ניתוח החלק הראשון בפרוייקט שאילתות

מגישים:

אלכס לוינזון 308636141

הלל דודיאן 318593720

האינדקס מורכב משלשה קבצים:
נתאר אותם ואת תהליך היצירה שלהם
(התרשימים מופיעים בהמשך)

parsed_reviews.txt

קובץ זה הינו קובץ טקסט המכיל מידע שיש לשמור עבור כל סקירה, כפי שהוגדר בהוראות התרגיל.

לכל אחת מהסקירות מוקצים 6 שורות בקובץ:
id שהקצאנו למסמך (reviewId)
id של המוצר (productId)
helpfulness מונה
helpfulness מכנה
score
מספר המילים בטקסט

בקובץ זה נעשה שימוש בעת קריאה לחלק מהפונצקיות של המחלקה IndexReader. קובץ זה **לא נטען במלואו** בשום שלב בזיכרון לאחר יצירתו.

על מנת להוציא מידע ממנו, נגשים אך ורק לשורה הרלוונטית ומקבלים את המידע, על ידי הנוסחא:

(reviewId-1) * NUM_OF_REVIEW_ATTRIBUTES + X

כאשר X מספר התכונה שאותה רוצים לקבל (לפי הסדר שתואר לעיל)
NUM_OF_REVIEW_ATTRIBUTES – קבוע ששומר את מספר התכונות ששמרנו (כלומר 6)

index_dictionary.pkl
הקובץ שומר בתוכו אובייקט ונשמר בצורה בינארית. האובייקט מחזיק את המילון. בחרנו לממש את המילון בשיטת dictionary as string.

המילון מורכב משלשה מערכים, כל מערך בגודל מספר המילים ללא הכפילויות של כל הסקירות

בנוסף, המילון שומר במשתנה dictionary_string את כל המילים של הסקירות, ללא כפילויות ובצורה ממויינת. (המילים נשמרות ללא רווחים). פירוט אודות המערכים:

position_ptr – בכל תא במערך מופיע מיקום של המילה ב dictionary_string. כלומר עבור המילה הראשונה המיקום יהיה 0, ולכל מיקום שאחריו, נשמור את ערך המיקום הנוכחי ונצרף אליו את אורך המילה הנוכחית – מה שיהווה את המיקום של המילה הבאה. מערך זה בעצם מאפשר לנו לבצע חיפוש בינארי על ה dictionary_string ולהגיע למיקום של המילה המבוקשת.

קובץ זה נטען במלואו לזיכרון כאשר יוצרים מופע של IndexReader.

Word_freq – בכל תא במערך נשמור את סך כל המסמכים בהם מופיע המילה המתאימה ב dictionary string. גישה לתא כלשהו במערך מתאפשרת כתוצאה מחיפוש בינארי שמבוצע על המערך הקודם בעזרת ה position_ptr.

Bin_file_pos – בכל תא במערך מופיע המיקום של רשימת זוגות: (docId, Frequency) עבור אותה מילה. הסבר על רשימת הזוגות יופיע בהמשך. גישה לתא כלשהו במערך מתאפשרת כתוצאה מחיפוש בינארי שמבוצע על המערך הקודם בעזרת ה position_ptr.

doc_freq.bin

קובץ זה אחראי על הקידוד והפענוח של רשימת הזוגות (מזהה סקירה, שכיחות המילה בסקירה). רשימה זו נוצרת תוך כדי תהליך יצירת קובץ ה index_dictionary.pkl.

שיטת הקידוד בה אנחנו משתמשים:
לכל מילה יש רשימה של מזהיי הקבצים בה היא מופיעה (docId), ומספר הפעמים בהם היא מופיעה (frequency).

מימשנו שתי סוגים שונים של קידודים

עבור מספרי הdocId:
תחילה שמרנו את המרווחים בין מספרי מזהיי הסקירות, לאחר מכן השתמשנו בקידוד מסוג Varint Group, לדעתנו זו דרך יפה ונכונה יותר מאשר encoding varint

עבור מספרי ה - frequency:
השתמשנו בקידוד מסוג encoding varint וזאת מכיוון שאין אפשרות לשמור את המרווחים בינהם, מכאן שאין אפשרות להניח על גודל מקסימלי של מספר (ב -Varint Group ההנחה שהמספרים יתפסו עד 32 סיביות)
(frequency מסודר לפי מזהיי המסמכים, כלומר הן לא ממוינות ולכן לשמור את המרווחים ביניהם, לא יצמצם מקום)

בקובץ זה נעשה שימוש בעת קריאה לחלק מהפונצקיות של המחלקה IndexReader. קובץ זה **לא נטען במלואו** בשום שלב בזיכרון לאחר יצירתו.

ניתוח הגודל הצפוי בבתים של האינדקס:

parsed_reviews.txt

לכל סקירה יש 6 שורות בקובץ זה.

id שהקצאנו למסמך (reviewId) – עבור כמות גדולה של סקירות, ה- docid יהיה גבוה, לכל היותר עשר ספרות שנכתבו כתווים (עשרה בתים)
id של המוצר (productId) – 10 תווים בדיוק לכל סקירה (עשרה בתים)
helpfulness מונה – ספרה אחת שנכתבה כתו (בית אחד)
helpfulness מכנה – ספרה אחת שנכתבה כתו (בית אחד)
score – 3 תווים (3 בתים)
מספר המילים בטקסט – מקסימום 10 אלף מילים בסקירה (5 תווים = 5 בתים)
ירידת שורה בין כל תכונה – 6 תווים בכל סקירה (חוץ מבסקירה האחרונה, אבל זה זניח) – 6 תווים, כלומר 6 בתים

סך הכל עבור סקירה אחת:
6+5+3+1+1+10+10= 36 בתים

עבור 100 סקירות 3600 בתים
עבור X סקירות 36X בתים לקובץ זה

index_dictionary.pkl

הגודל תלוי ברמת השונות של כל המילים בכל הסקירות,
יש 3 מערכים כל אחד בגודל n, כאשר n הוא כגודל כמות המילים השונות שיש בכל קובץ הסקירות.
במקרה הגרוע, אם כל המילים שונות אז גודל המערכים יהיו בכמות כל המילים בכל הסקירות.
אם כל המילים דומות, כלומר יש רק מילה אחת, אז גודל המערכים יהיה תא אחד.

נניח קיימות n מילים שונות ואורך ממוצע של מילה הוא 5
אז במחרוזת הארוכה ששמורת את כל המילים, **5n בתים**

שלושה מערכים, כל מערך בגודל n, 8 בתים בתא (קבוע שפייטון מקצה)
3n*8=**24n**

מצביע למערך – 4 בבתים, יש שלושה מערכים, 4*3=**12 בתים**
מצביע לאובייקט – **8 בתים**

סך הכל, **29n+20** בתים

הערה:

הקובץ הוא קובץ בינארי, החישובים שלנו הם לפני ההמרה לבינארית
ולכן החישוב גדול מגודל הקובץ עצמו,
החישוב שלנו הוא הערכת סדר גודל.

doc_freq.bin

נמשיך עם ההנחה שקיימות n מילים שונות
אזי יהיו n רשימות, כל רשימה בנויה מ-
data, docId, freq, docId, freq, docId, freq, docId, freq, **data**, docId, freq, docId, freq
כלומר מידע על הבלוק באורך בית אחד
אחריו לכל היותר 4 זוגות של docId, freq
אם מילה מופיעה ביותר מ4 מסמכים יהיו עוד בתים של data
כלומר עבור מילה שאורך רשימת המסמכים שלה היא K – יהיו **k/4** (עיגול כלפי מעלה) בתים לdata

עבור כל מסמך – הנחנו בתרגיל שהוא עד 4 בתים
נזכיר כי, מספר המסמך בקובץ זה, הוא ההפרש ממספר המסמך שלפניו (חוץ מהראשון)
וכך הקטנו את מספר הבתים שמספר המסמך צריך
נניח כי לרוב docid מסתפק ב**בית אחד**

Freq – בקובץ זה, הוא מספר המופעים של המילה במסמך
קודד על ידי varint, כלומר 7 בתים למספר (כי הסיבית הראשונה היא מידע)
ולכן בבית אחד יכול להיות עד 128, כלומר אם מילה מופיעה במסמך עד 128 פעמים הFreq יסתפק בבית אחד.
עבור מסמכים עם טקסט קטן, כמו בתרגיל זה, אפשר להניח שהfreq יהיה **בית אחד**.

לכן עבור מילה שאורך רשימת המסמכים שלה היא K:

k/4 (עיגול כלפי מעלה) + 2k בתים

אם n מספר המילים השונות ו-K הוא המספר הממוצע של אורך רשימת המסמכים למילה,
אז גודל הקובץ הוא: **n*(K/4+2K)** בתים

תרשימים המתארים את מבנה האינדקס

נתאר את מבנה האינדקס על קובץ הסקירות הבא:

```
product/productId: B000GX
review/userId: A2LGWBIT9WCTVA
review/profileName: D. Gesswein
review/helpfulness: 0/0
review/score: 5.0
review/time: 1173312000
review/summary: Excellent chip!
review/text: we like to learn java a big fan

product/productId: B000G6
review/userId: A3TX8RH943OLBM
review/profileName: annie
review/helpfulness: 3/5
review/score: 2.0
review/time: 1214006400
review/summary: salt and vinegar chips
review/text: a fast learning skills a big fun
```

כתוצאה מניתוח הקובץ הנ"ל מתקבלים קצבי האינדקס הבאים:

parsed_reviews.txt

```
1
B000GX
0
0
5.0
8
2
B000G6
3
5
2.0
7
```

Index_dictionary.pkl

position_ptr	word_freq	bin_file_pos
1	2	0
2	2	5
5	1	10
8	1	13
12	1	16
15	1	19
19	1	22
24	1	25
32	1	28
36	1	31
42	1	34

Bin_file_pos.bin

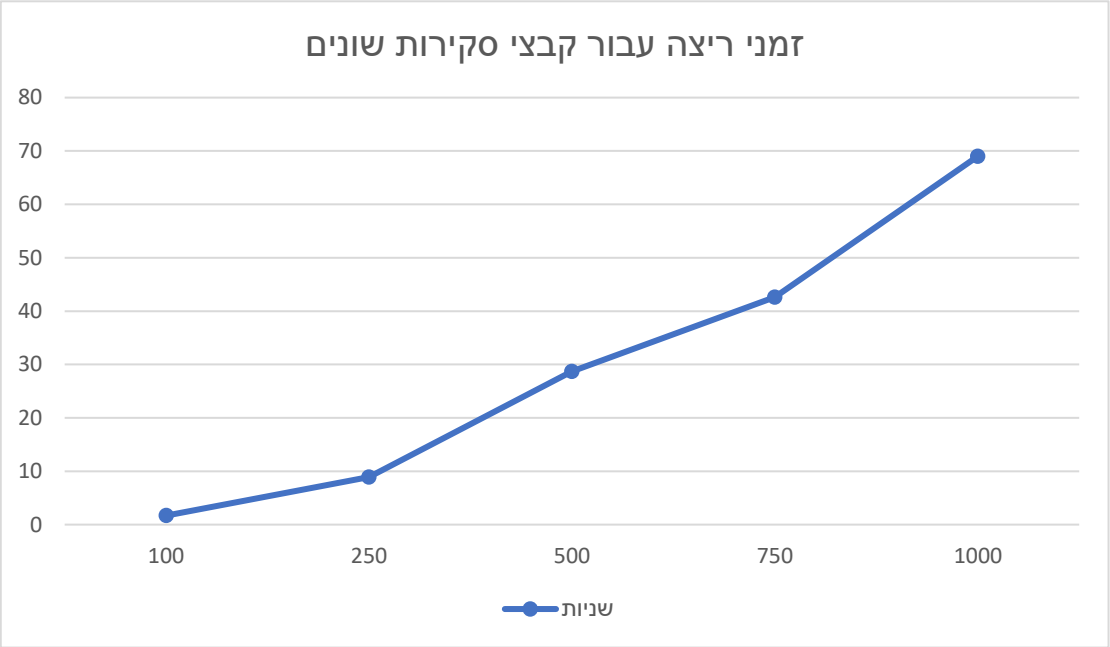
הערה: הקובץ המקודד לבינארית **אינו** מכיל רווחים ושורות חדשות
אך נציג אותו בצורה כזו לשם נוחות ההסברים

```
00000000 00000001 10000001 00000001 10000010
00000000 00000001 10000001 00000001 10000001
00000000 00000001 10000001
00000000 00000010 10000001
00000000 00000010 10000001
00000000 00000001 10000001
00000000 00000001 10000001
00000000 00000010 10000001
00000000 00000001 10000001
00000000 00000010 10000001
00000000 00000001 10000001
00000000 00000001 10000001
```

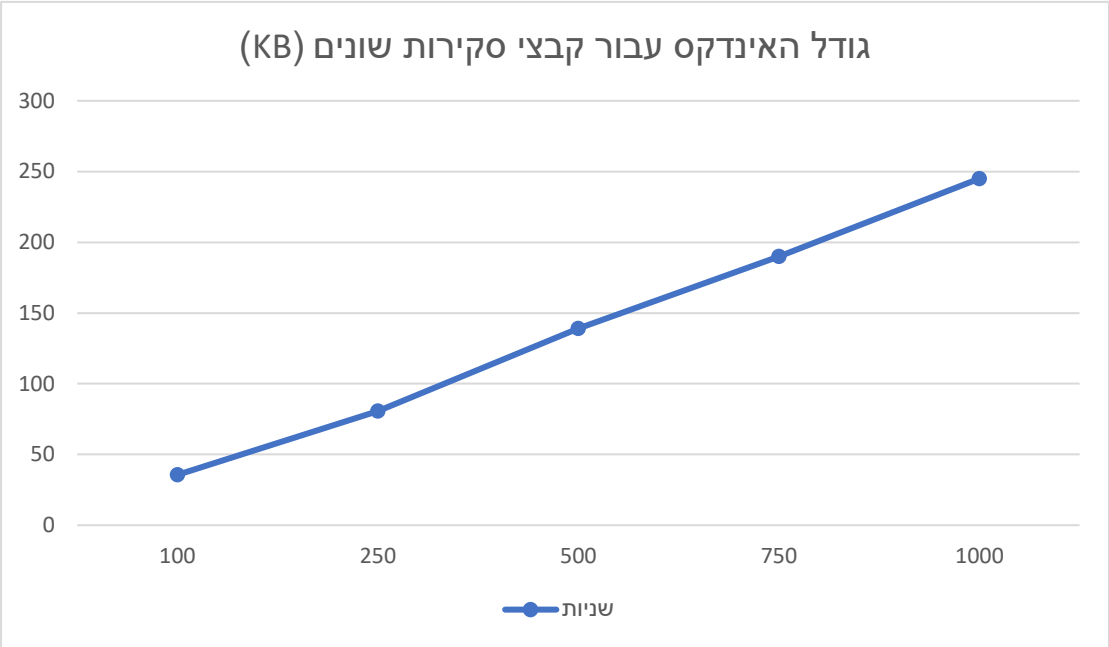
בכל שורה מוצגת רשימה עבור מילה אחרת
כל רשימה מחולקת לבלוקים של 4 כך שהבית הראשון בכל בלוק הוא מידע על כמות הבתים בבלוק אחריו מופיעים זוגות של: מספר קובץ בו המילה מופיעה וכמות המופעים שלה
כאשר כל מספר קובץ (חוץ מהראשון) הוא מרווח מהקובץ שלפניו (חוץ מהראשון)
בתמונה רואים רשימות עבור 12 מילים,
נפרט על השורה הראשונה:
הבית הראשון כולו אפסים, משמע שכל מספר קובץ זקוק לבית אחד בלבד
הבית השני הוא מספר הקובץ הראשון שבו המילה מופיעה – 1
הבית השלישי הוא מספר השכיחות של מילה זו בקובץ מספר 1 והוא מקודד בסוג הקידוד **varint**
כלומר יופיע 1 בביט הראשון אם בשביל המספר הזה נזדקק לבית 1 (0 אחרת)
ולכן השכיחות הכתובה בבית זה היא 1
הבית הרביעי זה המרווח של מספר הקובץ השני ממספר הקובץ – כתוב שם 1, הקובץ הראשון היה 1 ולכן מספר הקובץ השני הוא 2
הבית החמישי – הוא מספר השכיחויות בקובץ השני, מקודד ע"י **varint**, ולכן השכיחות שכתובה בבית זה היא 2.

נספח:

תיאור התרגיל על ידי גרפים



בדקנו את זמן הריצה של התוכנית על 100 סקירות ואז על 250 וכן הלאה



בדקנו את גודל סך כל הקבצים של התוכנית על 100 סקירות ואז על 250 וכן הלאה
ניתן לראות שקיים יחס ליניארי בין גודל האינדקס לבין מספר הסקירות