flu COVID-19 , Kills .

כנראה בגלל ה cos similitary הטוויט יצא ראשון בדירוג.

טוויט מדורג שני :

בטוויט מוזכר המילה month . בשאילתא מוזכרת המילה year וdate . ייתכן שמכיוון שמימשנו thesarus המילה Month הצטרפה לשאילתא ו,ולכן חזר מסמך זה במקום זה.

טוויט מדורג שלישי :

הטוויט מדבר על מפגש שסוקר כיצד ארגונים מתמודדים עם מחלת הקורונה.

בטוויט מוזכר פעמיים COVID-19. פעם אחת בגוף הטוויט , ובפעם השנייה בכתובת הurl הנמצאת בטוויט.

כנראה בגלל שה-max term של המילה COVID-19 הכי גבוה בטוויט זה, הטוויט חזר גבוה בדירוג.

טוויט מדורג רביעי :

הטוויט מדבר על מפגשיים חברתיים ושירה בזמן הסגר. בשאילתא מוזכר covid 19 , יתכן שבעזרת המימוש של הthesarus , "סגר" היא מילה שקשורה ל covid 19 ולכן חזר טוויט זה במקום הרביעי.

טוויט מדורג חמישי :

בטוויט מוזכר season . לאחר שעשינו stemming על הדאטה, יכול להיות שזו הסיבה שהטוויט חזר במקום החמישי.

שאילתא 4:

The coronavirus pandemic is a cover for a plan to implant trackable microchips and that the Microsoft co-founder Bill Gates is behind it

טוויט מדורג ראשון :

השאילתא מדברת על סיפור כיסוי של חקירה אפימדיולוגית של מיקורסופוט כדי להתקין שבבים על אנשים כדי לעקוב אחריהם.

הטוויט מזכיר בדיוק את אותו טענה ולכן דורג ראשון.

טוויט מדורג שני :

בטוויט וגם בשאילתא מוזכר Bill Gates כישות. בנוסף לכך בטוויט מוזכרת המילה vaccination , והיא בהחלט מילה שיכולה להיות מוחזרת עקב המימוש שלנו ב-thesarus ,אשר היא דומה בcontext שלה לחקירה אפימידיולוגית , אשר מטרתן של שתיהן הם אמצעים להתמודדות נגד הקורונה.

ייתכן שזו הסיבה שהטוויט דורג במקום השני.

טוויט מדורג שלישי :

כמו שהסברתי בטוויט שמדורג ראשון, הטוויט מדבר על Bill Gates ועל החיסונים שהוא מציע נגד נגיף הקורונה. אמנם בטוויט לא מוזכר חקירה אפימיודלוגית כמו שמוזכר בשאילתא, אבל מוזכר vaccination אשר מתאר הצעה של ביל גיטס לטיפול בקורונה.

טוויט מדורג רביעי :

הטוויט נותן קישור לראיון שעשו עם ביל גיטס. בטוויט מוזכר פעמיים Bill Gates (פעם אחת עם hashtag) וכן מוזכר Microsoft , ו - vaccines . ייתכן כמו שציינו קודם עקב כך שמימשנו theasarus ו vaccines היא מילה שהקשרה דומה ל-חקירות אפיומודולוגיות שמוזכרים בשאילתא, זה נתן עוד משקל לעך שהטוויט חזר בדירוג זה.

טוויט מדורג במקום החמישי:

CORONAvirusכתובה כאן קצת בצורה שונה ולא בדרך רגילה(בצורה יותר נדירה). ייתכן שעקב כך, וכמובן עקב זה שהמילה מופיעה גם בשאילתא וגם בטוויט זה , הטוויט דורג במקום זה.

שאילתא 7:

Herd immunity has been reached.

טוויט מדורג ראשון :

השאילתא מדברת על חיסון עדר בציבור.

בטוויט לא מוזכר נושא אבל מוזכר המילה reached"" כמו בשאילתא וייתכן עקב כך שמילה זו היא נדירה, חזר מסמך זה.

טוויט מדורג שני :

גם טוויט זה יכול לאשש את הטענה שלי בדירוג הראשון , שהטוויט לא מדבר על נושא השאילתא אבל גם פה מוזכרת המילה reached"" , שיכול להיות שהיא נדירה בקורפוס , ולכן דורג טוויט זה במקום זה.

טוויט מדורג שלישי :

בטוויט מוזכר המילה privillege . בשאילתא מוזכרת המילה immunity .

שתי המילים נרדפות ולכן מכיוון שמימשנו thesarus , ייתכן שהמילה חזרה בשאילתא ולכן חזר מסמך זה.

טוויט מדורג רביעי :

אותה סיבה בדיוק המופיעה בטוויט המדורג שלישי.

לא נמצא טוויט זה.

טוויט מדורג חמישי :

יכול להיות שעקב כך שמוזכרת המילה Chasing ובגלל שמיממשנו thesarus , המילה דומה ל"reached " ולכן דורג טוויט זה במקום זה.

שאילתא 8:

"Children are “almost immune from this disease.”

טוויט מדורג ראשון :

השאילתא מדברת על כך שילדים חסינים ממחלת הקורונה.

הטוויט מזכיר את אותו טענה בגיחוך. עקב כך שמוזכרת אותו טענה שמוזכרת בשאילתא והמודל שלנו כרגע לא יודע להבדיל לבאיזה נימה נאמרה האמירה, הטוויט דורג ראשון .

טוויט מדורג שני :

בטוויט רואים בפירוש שה max term שלה הוא Children"" (4 פעמים) ולכן הטוויט דורג במקום זה.

טוויט מדורג שלישי :

בטוויט זה מוזכרת המילה Children"" ולכן נראה שזו הסיבה שדורג טוויט זה במקום זה.

טוויט מדורג רביעי :

בטוויט זה מוזכרת המילה Children"" וכן המילה "kids" . ייתכן שעקב כך שמימשנו theasarus וkids היא מילה דומה לchildren , וכן מופיעה המילה children בעצמה, הטוויט דורג במקום זה.

טוויט מדורג חמישי :

בטוויט רואים שה max term שלה children"" ( המילה מופיעה פעמיים) וייתכן שזו הסיבה שהטוויט דורג במקום זה.