**מגישות: בינה רוזנטל ומיכל גבאי**

# <u>קובץ אנליזה</u>

## א. <u>מבנה האינדקס שמימשנו</u>

הפונקציה write(self, inputFile, dir) במחלקה indexWriter אחראית על בניית האינדקס על הדיסק.

הפרמטרים לפונקציה:

inputFile - נתיב לקובץ המכיל את נתוני הביקורות (מידע גולמי).

dir- התיקייה בה יווצרו כל קבצי האינדקס.

בתרגיל זה עבדנו על תמיכה בערכות נתונים גדולות מאוד.

בעבודה עם קבצים גדולים, לא ניתן לקרוא לזיכרון בבת אחת את כל המידע ולנתח אותו, לכן פעלנו בצורה הבאה:

חילקנו את המידע ל"בלוקים" של 100 אלף ביקורות ולכל בלוק בנינו אינדקס חדש.

לבסוף, מיזגנו את כל האינדקסים יחד לאינדקס סופי יחיד. תוכנית המיזוג תפורט בהמשך.

(עם זאת, ההנחה היא כי עבור המילון יש תמיד מספיק מקום בזיכרון).

- בשלב ראשון, הפונקציה קוראת את קובץ המידע הגולמי ושומרת, לכל review, את הנתונים הבאים:
  productId, helpfulness, score, text
  שדה הhelpfulness נשמר כ-2 מספרים שלמים, מונה ומכנה.
  שדה הtext מחולק למילים נפרדות- tokens- ועובר נרמול לאותיות קטנות.

- בניית הdictionary:

מעבר על כל הtokens והכנסתם למילון ללא חזרות, תוך כדי ספירת הנתונים הבאים:

total times appearience- מספר המופעים של המילה בכל האוסף.

reviews number- מספר הביקורות בהם המילה מופיעה.

review list- רשימת תפוצה - מבנה dict של פייתון השומר את ה review ID's בהם המילה מופיעה ואת מספר הפעמים שהיא מופיעה בכל ביקורת (freq).

הערה: review ID's ניתנים לכל ביקורת בסדר עולה, החל מ-1.

<u>כתיבת קבצי האינדקס על הדיסק:</u>

1. כתיבת רשימות התפוצה לקובץ בינארי lists.bin.

דחיסת המידע:

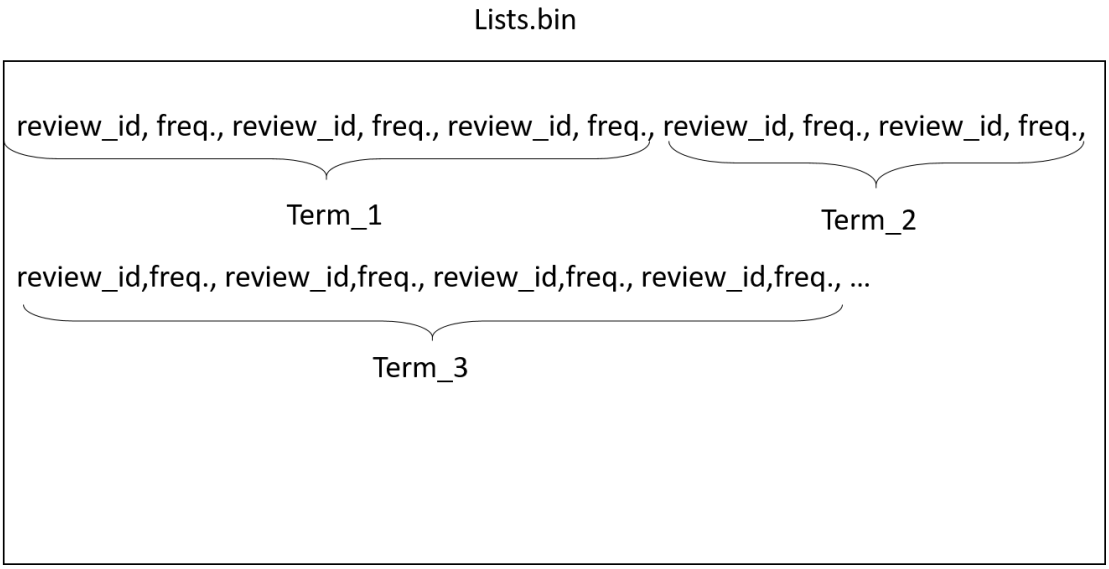
- חישוב gaps של הreview ID's כדי להקטין את גודל המספרים שיכתבו לקובץ.
- דחיסת Length-Precoded Varint Encoding על כל המספרים ברשימה (reviewID + freq.)

כתיבת הרשימות review list מתבצעת לכל token בצורה ממוינת.

תוך כדי הכתיבה לקובץ, אנו מחשבות את אורך הרשימה בבתים ושומרות את זה ב dictionary עבור ה token המתאים (דריסת השדה review list).

**מגישות: בינה רוזנטל ומיכל גבאי**

תרשים הקובץ:




כאשר, כמובן, המידע על הקובץ שמור בצורה בינארית לאחר קידוד.

2. כתיבת ה dictionary לקובץ dict.txt:

כתיבת ה tokens בצורה ממוינת אל הקובץ, לכל token אנו כותבות גם את המידע הבא:

reviews number, total times appearience- הוסבר לעיל

list size in bytes- אורך (בבתים) של רשימת התפוצה שלו, כפי שחושב בשלב הקודם.

תרשים הקובץ:

Dict.txt

Term_1(x,y,z)Term_2(x,y,z)Term_3(x,y,z)
Term_4(x,y,z)...

X- total times appearance
Y- reviews number
Z- list size in bytes

3. כתיבת הקובץ meta data.txt:

כל שאר המידע על הביקורות ישמר בקובץ זה, באופן הבא:

- בשורה ראשונה נשמור את מספר הביקורות באוסף.
- בשורה שניה נשמור את מספר הtokens באוסף.
- בשורות הבאות נשמור, עבור כל reviewID, את השדות הבאים:

r_id- מזהה הביקורת

p_id- מזהה המוצר

h_numerator- המונה של helpfulness

h_denominator- המכנה של helpfulness

Score

txt_len- מספר המילים בtext

השדות נשמרים עם רווחים ביניהם ושורה חדשה לפני כל ביקורת.

תרשים הקובץ:

**מגישות: בינה רוזנטל ומיכל גבאי**

Meta data.txt

```
num reviews
num tokens
r_id, p_id, h_numerator, h_denominator, score, txt_len
r_id, p_id, h_numerator, h_denominator, score, txt_len
```

פירוט תוכנית המיזוג:

1. מיזוג הקבצים dict.txt ו- lists.bin :

התוכנית קוראת בכל פעם שני buffers של dict.txt ושני buffers של lists.bin למיזוג ופותחת buffer שלישי, לכל אחד, אליו יכתב המידע הממוזג.

אנו עוברות על כל הtokens, בכל פעם משוות בין 2 מילים וכותבות ל buffer השלישי באופן ממוין.

תוך כדי, פנינו לרשימות התפוצה המתאימות ומיזגנו גם אותם לbuffer המיזוג של הרשימות.

2. מיזוג קבצי הmeta data.txt:

התוכנית קוראת בכל פעם שני buffers של meta data.txt למיזוג ופותחת buffer שלישי אליו יכתב המידע הממוזג.

בשני השורות הראשונות- יש צורך רק לסכום את המספרים (ראה לעיל מבנה הקובץ).

שאר השורות- שרשור המידע קובץ אחר קובץ.

## ב. <u>קריאת האינדקס</u>

בעת יצירת האובייקט IndexReader נפתחים שלושת קבצי האינדקס לקריאה.

- dict.txt- נקרא כולו מהדיסק לזיכרון לתוך מבנה נתונים dict של פייתון, כאשר ה key הוא token וב value שאר המידע על הtoken.
  בנוסף, תוך כדי המעבר על הtokens, אנו סוכמות את השדה list size in bytes כדי לשמור מצביע לתחילת רשימת התפוצה המתאימה בקובץ lists.bin.
- lists.bin- בעת יצירת אובייקט IndexReader עוד לא נקרא ממנו כלום לזיכרון. מקובץ זה נקרא חלקים בהמשך, באופן הבא:
  בפונקציה getReviewsWithToken יש צורך לקרוא את רשימת התפוצה של token נתון.
  בעזרת המצביע ששמרנו לתחילת הרשימה, נוכל לגשת למיקום המתאים בקובץ lists.bin ולקרוא ממנו את הבתים הרלוונטיים (בעזרת האורך בבתים ששמרנו).
- meta data.txt- בעת יצירת אובייקט IndexReader עוד לא נקרא ממנו כלום לזיכרון, נקרא ממנו חלקים בהמשך באופן הבא:
  עבור כל הפונקציות המבקשות מידע על פי reviewID נתון, נחפש בקובץ שורה המתחילה עם הreviewID המתאים ונחזיר את המידע המבוקש.

## ג. <u>ניתוח גודל האינדקס</u>

(ניתוח תאורטי. ניתוח ביצועי התוכנית בפועל מפורט בהמשך)

<u>הנחות:</u>

- מספר הreviews באוסף הוא n, עם 70 מילים בכל review (text).
- גודל האוסף (מס' המילים באוסף) הוא 70*n
- בממוצע, מילה מופיעה ב- n/80 מה- reviews
- אורך מילה ממוצעת הוא 8 אותיות

**מגישות: בינה רוזנטל ומיכל גבאי**

- כל אות דורשת מקום אחסון של byte
- ניתן לאחסן את כמות המופעים של הביטוי (term frequency) ב-4 בתים.
- את גודל רשימת התפוצה בבתים, בממוצע, ניתן לרשום ב-4 בתים.
- ניתן לאחסן מזהה ביקורת (reviewID) ב-4 בתים.

1. ניתוח הקובץ dict.txt:

את גודל אוצר המילים (המילים השונות באוסף) נסמן ב- numOfTokens.

ע"פ חוק Heaps (הנלמד בהרצאה):

numOfTokens = k*√70*n כאשר 30 <= k <= 100

אם כך, ע"פ החישוב וההנחות לעיל וע"פ מבנה הקובץ dict.txt, הגודל הצפוי של הקובץ:

numOfTokens * (token_len + x + y + z) =

k*√70*n * (8bytes + 4bytes + 4bytes +4bytes) =

20*k*√70*n Bytes

<u>הערה:</u> הכתיבה היא לקובץ txt ולכן מספר הבתים שכל מספר תופס הוא log(num) – מספר הספרות. כדי שנוכל לחשב, הנחנו שמספר הספרות הוא בממוצע 4 ← 4 בתים.

2. ניתוח הקובץ lists.bin:

נסמן ב-numReviews את מספר הביקורות בהם token מופיע, בממוצע.

ע"פ ההנחה השלישית, numReviews = n/80

נסמן ב- freq. את מספר הפעמים שמילה מופיעה ב review מסוים.

מכיוון שכל review מכיל 70 מילים (הנחה מס' 1), ה- freq. שווה לכל היותר 70 ותופס 2 בתים בלבד (לאחר קידוד LP Varint).

אם כך, ע"פ מבנה הקובץ, הגודל הצפוי של הקובץ הוא:

numOfTokens * numReviews * (review_id + frequency) =

k*√70*n * n/80 * (4bytes +2bytes) =

3/40* k*n*√70*n Bytes

3. ניתוח הקובץ meta data.text:

נסמן num_reviews = n להיות מספר הביקורות באוסף.

השדה product_id מורכב (בד"כ) מ-10 תווים ← 10 בתים.

השדה score הוא מספר בין 1-5 ← 1 בתים.

השדה txt_len הוא לכל היותר 70 ← 2 בתים (כתיבה לקובץ txt).

אם כך, ע"פ מבנה הקובץ, הגודל הצפוי של הקובץ הוא:

num_reviews + num_tokens + num_reviews * (r_id + p_id +

h_numerator + h_denominator + score + txt_len) =

log(n)bytes + log(70*n)bytes + n * (4bytes + 10bytes +4bytes + 4bytes

+ 1bytes + 2bytes) =

log(n)bytes + log(70*n)bytes + 25*n Bytes

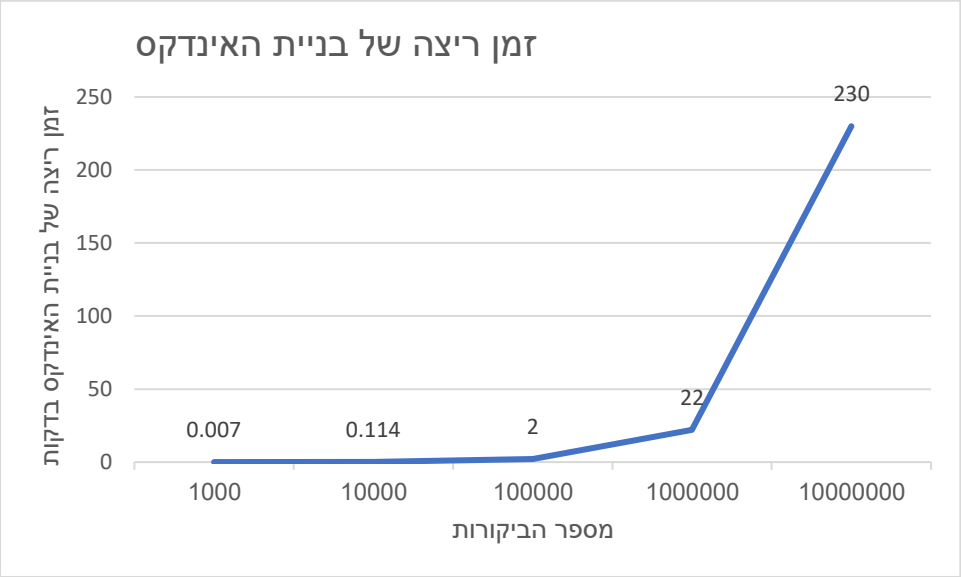
<u>הערה:</u> הכתיבה היא לקובץ txt ולכן מספר הבתים שכל מספר תופס הוא log(num) – מספר הספרות. כדי שנוכל לחשב, הנחנו שמספר הספרות הוא בממוצע 4 ← 4 בתים.

## ד. **ניתוח ביצועים:**

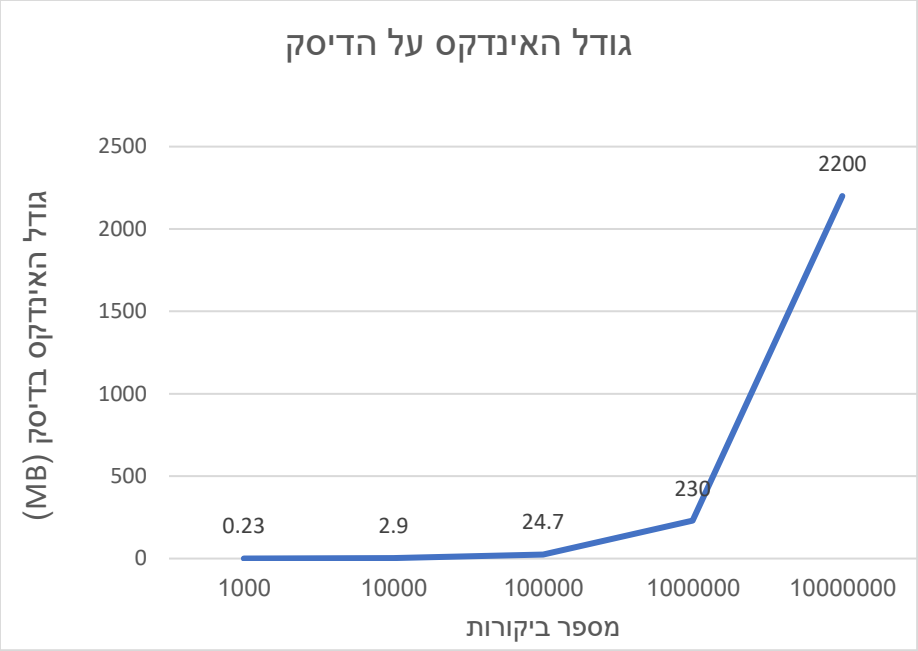
הרצנו את תוכנית בניית האינדקס (כולל מיזוג) ואת תוכנית קריאת האינדקס תחילה על קובץ עם 1000 ביקורות ולאחר מכן עם מספר עולה של ביקורות, בכל פעם פי 10 ביקורות מהניסוי הקודם, עד לקובץ של 10 מיליון ביקורות.

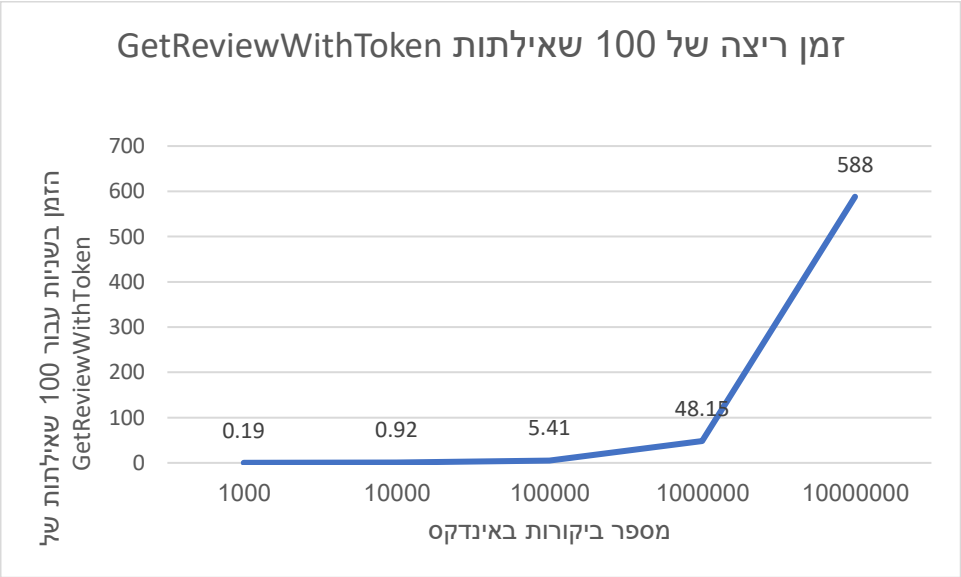
התוכנית התמודדה יפה עם הקובץ של ה10 מיליון ביקורות ובנתה עבורו אינדקס סופי ממוזג במבנה הרצוי.

- ❖ פירוט האנליזה בעזרת גרפים המראים את הביצועים כפונקציה של מספר הביקורות באינדקס:

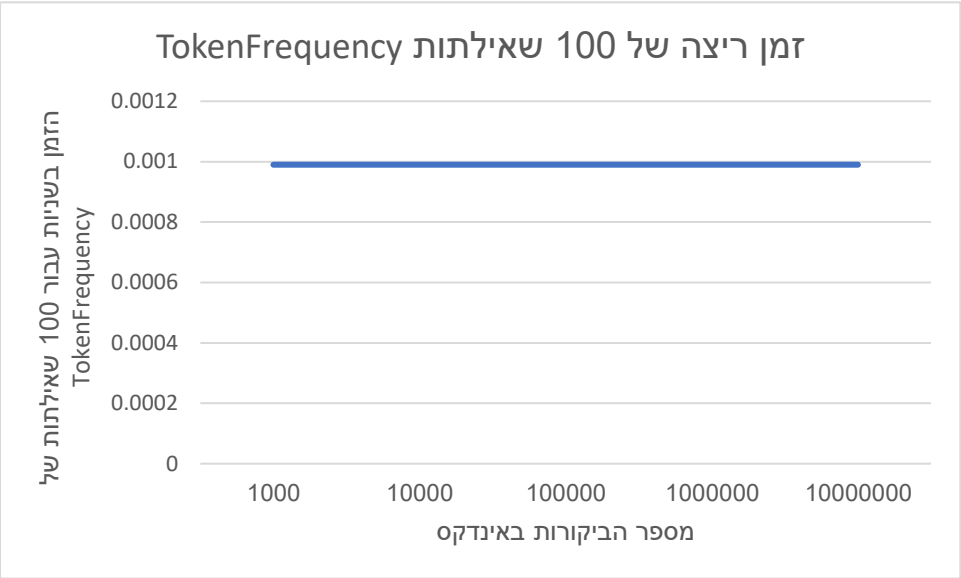


**מגישות: בינה רוזנטל ומיכל גבאי**



**מגישות: בינה רוזנטל ומיכל גבאי**




❖ ביצועי התוכנית תלויים במחשב עליו הרצנו, נציין את הנתונים החשובים:

- מערכת הפעלה: Windows 10 Pro

  סוג מערכת: מערכת הפעלה של 64 סיביות
- מעבד: Intel® Core™ i5-5250U CPU @ 1.60GHz
- גודל הזיכרון(RAM): 4.00 GB
- סוג הדיסק: APPLE SSD SM0256G