

תיעוד הקוד

קובץ vsm_ir.py

כאשר תכתב פקודה בשורת הפקודה, התוכנית שלנו תבדוק האם הארגומנט הראשון הינו "create_index" או "query" ובהתאם תריץ את החלק הרלוונטי.

אם הארגומנט הינו create_index, נריץ את הפונקציה create_index().

אם הארגומנט הינו query, נריץ את הפונקציה query().

הפונקציה create_index():

פונקציה זו בונה אובייקט בשם InvertedIndexDictionary, ובונה בעזרתו מילון שבנוי בצורה הבאה:

1. מפתח "TF" שערכו מילון בו key הוא מילה, והvalue הוא מילון בעצמו שבתוכו key הוא המסמך והvalue הוא כמה פעמים המילה מופיע במסמך הנ"ל.
2. מפתח "len_by_doc_name" שערכו הוא מילון בו key הוא המסמך והvalue הוא כמות המילים במסמך הנ"ל.
3. מפתח "normal_IDF" שערכו הוא מילון בו key הוא מילה והvalue הוא ערך IDF שלה כפי שלמדנו בכיתה.
4. מפתח "BM25_IDF" שערכו הוא מילון בו key הוא מילה והvalue הוא ערך IDF שלה כפי שלמדנו בכיתה.

קובץ "InvertedIndexDictionary"

מטרת קובץ זה היא ליצור את הקובץ vsm_inverted_index.json שמכיל את הערכים שתוארו לעיל (המילון שנוצר מcreate_index). הפונקציות שלו הן:

1. בנאי – מאתחל מספר שדות בסיסיים של האובייקט (stop_words, dict, path_to_xml_dir, count_of_docs)
2. Build_inverted_index – פונקציה שמכניסה לתוך השדה dict את התוכן של המילון שמתואר לעיל. אנו עוברים על כל קובץ בתיקיית הקבצים הרלוונטית, מוציאים ממנו את ערכי Inverted Index, עוברים אותם ומוסיפים אותם להתאם למילון. בסוף הפונקציה אנו גם מחשבים את ערכי IDF לtf-idf ולbm25 בהתאם.
3. Get_inverted_index_of_file – פונקציה זו מקבלת שם של קובץ שעליו היא תעבוד, והיא בונה מילון של Inverted Index עבור כל documents שנמצאים בקובץ הזה. עוברים על documents אחד אחד, ומוסיפים את ערכי TF והdocument size הרלוונטיים.
4. Get_tokenized_words_from_document – מקבל document ומוציא את כל המילים הרלוונטיות ממנו (כל מה שתחת TITLE ABSTRACT או EXTRACT) וקורא לפונקציה שעושה להן filer tokenize וכדומה
5. Get_tokenized_filtered_and_stemmed_words – מבצע tokenize, מבצע filter לפי stop_words שהוגדרו בבנאי ומבצע stem (מעביר לצורת שורש) של כלל המילים בdocuments.
6. Merge_two_dicts – פונקציה סטטית שעושה merge בין שני מילונים
7. Save_data_to_file – שומרת את תוכן המילון dict לקובץ בפורמט json
8. Load_data_from_file – מוציאה את התוכן מתוך קובץ json אל מילון
9. Get_bm25_idf – מחשבת ערך idf של bm25.

הפונקציה query():

יוצרת את המילון הנדרש עליו עליו הרחבנו בחלק הקודם.

קוראת לפונקציה ret_info(dict) אותה אנחנו מייבאים מהקובץ "informationRetreivalGivenQuery", שם מתבצע כל החלק השני במטלה של אחזור המידע.

קובץ "informationRetreivalGivenQuery"הפונקציה ret_info(dict):

מטרתה להחזיר את כל מזהי המסמכים הרלוונטים ביותר לשאילתא הנתונה, כשהם מדורגים לפי ציון הדירוג. את התוצאות היא תשמור בקובץ "ranked_query_docs.txt".

נעזרת במילון שיצרנו בחלק הראשון של המטלה.

עבור השאילתא הנתונה, תחילה נריץ את הפונ' simplify_q_input().

הפונקציה simplify_q_input(): תחילה מקבלת את השאילתא משורת הפקודה.

מטרתה לבצע tokenization, הסרת מילים שכיחות (stopwords) וביצוע stemming למילים שאינן stopwords, והחזרת השאילתא לאחר פעולות אלו.

לאחר שהפונ' הנ"ל תסיים לרוץ, נבדוק אם היא מחזירה ערך False ואם כן נחזיר למשתמש שיש בעיה בשאילתא.

אחרת, נבדוק האם ranking שקיבלנו משורת הפקודה הינו tfidf או bm25.

אם "tfidf" = ranking נריץ את הפונקציה הבאה-

$$tf_idf(q, inverted_index, normal_IDF, all_docs_len) - tf_idf(q, inverted_index, normal_IDF, all_docs_len)$$

tf-idf כפי שלמדנו בכיתה.

תחילה נעבור על כל המילים ב inverted_index שלנו ונשמור במילון את הערכים כך שה key הוא מילה, וה value הוא מילון בעצמו כאשר key שלו הוא המסמך וה value יהיה ציון ה-tf idf (שזה מכפלת ה-TF וה-IDF שכבר קיימים לנו במילון).

לאחר מכן נבדוק מה המילה שמופיעה הכי הרבה פעמים בשאילתא בשביל חישוב ה-Wq.

כשיש בידנו את המידע הזה, נעבור על כל מילה בשאילתא ונבדוק כמה פעמים היא מופיעה, ונחשב את ה-W שלה ע"י מכפלת ה-IDF עם ה-TF (שזה מס' הפעמים שהיא הופיעה לחלק למס' של המילה שמופיעה הכי הרבה פעמים בשאילתא). נשמור את הערכים הללו במילון בשם Wq ששם key הוא מילה וה value זה החישוב שעשינו.

לאחר מכן, נחשב את cosine similarity של כל מסמך לשאילתא, באמצעות הנוסחה שלמדנו בכיתה. צריך לעבור על כל המילים ולחשב עבורם את ה-W_{ij}, ואז לעבור על כל המילים בשאילתא ולחשב עבורם את ה-W_{iq}. לאחר מכן להמשיך לפי הנוסחה ולשמור את כל הערכים במילון כאשר key הוא המסמך וה value זה החישוב שעשינו.

נחזיר את כל המסמכים בסדר הפוך (לפי ציון הדירוג שלהם).

נקבע גם סף threshold שנחזיר מסמכים רלוונטים עם ציון שיותר גבוה ממנו.

אם "bm25" = ranking נריץ את הפונקציה הבאה-

$q_bm25(q, inverted_index, all_docs_len, bm25_IDF, avgdl)$ – מטרתה לחשב את ציון bm25 כפי שלמדנו בכיתה.

נעבור על כל המסמכים שקיימים, ועבור כל מסמך נעבור על כל המילים בשאלתא ונבצע את החישוב שלמדנו בכיתה.

בסופו של דבר נשמור את הערכים במילון כאשר key הוא המסמך value הוא הציון.

נחזיר את כל המסמכים בסדר הפוך (לפי דירוג הציון שלהם).

נקבע גם סף threshold שנחזיר מסמכים רלוונטים עם ציון שיותר גבוה ממנו.

בסופו של דבר, נכתוב לקובץ בשם "ranked_query_docs_txt" את כל המסמכים הרלוונטים שציונם יותר גבוה מהthreshold שהגדרנו על ידי מעבר על המסמכים שהחזרנו, שהם כאמור מסודרים בציון יורד אז אכן המסמך הראשון שנחזיר יהיה בעל הציון הגבוה ביותר.

קובץ "ranked_query_docs_txt" – הקובץ שאליו נכתוב את כל המסמכים הרלוונטים שאנחנו מחזירים עבור השאלתא. מסודר מהציון הגבוה ביותר לנמוך ביותר שעובר את סף threshold שהגדרנו.