תיאור הקוד:

הקוד מחולק לשני חלקים עיקריים כפי שמתואר בדרישות התרגיל:

1. בניית ה-inverted index, מתבצע בפונקציה אשר מקבלת נתיב לתיקיית הקבצים –

def build\_inverted\_index(path):

* כל קובץ xml שנמצא בתיקייה עובר פרסור בפונקציה -

def parse\_one\_xml\_file(doc):

תהליך הפרסור עובר רשומה רשומה (של RECORD) ומחלץ את כל הטקסט שמצאנו להיות רלוונטי כלומר את האלמנטים הבאים: TITLE, EXTRACT, ABSTRACT, TOPIC, RECORDNUM.

* לאחר מכן כל הטקסט שחולץ עובר תהליך של ניקוי על מנת לבצע ספירה ושמירה של המילים באופן מסודר. תהליך זה מתבצע בפונקציה-

count\_word\_in\_text(record\_number, record\_text)

* כחלק מתהליך הניקוי אנו מסירים stopwards, מסירים מספרים ותווים מיוחדים, ומבצעים גם steaming למילים.
* מבצעים שמירה במילון לכל מילה כמה פעמים היא הופיעה במסמך.
* בסוף התהליך הנ"ל עבור כל מסמך ומסמך, מתבצעת קריאה לפונקציה המחשבת את ציוני ה-TF-IDF- לפי נרמול במילה שמופיעה הכי הרבה פעמים במסמך (כפי שלמדנו בהרצאה), ושמירה של החישובים במילון.

def update\_tfidf\_scores():

* ולבסוף שמירה של מבנה הנתונים לקובץ כנדרש.

1. החזרת מסמכים רלוונטים לשאילתה:
   * מתבצעת טעינה של המילון מהקובץ לזיכרון
   * מתבצעת קריאה לפונקציה-

def print\_relevant\_documents(query, print\_to\_file=True):

שבה מבצעים קריאה לפונקציה-

def build\_query\_vector(query):

שמבצעת ניקוי והכנה של השאילתה כפי שעשינו למסמכים בשלב 1.

* לאחר מכן מבצעים חישוב cosim בין השאילתה לבין כל מסמך ומסמך על מנת לדרג את המסמכים הרלוונטים (כפי שראינו בהרצאה).
* ולבסוף הדפסה של המסמכים הרלוונטים לפי סף ציון של 0.08 ב- cosim שחושב. סף זה מקסם עבורנו את ה-F score ואפשר לנו להחזיר את התוצאות הטובות ביותר.

מבנה ה- inverted index:

עיקר המידע הנשמר במבנה הנתונים הוא המילון בעל המבנה:

* מילון הממפה בין token לבין מילון נוסף שהמפתחות שלו הם מספרי המסמכים בהם המילה מופיעה.
  + המילון הפנימי ממפה בין מספר מסמך לבין מילון נוסף שמחזיק:
    - כמה פעמים המילה הופיעה במסמך, ואת ציון ה-tf-idf.

בנוסף מבנה הנתונים מחזיק מידע כללי על המסמכים תחת המפתח DOC\_INFO:

* AMOUNT\_OF\_DOCS\_IN\_CORPUS- כמה מסמכים יש.
* DOC\_WEIGHT- מיפוי בין כל מסמך לבין המשקל שלו.