**דוח מנוע – חלק ב'**

1. **הסבר על אופן פעולת המנוע:**

**Document**: במחלקה זו הוספנו את השדה "title" שיעזור לנו לדרג את המסמכים הרלוונטיים בצורה הטובה ביותר. מסמך בו הכותרת מכילה חלק או את כל מילות השאילתא יקבל דירוג גבוה יותר ויהיה רלוונטי יותר.

הוספנו משתנה ששומר את 10 המילים הראשונות במסמך. שמרנו משתנה זה מכיוון שאם מילות השאילתא מופיעות בהתחלה אז כנראה שהמסמך הוא יותר רלוונטי.

**MergeConnectedWords**: מחלקה זו ממיינת לקובץ אחד את כל זוגות המילים העוקבים. מחלקה זו משמשת אותנו להשלמה אוטומטית במסך הראשי, כאשר המשתמש מקליד מילה ורווח הוא יקבל את 5 האופציות הכי נפוצות במאגר להשלמה אוטומטית.

MergeInEnd**:** פונקציה אשר מאחדת בין 2 קבצים זמניים ויוצרת קובץ חדש מאוחד של השניים כך שהוא ממוין.

Merge**:** פונקציה זוקוראת 2 קבצים זמניים במקביל ומשווה בין המילים שבכל שורה. במידה ומדובר בשורות בעלי צמדי מילים זהים, הפונקציה קוראת לMergeRow על מנת לאחד את השורות.

MergeRow**:** פונקציה זו מאחדת בין שתי שורות שהינן בעלי צמדי מילים זהים.

checkArrayList**:** פונקציה זו בודקת ומרדדת את קובץ "connectedWords" כך שיכיל רק את 5 ההשלמות הטובות ביותר עבור מילה במאגר.

getTheDic: פונקציה זו בונה מילון חדש עבור כל צמדי המילים שקיימים במאגר המסמכים אשר אותם היא קוראת מקובץ הסופי של הצמדים "ConnectedWordsNew" ומעלה אותו לזיכרון.

**Parse** : השינויים שביצענו במחלקה זו:

1. במחלקה שמרנו את המילים ואת המיקום שלהם בlinkedlist כך שלפני מעבר לindexer, נבדוק מהם צמדי המילים הקיימים במסמכים שנבדקו באיטרציה, ובמידה והמיקומים של צמד מילים הינו אחד אחרי השני נשמור את הצמד ואת מס' המופעים שלו במסמך מסויים.

אחרי מעבר על כל המסמכים שנבדקו באיטרציה מסויימת ויצירת מילון צמדים של מילים והמסמכים בהם הם הופיעו, נרשום לקובץ זמני את הפרטים על צמד מילים מסויים שהופיע במילון, נבצע זאת על כל המילים שמופיעים במילון.

1. ביצענו חלוקה בין הרצת parser עבור המסמכים (כמו שבוצע בחלק א') לבין הרצה עבור שאילתות.

**RankedDoc:** מחלקה זו יוצרת אובייקט שמכיל את שם המסמך והדירוג שלו. במחלקה זו נעזר בעת דירוג המסמכים במחלקה Ranker.

**Ranker:**

מחלקה זו אחראית לדרג את המסמכים ולהחזיר את 50 המסמכים הרלוונטים ביותר לשאילתא.

הפונקציות הקיימות במחלקה:

GetSemanticFromQuery- פונקציה זו עוברת על מילות השאילתא. כל מילה היא שולחת לפונקציה GetSemantic שמחזירה רשימה של המילים שדומות בקונספטים.

GetSemantic- פונקציה זו מקבלת מילה ומחזירת רשימה של מילים שבעלי קונספט זהה. מילים אלו יהיו מסוג פועל ושם עצם.

getTheDocumnet- פונקציה זו מחלצת עבור כל מילה בשאילתא את המסמכים בהם היא הופיעה ומעדכנת במילון של קבצים את הקובץ הרלוונטי ואת הפעמים בהם הופיעה המילה בקובץ. (מילון זה יקרא בשם "m\_query\_relvent\_doc").

CalculatingDocumentsBM25- פונקציה זו מחשבת את אלגוריתם Bm25 עבור כל מסמך שמופיע במילון של המסמכים הרלוונטים בשאילתא (שנמצאו בפונקציה getTheDoc).

getTheRelevantDoc\_Title- פונקציה זו בודקת עבור כל מילה בשאילתא, האם היא הופיעה בכותרת המסמך שהוגדר כרלוונטי. היא מוסיפה את שם המסמך ואת מס' המילים מהשאילתא שמופיעות בכותרת למילון בשם "query\_in\_title". פונקציה זו מחזירה את המילון.

CalcultaingDocumentsCosSim- פונקציה זו מחשבת את אלגוריתם CosSim עבור כל מסמך שמופיע במילון של המסמכים הרלוונטים בשאילתא.

CalculateFinalRanking- פונקציה זו מחשבת את הדירוג הסופי של מסמך. היא מקבלת את הדירוגים שהתבצעו בפונקציות לחישוב. (coSim,title,BM25,Semantic) ומשקללת את כל החישובים לדירוג סופי של המסמך ע"פ משקל שניתן לכל אלגוריתם. פונקציה זו מחזירה מילון של 50 מסמכים הרלוונטיים ביותר לשאילתא ואת הדירוג של כל מסמך ממוינים מהמסמך הרלוונטי ביותר לזה שפחות.

WriteToFile\_rankedDoc **–** פונקציה זו כותבת לקובץ "queries\_ranked" את 50 המסמכים הרלוונטים ביותר לשאילתא ע"פ הפורמט לתוכנת הtrec evel להערכת התוצאות.

getStart10- פונקציה אשר בודקת האם לפחות אחת ממילות השאילתא מופיעה ב10 המילים הראשונות במסמך ובונה מילון שמכיל עבור כל מסמך את שמו ואת מספר המילים מהשאילתא שמופיעות בתחילת המסמך.

**Searcher:**

מחלקה זו אחראית על מס' דברים:

* מאפשר השלמה אוטומטית לשאילתת המשתמש.

מחלקה זו נעזרת במילון צמדי המילים שהועלו לזיכרון לאחר יצירת המילון או לאחר לחיצה על LOAD. בעזרת מילון צמדי המילים, מתאפשרת ההשלמה האוטומטית במסך הראשי למשתמש בעת הקלדת שאילתא.

* מאפשר חיפוש על פי שפת המסמך במקור.

במידה והמשתמש סימן בחלון הראשי מציאת מסמכים ע"פ שפות מסוימות, רשימת השפות הללו תישלח למחלקה Searcher אשר מעבירה את הרשימה לRanker ושם מתחשבת ברשימה זו.

* הSearcher מקבל את השאילתא ועושה לה פירסור במחלקת "parse" (מעביר את השאילתא את כל החוקים שכתבנו). פונקציה זו הינה פונקצית sentToParser.
* מעביר את השאילתא למחלקה Ranker שדואגת להחזיר את 50 המסמכים הרלוונטים ביותר לשאילתא.

**WordsConnected**: מחלקה זו יוצרת אובייקט שמכיל את המילה והמיקום שלה במסמך, מחלקה זו עוזרת לנו להשלמה האוטומטית במסך הראשי.

**ArrayConnected**: מחלקה זו יוצרת אובייקט שמכיל את שם המילה ומילון (שם המסמך ומס מופעים של המילה במסמך). נשתמש באובייקט זה עבור שמירת צמדי המילים שמופיעים במסמך.

* **הסבר על ההשלמה האוטומטית:**

ביצענו מספר פעולות על מנת לבצע את ההשלמה האוטומטית:

1. שמרנו בקבצים זמניים את כל צמדי המילים שעבורם נשמר המסמכים בהם הופיעו ואת מספר המופעים שלהם בכל מסמך שהיו במסמכים. צמדי המילים שנשמרו הינם מילים שהופיעו אחת אחרי השנייה – ללא stop words.

בסה"כ נוצרו 50 קבצים זמניים של צמדי מילים.

1. איחדנו בין כל הקבצים הזמניים לקובץ אחד שמכיל את כל צמדי המילים ואת מספר המופעים הכולל שלהם בכל המסמכים לקובץ סופי שנקרא "connected Words".
2. ביצענו רידוד של צמדים, כלומר עבור כל מילה שהופיעה במקום הראשון בצמד, בדקנו את כל האפשרויות עבורה.

ובסופו של דבר, שמרנו בקובץ נפרד את 5 ההשלמות עבור המילה שהופיעו הכי הרבה פעמים. לדוגמה, עבור המילה dog הופיעו 10 צמדים אפשריים שמתחילים במילה זו ועל כן שמרנו רק את ה5 הצמדים שהופיעו יותר פעמים מאשר שאר ההשלמות

הרציונל בהבאת אלגוריתם זה הוא השלמה אוטומטית של צמד מילים שהופיעו הכי הרבה פעמים ובסבירות גבוהה המשתמש יקבל עבורם תוצאות ואף מס' גבוה יותר של מסמכים.

במידה והיה קיים נתונים אודות חיפושים קודמים של משתמשים אחרים במנוע היה ניתן להחזיר השלמות אחרות או נוספות שכנראה המשתמש גם יחפש אותם וירצה לקבל עבורם מסמכים רלוונטיים.

בנוסף, במידה והיה לנו מידע קודם על המשתמש- העדפותיו, חיפושים קודמים שלו, תחומי עניין בתחום מסוים היינו יכולים להחזיר השלמות שכנראה רלוונטיות יותר אליו.

* **אלגוריתם הדירוג:**

אלגוריתמים אשר מימשנו בעבודה ופרטים נוספים שהתחשבנו בדירוג:

1. BM25 כפי שהתבקשנו לעשות.
2. אלגוריתם לטיפול סמנטי שמתחשב במילים בעלי סמנטיקה דומה.
3. CosSim
4. התחשבות בכותרת של מסמך.
5. התחשבות ב10 מילים הראשונות במסמך.

פירוט אודות המרכיבים:

**BM25**: באלגוריתם זה הגדרנו את הפרמטרים k1,k2,b בצורה הבאה: K1=1.1 ,K2=1000, b=0

אלגוריתם זה נותן דירוג עבור כל מסמך שלפחות מילה אחת מהשאילתא הופיעה בו.

לאלגוריתם זה נתנו את החשיבות הגבוהה ביותר בחישוב הדירוג הכולל מכיוון שהינו נותן את הדיוק הטוב ביותר למציאת מסמכים רלוונטיים.

**התחשבות במילים בעלי סמנטיקה דומה**: נעבור על המילים שמופיעים בשאילתא.

עבור כל מילה בשאילתא, נשלח אותה לאלגוריתם ונקבל את כל המילים בעלי ההקשר והמשמעות הדומה. המילים שהתקבלו הינם מסוג שם עצם, שם תואר, פעלים ותואר הפועל.

אך אנו השתמשנו רק בסוגים פעלים ושם עצם, כיוון שראינו שלמילים אלו יש סמנטיקה קרובה יותר למילים שבשאילתות.

אנו נתנו לאלגוריתם סמנטי זה דירוג נמוך יחסית, כיוון שיכולים להיות מצבים בהם האלגוריתם יחזיר מילים שאולי דומים למילות השאילתא אך בהקשרים שונים, יהיו בעלי משמעות אחרת לגמרי מהשאילתא המקורית.

**CosSim**: אלגוריתם זה מתייחס לנוסחה : cosSim =

המשקל שנתנו לכל מילה יעשה ע"י ציון בינארי (1 אם מילה מופיעה או 0 אם לא) .

בנוסף, חישבנו עבור כל מילה שנמצאת בשאילתא את חישוב הIDF שלה.

בעת ביצוע החישוב הסופי, עבור כל מילה שמופיעה גם במסמך וגם בשאילתא, נכפול את משקל המילה שהינו 1 (שהינו זהה בין השאילתא לבין משקלו במסמך) בערך הIDF שלה. נחבר את כל ערכי המילים שהתקבלו.

נחלק את הסכום הסופי שהתקבל במכפלה בין אורך המסמך ואורך השאילתא.

**התחשבות בכותרת:** נעבור על המילים שמופיעים בשאילתא. עבור כל מילה נבדוק האם היא הופיעה בכותרת של המסמכים שבהם הופיעה לפחות אחת ממילות השאילתא. במידה וכן ניתן לה ציון. במידה ובכותרת הופיעה יותר ממילה אחת ניתן ערך גבוה למסמך זה.  
אנו סבורות שבמידה ויותר ממילה אחת מהשאילתא שמופיעה בכותרת זה אומר שהמסמך הינו מאוד רלוונטי ועל כן נחזיר אותו כאחד מהמסמכים הרלוונטיים לשאילתא.

**התחשבות ב10 מילים ראשונות במסמך:** עבור כל מסמך שמרנו את 10 המילים הראשונות. בעת ביצוע החישובים בRANKER נשלח את השאילתא עם המסמכים שהוגדרו כרלוונטיים לפונקציה "getstart10" אשר בודקת אם לפחות אחת ממילות השאילתא הופיעה בתחילת המסמך. במידה וכן, ניתן משקל נוסף עבור מסמך זה.

אנו סבורות שבמידה ולפחות אחת ממילות השאילתא מופיעה בהתחלה אז המסמך הינו רלוונטי, על כן נחשב מדד זה.

* **הסבר על האלגוריתם הסמנטי:**

בחלק זה השתמשנו בקוד פתוח, שנמצא באתר Github.

בכתובת : https://github.com/zacg/WordNetAPI

הוספנו DLL חיצוני לפרויקט.

האלגוריתם הסמנטי הוא בעצם אלגוריתם של קונספטים. עבור כל מילה יוחזר המילים בעלי ההקשר והמשמעות הדומה ביותר.

עבור כל מילה בשאילתא, נשלח אותה לאלגוריתם ונקבל את כל המילים בעלי ההקשר והמשמעות הדומה. המילים שהתקבלו הינם מסוג שם עצם, שם תואר, פעלים ותואר הפועל.

אך אנו השתמשנו רק בסוגים פעלים ושם עצם, כיוון שראינו שלמילים אלו יש סמנטיקה קרובה יותר למילים שבשאילתות.

התחשבנו במילים אלו בדירוג הסופי של המסמך, בכך שעבור המילים עם הסמנטיקה הדומה חישבנו את BM25 ולאחר מכן הוספנו את התוצאה לשקלול הסופי של דירוג מסמך**.**

* חישוב ושקלול משקלי הדירוג:

קבענו את משקלי הדירוג אחרי הרבה שינויים במשקל שנתנו עבור כל פרמטר בחישוב.

1. לBM25 נתנו את המשקל הרב ביותר במידה ולא מופיע יותר מ2 מילים מהשאילתא בכותרת.
2. במידה ולפחות 2 מילים מהשאילתא מופיעות בכותרת יינתן ציון לtitle שהינו 10 כך שהערך הסופי יהיה 1.(כשמכניסים לנוסחא).
3. ראינו כי מילים סמנטיות אינם עוזרות להעלאת הדירוג לקובץ וכי אף גורע. בנוסף, חיפוש עבור כל המילים הסמנטיות בקובץ לוקח זמן רב.

את שאר המשקלים, קבענו ע"פ ניסיונות רבים בשינויים ועבור המשקלים הנתונים למטה קיבלנו את מספר המסמכים הרלוונטי הטוב ביותר.

Ranking (D,score) = title \* 0.1 + BM25 \* 0.8 + cosSim\* 0.1 + Semantic\* 0 + start10words \* 0.2

**הסבר על הנתונים בקובץ ה- posting ובמילון**

במימוש שלנו שמרנו קובץ עזר ששמו "Dictionary" שהינו מכיל את הterms שנמצאים בזיכרון. (בעת טעינת התוכנית אנו קוראים מקובץ זה את ה terms ובונים מילון שמכיל את הנתונים עבור כל term, לכל term שמור tf,df ).

בנוסף לקובץ זה קיים לנו קובץ הפוסטינג שבו נשמר אך ורק השרשור של כל term , באיזה מסמך הופיע וכמות המופעים באותו מסמך.

כאשר נחפש מילה כלשהי, תחילה נחפש אותה בקובץ הdictionary ובמידה והגענו למילה נחפש את השרשור שלה בקובץ הפוסטינג ע"פ השורה שהינה זהה בין שני הקבצים. (השרשור של מילה יופיע בשורה שבה הterm יופיע בקובץ הdictionary).

השתמשנו בקבצים למעלה כאשר רצינו למצוא עבור כל מילה בשאילתא את המופעים שלה שקיימים במאגר.

על כן, בRanker עבור כל מילה בשאילתא מצאנו את המסמכים בהם היא מופיעה במאגר. על ידי שרשור זה מצאנו מסמכים אשר רלוונטים לשאילתא ובהם נשתמש לחישוב הדירוג עבור כל שאילתא.

קבצי עזר נוספים ששמרנו:

1. קובץ doc – קובץ זה מכיל את הפרטים על המסמכים שנמצאים במאגר. עבור כל מסמך שמרנו את הפרטים הבאים: שם המסמך, אורך המסמך, שפת המסמך, המילה השכיחה ביותר במסמך (במידה ואין מילה כזו יהיה השם "NoCommon") ומספר המופעים שלה, כמות המילים הייחודיות במסמך, כותרת המסמך.

במסמך זה נעזר כאשר נרצה לחשב את האלגוריתמים לדירוג מסמך.

1. קובץ ConnectedWords – קובץ זה מכיל עבור כל מילה את 5 ההשלמות הנפוצות ביותר עבורה במאגר ואת מס' הפעמים. בקובץ זה נעזר עבור ההשלמה האוטומטית שתציג למשתמש במקרה שירשום מילה אחת, את 5 האופציות הנפוצות ביותר במאגר.

* **שימוש בקוד פתוח:**

השתמשנו בקוד פתוח עבור האלגוריתם הסמנטי, לקחנו אותו מאתר GITHUB.

בכתובת : https://github.com/zacg/WordNetAPI

השתמשנו בקוד פתוח זה על מנת לדרג את המסמכים כך שיוחזרו המסמכים הרלוונטים ביותר לשאילתא.

במחלקה RANKER כאשר דירגנו את השאילתא, שלחנו בכל פעם מילה אחת מהשאילתא לפונקציה שמחזירה את המילים בעלי הסמנטיקה הדומה וקיבלנו חזרה את כל המילים בעלי קונספט זהה.

לאחר מכן שלחנו את המילים עם הקונספט הדומה לפונקציית חישוב BM25 שתעזור לנו לדרג את השאילתא.

1. **הערכה של המנוע:**

עם Stemming:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| מס' שאילתא | מילות השאילתא | Precision לשאילתא | Recall לשאילתא | Precision@5 | Precision @15 | Precision@30 | MAP |
| 11 | Space Program | 0.16 | 0.086 | 0.2 | 0.2 | 0.1667 | 0.0195 |
| 12 | Water Pollution | 0.56 | 0.122 | 0.6 | 0.7333 | 0.5667 | 0.0799 |
| 82 | Genetic Engineering | 0.3 | 0.272 | 0.2 | 0.2 | 0.3333 | 0.0885 |
| 118 | International Terrorists | 0.16 | 0.0246 | 0 | 0 | 0.2 | 0.0037 |
| 142 | Impact of Government Regulated Grain Farming on International | 0.8 | 0.049 | 0.6 | 0.8 | 0.7667 | 0.0374 |
| 189 | Real Motives for Murder | 0.28 | 0.0239 | 0.2 | 0.2 | 0.2 | 0.0059 |
| 341 | Airport Security | 0.12 | 0.285 | 0 | 0.1333 | 0.1333 | 0.0378 |
| 347 | Wildlife Extinction | 0.66 | 0.44 | 0.6 | 0.6667 | 0.5667 | 0.2743 |
| 367 | piracy | 0.2 | 0.106 | 0.2 | 0.2 | 0.1333 | 0.0249 |
| 374 | Nobel prize winners | 0.14 | 0.318 | 0 | 0.0667 | 0.1667 | 0.0484 |
| 399 | oceanographic vessels | 0.28 | 0.233 | 0 | 0.2 | 0.2333 | 0.0556 |
| 410 | Schengen agreement | 0.36 | 0.947 | 0.2 | 0.2667 | 0.2667 | 0.2720 |
| 416 | Three Gorges Project | 0.34 | 0.607 | 0.4 | 0.4 | 0.4 | 0.2456 |
| 431 | robotic technology | 0.52 | 0.577 | 0.6 | 0.6 | 0.5 | 0.3010 |
| 450 | King Hussein, peace | 0.26 | 0.071 | 0.2 | 0.2 | 0.2667 | 0.0165 |

MAP: 0.100733

בלי Stemming:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| מס' שאילתא | מילות השאילתא | Precision לשאילתא | Recall לשאילתא | Precision@5 | Precision @15 | Precision@30 | MAP |
| 11 | Space Program | 0.18 | 0.097 | 0.2 | 0.2 | 0.1667 | 0.0188 |
| 12 | Water Pollution | 0.58 | 0.127 | 0.6 | 0.667 | 0.6 | 0.0831 |
| 82 | Genetic Engineering | 0.42 | 0.381 | 0.2 | 0.2667 | 0.4667 | 0.3818 |
| 118 | International Terrorists | 0.42 | 0.064 | 0.4 | 0.2667 | 0.3667 | 0.0648 |
| 142 | Impact of Government Regulated Grain Farming on International | 0.74 | 0.0457 | 0.6 | 0.6667 | 0.6667 | 0.0316 |
| 189 | Real Motives for Murder | 0.44 | 0.0376 | 0.4 | 0.4 | 0.4 | 0.0177 |
| 341 | Airport Security | 0.1 | 0.238 | 0 | 0.0667 | 0.1 | 0.0274 |
| 347 | Wildlife Extinction | 0.66 | 0.44 | 0.2 | 0.5333 | 0.5667 | 0.2531 |
| 367 | piracy | 0.2 | 0.106 | 0.2 | 0.2 | 0.1333 | 0.0249 |
| 374 | Nobel prize winners | 0.08 | 0.1818 | 0.4 | 0.2 | 0.1 | 0.1043 |
| 399 | oceanographic vessels | 0.24 | 0.2 | 0.2 | 0.2667 | 0.2333 | 0.0477 |
| 410 | Schengen agreement | 0.36 | 0.94 | 0.2 | 0.2667 | 0.2667 | 0.2735 |
| 416 | Three Gorges Project | 0.38 | 0.678 | 0.4 | 0.4667 | 0.4667 | 0.3301 |
| 431 | robotic technology | 0.14 | 0.155 | 0.2 | 0.1333 | 0.1333 | 0.0434 |
| 450 | King Hussein, peace | 0.34 | 0.09 | 0.4 | 0.2667 | 0.3333 | 0.0323 |

MAP: 0.115633

1. **סיכום:**

האתגרים בפרוייקט:

1. להגיע לכמה שיותר מסמכים רלוונטים בדומה לתוצאות שהתקבלו במסמך "qrels". עקב מאגר מסמכים מאוד גדול קשה להחזיר 50 מסמכים רלוונטים לשאילתא והבדל קטן באלגוריתם יכול לשנות בכל פעם את סדר גודל המסמכים הרלוונטים שיחזרו.
2. מציאת דרך להעלות ולפרסר את המסמכים בזמן הנמוך ביותר עקב כמות מסמכים גדולה. זמן העלאת הקבצים הינו רב ועל כן יש למצוא שיטות או להשתמש בקבצים מסוג מסויים עבור עמידה בזמנים. לדוגמה בהתחלה חשבנו להשתמש בקבצי XML , שיטה שעל ידיה ניתן למצוא ולהוסיף נתונים בצורה המהירה ביותר, בסופו של דבר לא השתמשנו בשיטה זו כיוון שלא עמדה בהוראות הפרוייקט. (לא ניתן לעדכן ולהוסיף למסמך בשיטה זו מבלי לטעון את כל הקובץ לזיכרון).

בעיות שנתקלו בהם:

1. מציאת הפרמטרים הטובים ביותר עבור אלגוריתם BM25 וכך להוביל לתוצאות טובות יותר. שמנו לב שבכל שינוי קל בפרמטרים יש הבדל ניכר במס' המסמכים הרלוונטיים שחוזרים. לאחר ניסיונות רבים, הגענו לתוצאות לעיל שהחזירו לנו את התוצאות הטובות ביותר.
2. לפני שביצענו רידוד למציאת 5 ההשלמות הנפוצות ביותר למילה, ניסינו להעלות את הקובץ ללא רידוד, דבר שלא צלח עקב כך שהקובץ היה גדול מדי. ולכן החלטנו לבצע רידוד ולשמור רק את 5 האופציות הנפוצות ביותר ולא להעלות את כל הקובץ לזיכרון ורק לאחר מכן לשלוף את 5 האופציות הנפוצות ביותר.