**IR final project**

1. **Student IDs and emails.**

Name: Eden Tzarfaty Name: Yadin Deri Id: 209299684 Id: 206116204 E-mail: [edenrivk@post.bgu.ac.il](mailto:edenrivk@post.bgu.ac.il) E-mail: [yadinde@post.bgu.ac.il](mailto:yadinde@post.bgu.ac.il)

1. **A link to a GitHub repo with the code.**

Link- https://github.com/YadinDeri/IR-project-WIKI-Z.git

Search engine domain - http://[35.226.44.201:8080](http://35.226.44.201:8080/search_body?query=paris)/

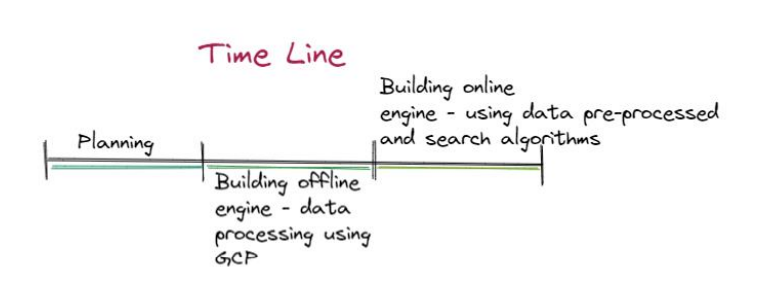
1. **A link to a Google Storage Bucket, where all indexing data you calculated resides and is publicly accessible. Instructions for making your bucket public are here.**

Link- <https://console.cloud.google.com/storage/browser/ir-project-z>

1. **List all index files with human-readable sizes, pasted in an appendix at the end of your report.**

Link- <https://storage.googleapis.com/ir-project-z/listfiles.txt>

1. **A description of key experiments you ran, how you evaluated them, and the key findings/takeaways**

ניסוי המפתח שערכנו מתבססים על שיטות שונות למציאת דמיון בין מסמך לשאילתה. שיטת העבודה מתבססת על ציר הזמן הזה **:**

**חלק א' :**

תכנון השדות והפונקציות הדרושות, העיבוד אשר נדרש לבצע על הקבצים הגולמיים של הקורפוס על מנת להשיג חישוב מינימאלי בזמן ריצת שאילתה.

**חלק ב' :**

בנייה בפועל בעקבות התכנון של חלק א'.

Inverted Index creation .1

Index Inverted הוא מבנה נתונים המקשר בין מסמכים למילים. נרצה להחזיק את קובץ האינדקס בזיכרון הראשי כיוון שביחס לכל המאגר הגודל שלו קטן.

לכל מסמך במאגר נעשה tokenizing , כלומר, נמיר אותו לtokens ומפעילים עליו text operations כך שהתוצאה הסופית היא אוסף של terms.

הIndex Inverted מחזיקlist posting שמכילה מידע אודות מיקום של כל מופע של כל ביטוי במסמך בנוסף למאפיינים נוספים.

בפרויקט שלנו יצרנו 3 סוגים שונים של index Inverted (בפועל 5):

1. inverted\_index\_body

כחלק מהיצירה של הInverted Index בעבור גוף המסמך שבנינו בעבודה שלוש, יצרנו מילונים נוספים והם:

* מילון שממפה מספר מזהה של מסמך לאורכו– self.doc\_len
* מילון שממפה מספר מזהה של מסמך לכותרתו – self.doc\_name
* מילון שממפה מספר מזהה של מסמך לסכום משקלי המילים בריבוע שמופיעות באותו מסמך– self.dominator\_mapping
* מילון שממפה מילה בקורפוס לערך הidf שלה – self.idf

1. inverted\_index\_anchor/title

גם ביצירת Index Inverted לכותרת המסמך ולעוגני המסמך הסתמכנו על הקוד ששימש אותנו בעבודה שלוש כדי ליצור instance של מחלקה זאת עבור כל אחד מהחלקים בנפרד ובהמשך גם קבצים שאיתם נעבוד על פי מחלקה זאת. השינוי המרכזי שבוצע מתרכז בדרך הייחודית שעלינו היה לבנות את השדה עבור כל אחד מאלה. נציין שבדומה ליצירת הindex על גוף המסמך, גם לאלה הוספנו מילון שממפה מספר מזהה של מסמך לכותרתו- . self.doc\_name

2. Page Rank

על מנת ליצור קובץ pagerank.csv שממפה מסמך לדירוג שלו השתמש ו בקוד של עבודה 3 .

3. Page View

על מנת ליצור מילון שממפה מסמך לכמות הצפיות שלו pageviews.pkl השתמשנו בקוד של עבודה 1.

ביצענו את חישובי הדמיון בקובץ frontend\_search.py בעזרת קריאת הקבצים הסופיים שנוצרים לאחר הסעיפים הנ"ל.

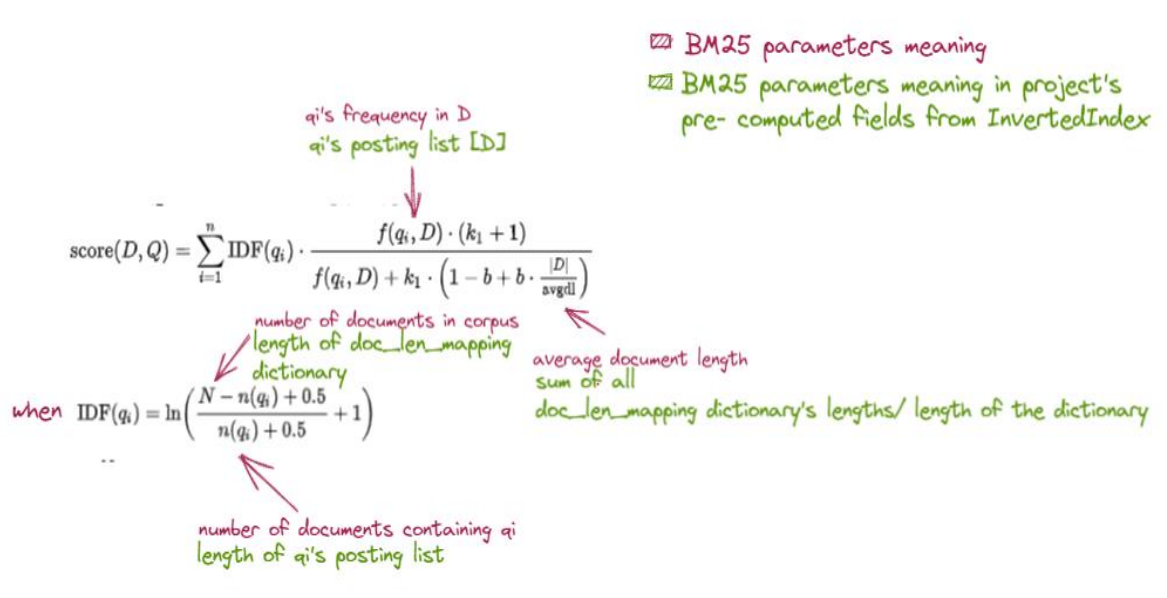
**חלק ג':**

בזמן ששאילתה מגיעה לאחת הפונקציות במנוע החיפוש היא עוברת תהליך tokenizing וoperations text כמו ששאר חלקי המסמך בקורפוס עברו על מנת "לנקות" את השאילתה.

השיטות המרכזיות לחישוב דמיון בהם השתמשנו:

**search() function – BM25 Algorithm** .1

BM25 היא פונקציית אחזור בעבור קבוצת מילים) אצלנו- שאילתה) אשר מדרגת סט של מסמכים )אצלנו הקורפוס הנתון הינו מסמכי וויקיפדיה) על סמך ה terms של השאילתה המתקבלת אשר מופיעים במסמך.

****שיטת הניקוד score – שאילתה Q ומסמך D:

יתרונות עיקריים:

* יעיל.
* מבוסס על שני רכיבים עיקריים – IDF TF.

חסרונות עיקריים:

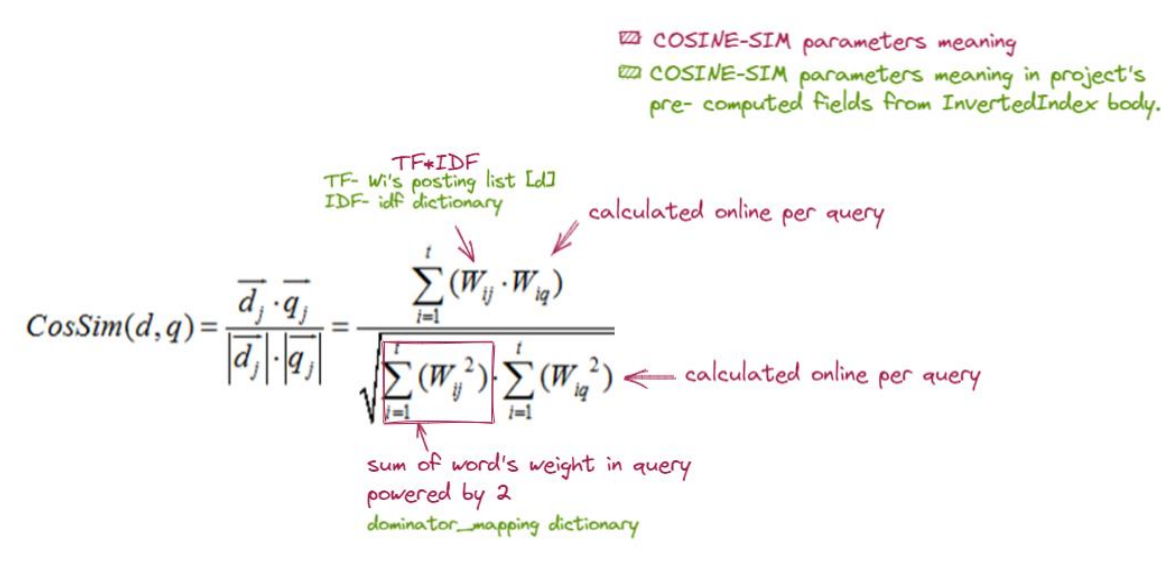
* קשה להרחבה.

**search() function – merge results** .2

חישבנו עבור כל אחד מחלקי המסמך את תוצאות ניקוד ה BM25 בעבורו – body, anchor, title בעזרת הפרמטרים שנוצרו ביצירת Inverted Index של כל אחד מהם. **מיפוי בין מסמך לאורכו חושב ב Inverted Index של הbody בלבד** ולכן השתמשנו בו בעבור כל אחד מחלקי המסמך לצורך החישובים על מנת לא לבצע שכפול קוד. לאחר מכן, לקחנו את כל אחד מתוצאות החישובים ומיזגנו אותם על מנת לקבל תוצאה מיטבית. החלטנו לתת לכותרת משקל של 0.65, לגוף 0.35 ולanchor 0.

**search\_body() function – TF-IDF, cosine similarity** .3

cosine similarity - שיטה לחישוב דמיון בין שאילתה למסמך המתבססת על מודל ייצוג וקטורי - מודדת את קוסינוס הזווית בין ווקטור השאילתא לווקטור המסמך.



יתרונות עיקריים:

* שיטה מתמטית חישובית פשוטה.
* אפשרי למימוש על מאגר גדול.
* מתחשבת בנתוני המסמך עצמו וגם בהופעת המילה במאגר.

חסרונות עיקריים:

* מתבסס על הופעה ספציפית של מילה במסמך.
* מסתכל על כל מילה בנפרד ולא על צירופי מילים.
* ייתכן מצב שבו מסמך שמופיעה בו רק מילה אחת בתדירות יחסית גבוהה מהשאילתה, ידורג גבוהה יותר ממסמך אחר שבו מופיעות כל המילים של השאילתא בתדירות נמוכה יותר.

**search\_title(), search\_anchor() functions – Boolean model** .4

Boolean model – מתבסס על תדירות המילה, מסמנת את ניקוד המילה ב1 או 0, כאשר 1 אם המילה נמצאת במסמך ו0 אם לא. גם על פי מודל זה האחזור מתבסס על הדמיון בין ששאילתה לבין מסמך. בפרויקט שלו בחנו את ציון דמיון השאילתא לחלק המסמך הרלוונטי עפ"י כמות המילים בשאילת אשר מופיעות בחלק המסמך הרלוונטי ועפ"י כמות זו דרגנו את המסמכים.

יתרונות עיקריים:

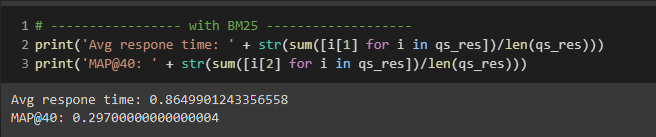
* קל ליישום.
* אם ערכת המסמכים המתקבלת קטנה מדי או גדולה מדי, ברור ישירות אילו אופרטורים יפיקו סט גדול יותר או קטן יותר, בהתאמה.

חסרונות עיקריים:

* התאמה מדויקת עשויה לאחזר מעט מדי או יותר מדי מסמכים.
* קשה לתרגם שאילתה לביטוי בוליאני.
* כל המונחים משוקללים באופן שווה.
* לא ניתן דירוג של המסמכים (העדר סולם דירוג).

1. **performance graph**

based on the following:

****1. Run\_frontend\_colab Jupiter notebook: “Test your app” part. 2. Script:

להלן התוצאות:

**לאחר מחשבה עמוקה והרצת ניסויים רבים הבנו כי שימוש ב25BM בפונקציית search נותנת לנו תוצאות חיפוש פחות טובות בהשוואה לחיפוש באמצעות כותרת ולכן לבסוף החלטנו לבצע חיפוש עפ"י כותרת.**