בחלק זה הוספנו חלק נוסף למערכת, על מנת לאפשר תמיכה באחזור מסמכים לפי השאילתות הנתונות. הוספנו את תקיית IR אשר מהווה את לוגיקת החישוב והחיפוש של השאילתות. בנוסף לכך הוספנו מודלים אשר משתמשים במחלקות הנ"ל.

להלן מחלקות מתקיית IR:

ממשק IRANKER

ממשק הנותן פונקציונליות דירוג למחלקה הממשת אותו.

List<String> returnRankedDocs(HashMap<AbstractTermDocumentInfo, SegmentFile> termDocuments)

הפונקציה מקבלת מפה של מילה וכל המסמכים אשר כוללים את המילה ומחזירה רשימה של מסמכים המדורגים לפי חישוב כלשהו.

ממשק ISearcher

ממשק הנותן פונקציונליות חיפוש למחלקה הממשת אותו.

HashMap<AbstractTermDocumentInfo, SegmentFile> search(HashMap<String, AbstractTermDocumentInfo> parsedQuery, List<Info> docs)

פונקציה המחזירה מפה של שורות מקבצי הפוסטינג כך שהTerm הוא המפתח.המילה הרלוונטית מופיעה גם במילון וגם במפה הנתונה. המפה בקלט מייצגת את מילות השאילתא לאחר פארסינג וtext operations.

מחלקת BM25Ranker

מחלקה זו ממשת את הממשק IRanker ומדרגת את המסמכים לפי נוסחאת bm25.

מחלקת Simple Searcher

מחלקה זו ממשת את הממשק ISearcher ומחזירה את כל המסמכים הרלוונטים השאילתא ללא דירוגם.

מחלקות הנוספו לתקיית הModel:

ממשק IInverted Index Model

ממשק הנותן פונקציונליות שליטה על הקבצים ההופכיים אשר נוצרים מתהליך הפרסור וה Text Operation

**void** StartInvertedIndex();

מאתחל את תהליך האינדוקס של הקבצים.

**void** ClearInvertedIndex();

מוחק את כל הקבצים

**void** setCorpusLocation(String location);

מגדיר את נתיב הקורפוס

**void** setStopWordsLocationLocation(String location);

מגדיר את נתיב הSTOP WORDS

**void** setPostLocation(String location);

מגדיר את נתיב קבצי הפוסטינג

**void** LoadDictionary();

טוען את המילון מהדיסק לזכרון התוכנית

List<String> getLanguage();

מחזיר את רשימת השפות בקורפוס

HashMap<String,String> getTermTF();

מחזיר עבור כל מילה את מספר המופעים שלה בקורפוס

**void** setStemmer(**boolean** selected);

מגדיר את הקישור לפנוקציית STEMMER בהתאם לבחירת המשתמש(המשתנה הבוליאני מייצג את סימון המשבצת בGUI)

**long** getTimeToFinish();

מחזיר את זמן הביצוע של עבודה כל שהיא במחלקה

**int** getDocumentDictionaryLength();

מחזיר את אורך מילון המסמכים

**void** SearchQueries(File f);

יוצר תהליך אחזור עבור שאילתות חיפוש הנמצאות בקובץ f

**void** SearchQuery(String text);

יוצר תהליך אחזור עבור שאילתת חיפוש הנתונה בtext

מחלקת Simple Invertd Index Model

מחלקה זו מקבלת נתונים מהVIEW MODEL ויוצרת את המאסטר(המחלקה אשר אחראית על תהלכי הפרסור או האחזור) המתאים. המחלקה מרחיבה את מחלקת observable לצורך התקשורת עם מחלקת VIEW MODEL וממשת את ממשק IInvertedIndexModel לצורך פונקציונליות שליטה בקבציה הInverted Index

מחלקת data muse object

המחלקה מייצגת מילה בAPI. מחזירה מילה אחת בכל פעם.

מחלקת Semantic Data Muse

המחלקה נגשת אל הAPI http://api.datamuse.com/words

מחלקת Query

המחלקה יורשת מAbstract Document ובנוסף שומרת את נתוני השאילתה הנלווים- narrative ו- description.

מחלקת Abstract Document

המחלקה הינה מחלקה מופשטת אשר מייצגת מידע עם נתיב, שם קובץ מספר מזהה וטקסט.

מחלקת Abstract Tokeinized Document

מחלקה מופשטת אשר מייצגת מידע של טקסט כלשהו לאחר פעולת הText Oparation

מחלקת Tokeniezed Query

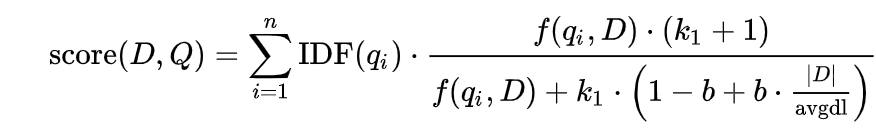
מחלקה אשר יורשת ממחלקת Abstract Tokeinized Document מייצגת את הטקסט של השאילתא לאחר תהליך הtext operation

מחלקת XML Reader Factory

מחלקה אשר מקבלת קובץ וסוג הקובץ ושולחת אותו לכתיבה המתאימה

על מנת לאחזר את המסמכים הרלוונטים, עלינו לבצע פרסור וText Operation. לכן אנו משתשמשים בכל המחלקות בהם השתמשנו עד לשלב הכתיבה של הקבצים הזמניים. שיטת הפרסור הינה זהה לחלוטין לחלק הקודם, בין כל שלב בחלק זה המידע עובר בתור מקבילי אשר מוסר מידע מעובד על השאילתות בין המחלקות השונות. השוני היחיד הינו במחלקה הראשונה, מאחר ואנו לא קוראים קבצים מוגדרים כמו הקורפוס אנו עוברים ישר לשלב השני שהוא הtokenize. בנוסף, אנו נעזרים במחלקת data provider לקבל מידע שברובו סטטי לכל המחלקות, לדוגמא: נתיב קבצי הפוסטינג, מילוני קבצי הפוסטינג השונים וכו'.

אלגוריתמים הכלולים במנוע:

אלגוריתם הדירוג- אלגוריתם דירוג של מסמכים עבור שאילתא נתונה. אלגוריתם זה בוחן את כל המילים הנמצאות בשאליתא ובמסמך ומנרמל את תוצאותיו לצורך השוואת איכות המסמכים.

 - תדירות המילה q במסמך D

* - גודל המסמך

avgdl – אורך ממוצע של מסמך בקורפוס

K ו- b הינם קבועים אשר עוזרים לנו לתת דירוג יותר איכותי (קרוב לתוצאות אותם היינו מצפים לקבל).

בנוסף לכך, משקלנו את המילים בצורה הבאה: מילים אשר הופיעו בשאילתה המקורי, הכפלנו את ערך הBM25 שלה ב1 אחרת הכפלנו את ערך הBM25 ב0.49

אלגוריתם למציאת 5 הישויות הדומיננטיות- במהלך האינדוקס הוספנו שדה במחלקת DocumentTermInfo אשר משרשר את 5 הישויות הדומיננטיות במסמך. את הישיות האלו מצאנו לפי כמות חזרות המילה במסמך. לאחר סיום הפרסינג מעוברת למחלקת SegmentFiles מפה של כל מילה במסמך ומידע עליה, ממידע זה מצאנו את 5 המילים (עם upper case) עם השכיחות הגבוהה ביותר.

אלגוריתם לשיפור סמנטי- על מנת לאחזר מסמכים היותר מתאימים לשאילתא הנתונה, השתמשנו באתר datamuse.com. נגשנו לאתר זה מספר פעם עם בקשות לקבוצות מילים שונות והן:

-מילים עם קשר סמנטי

-מילים בעלות אסוציאציה זהה

-מילים המתארות בערך את המילה

- מילים אשר הינן הכללה של המילה הנתונה

בנוסף לכך, שלחנו את הTOPIC שהוא הנושא ממנו יש לשלוף את המילים הסמנטיות. את הTOPIC יצרנו לפי חיתוך של המילים ב narrative ו- description של השאילתא. זהו גם הינו הקוד פתוח היחיד בו השתמשנו בחלק זה.

השתמשנו בנתוני הפוסטינג השונים:

מקבצי הפוסטינג של הTERM שלפנו את המסמכים אשר הTERM הופיע בהם עבור הSEARCHER. כמו כן, שלפנו את התדירות של המילה בכל מסמך עבור הBM25 .

בנוסף לכך, מתוך קובץ הפוסטינג של המסמכים, שלפנו את חמשת הישויות הדומיננטיות על מנת להדפיסם בVIEW .