דו"ח פרויקט בניית מנוע לאחזור מסמכים – חלק א'

1. עיצוב התוכנה:

פירוט מחלקות:

**מחלקת ReadFile:**

מחלקה האחראית על קריאת הקבצים והמסמכים שבתוכם.

שדות המחלקה:

שיטות המחלקה:

* readFile מקבלת נתיב לקובץ ספציפי במאגר ומחזיר LinkedHashMap כאשר המפתח הוא אובייקט מסוג Document והערך הוא String המכיל את תוכן המסמך הספציפי. השיטה עוברת על הקובץ באמצעות BufferedReader ומכניסה מסמך לתוך StringBuilder.

readFile קוראת לשיטה שנקראת readDoc אשר מעדכנת את ה-LinkedHashMap ובעצם מפרידה בין המסמכים בקובץ.

* readDoc מקבלת String המכיל את טקסט המסמך, int שמייצג את מיקום המסמך בקובץ, String המכיל את נתיב הקובץ בו נמצא המסמך ואת הLinkedHashMap אשר אותו יש לעדכן.

השיטה בוררת מתוך הטקסט של המסמך את הDOCNO מספר המסמך ואת התוכן עצמו של המסמך בעזרת Matcher המזהה אותם בטקסט. מהארגומנטים שקיבלה ומהDOCNO היא יוצרת אובייקט Document המכיל את כל הנתונים על המסמך ומצרפת לLinkedHashMap את הDocument כמפתח ואת התוכן של המסמך כערך. בנוסף השיטה מכניסה את הDocument שיצרה לתוך ה-List של כל המסמכים במאגר.

**מחלקת Parse:**

מחלקה המקבלת HashMap המכיל אובייקט Document במפתח וString כערך. כאשר האובייקט Document מכיל מידע על המסמך (יפורט בהמשך) והString מכיל את תוכן המסמך (כל מה שבין תגיות הTEXT).

המחלקה עוברת על תוכן המסמך, מפרסרת אותו ולבסוף יוצרת מבנה נתונים HashMap המכיל String כמפתח שהוא בעצם הterm לאחר פרסור ו-MetaData שזה אובייקט שמכיל את כל המידע הנחוץ על ה-Term. (יפורט בהמשך על אובייקט זה).

שדות המחלקה:

* Map של months המכיל String כמפתח ויכיל את חודשי השנה ואת הקיצורים שלהם (כלומר גם febuary וגם feb למשל) אל מול הערך המספרי שלהם (2 במקרה של פבואר) שיהיה גם שמור כString.
* HashSet של stopWords המכיל את כל ה-Stop Words.
* HashSet של Characters המכיל characters מסוימים: נקודה, פסיק, כל סוגי הסוגריים, גרשיים, נקודותיים ונקודה פסיק.
* שדות שונים של Patterns שיעזרו לנו לזהות תבניות במילים. שדות התבניות הם: תבנית לזיהוי מילה, תבנית לזיהוי מספר, תבנית לזיהוי מילה עם אות גדולה בתחילתה או שכולה באותיות גדולות, תבנית לזיהוי מילים עם מקפים ותבנית לזיהוי מספרים עם מקפים.

שיטות המחלקה:

* Parse: מקבלת HashMap של מסמכים Documents מול התוכן שלהם כString ומתחילה לעבור על הString עם שני מצביעים: pos אשר מצביע לאינדקס הראשון של המילה וindex אשר מצביע על האינדקס האחרון של המילה. הפונקציה רצה על תוכן המסמך עם שני הפוינטרים ולוקחת כל פעם מושג מהמסמך באמצעותם לתוך String שנקרא potentialTerm. על המושג הפוטנציאלי הפונקציה בודקת מקרים, שולחת לפונקציות הרצויות (שיפורטו למטה) על מנת לתקן פורמטים של מושגים. במידת הצורך ישנם עוד שני פוינטרים nextPos ו-nextIndex שעוברים קצת הלאה מהמילה ובודקים אם יש צורך לשרשר את המילים הבאות למושג שעליו אנחנו עובדים כרגע (כמו במקרה של תאריך שמכיל יום, חודש ושנה – נשרשרם ביחד לpotentialTerm אחד) השימוש הוא בעיקר בפונקציות של String: indexOf ו-substring. שנחשבות מהירות יחסית. כל מושג שפורסר נכנס למאגר הנתונים HashMap<String, MetaData> parsedterms שאותו בסופו של דבר הפונקציה מחזירה.
* cleanTerm: מקבלת String שיש בו נקודות, סוגריים, ובעצם מילה או מספר שמכיל סימנים לא רצויים ומנקה אותו מהם. מחזירה את הString נקי.
* numbers: מקבלת String שהוא מספר ומשנה אותו לפורמט הנכון, כלומר מספר עשרוני ישתנה לעד 2 אחרי הנקודה ופסיקים ירדו. מחזירה את הString בפורמט התקין.
* Dates: מקבלת term המוצג בפורמט תאריך מסוים שלפי ההוראות לא תקין ומשנה אותו לפורמט התקין. מחזירה את הString בפורמט התקין.
* Percent: מקבלת String בפורמט לא תקין של אחוז (מספר עם הסימן % או percentage ומשנה לפורמט התקין: number percent. קוראת לפונקציית numbers כדי לתקן את המספר לפורמט התקין ואז מחזירה את הString הסופי התקין.
* \*Dollar: מקבלת String הבנוי בצורה $number ומשנה אותו להיות number dollar. הסיבה לפונקציה זו היא שאם למשל מופיע במסמכים $35 וגם 35 dollar המושג לא יישמר פעמיים.
* upperCaseWords: מקבלת String המכיל מילים רצופות של אותיות גדולות בתחילת כל מילה ומחזירה מערך של כל מילה בנפרד שאינה stop word ללא אותיות גדולות ובתא האחרון במערך יש את הביטוי בשלמותו ללא אותיות גדולות.
* \*wordsWithHyphen: מקבלת String המכיל מילים עם מקפים ביניהם ומחזירה מערך של כל מילה בנפרד שאינה stop word.
* \*dotsBetweenWords: מקבלת ביטוי המכיל נקודות כמו למשל U.S.A ומשנה אותו להיות ללא נקודות )מהדוגמה: USA) ומחזירה אותו
* \*removeS: מקבלת מילה עם שייכות כמו Avigail’s ומוחקת את ה’s ממנה. מחזירה את הביטוי ללא השייכות (במקרה של הדוגמה מחזירה Avigail) הסיבה לכך היא שלא תשמרנה שתי מילים זהות רק בגלל שאחת מהן עם ‘s.
* setMonthMap פונקציית עזר שנועדה להכניס את כל החודשים למבנה הנתונים של החודשים.
* updatePotentialTerm: מקבלת Document (את זה שעליו עוברים כרגע בparse), מקבלת את String ה-term שאותו סיימנו לפרסר ואת הHashMap שאליו מכניסים את כל הterms המפורסרים. מעדכנת את כל התדירויות והמידע הנחוץ. במידה והterm לא קיים כלל במבנה הנתונים, ניצור MetaData חדש ונעדכן את כל התדירויות ל-1. נכניס את הDocument הנוכחי לMetaData (עם תדירות של 1 אם המסמך פעם ראשונה נכנס לMetaData של המושג, או שנעלה את הtf ב-1 אם המסמך כבר היה בMetaData של המושג.)
* getPosToAfterWhiteSpaces – פונקציית עזר שמכוונת לי את אינדקס ההתחלה pos או nextPos אל המילה הבאה שיש לפרסר. כלומר מדלג על whitespaces שבין מושג למושג. מחזיר את האינדקס הראשון של המילה הבאה. במידה ונגמר הקובץ מחזיר -1.

\* הפונקציות המסומנות בכוכבית הן חוקים שאנחנו הגדרנו לפרסר מעבר לחוקים שהתבקשו בעבודה.

**מחלקת Document:**

מחלקת עזר המכילה נתונים על מסמך ספציפי.

שדות המחלקה:

docNo – שדה String המכיל את המספר המזהה של המסמך.

Length- אורך המסמך (כמות מילים)

maxTf – int של התדירות של המילה הנפוצה ביותר במסמך.

שלושה פרטים נוספים שהחלטנו לשמור על מסמך:

commonTerm – String של המילה הנפוצה ביותר במסמך.

Path – שדה String המכיל את נתיב הקובץ שבו המסמך נמצא.

positionInFile – פוינטר למיקום המסמך בקובץ.

שיטות המחלקה הם getters ו-setters בלבד.

**מחלקת MetaData:**

מחלקת עזר אשר מכילה נתונים על term ספציפי.

שדות המחלקה:

Df – מכיל את כמות המסמכים בהם מופיע ה-term.

frequencyInCorpus – מכיל את כמות המופעים של הterm במאגר כולו.

frequencyInDoc – HashMap שמכיל כמפתח מסמך בו המילה הופיעה וכערך את הtf של אותו מסמך – כמה פעמים הופיעה במסמך.

שיטות המחלקה הם getters ו- setters.

**מחלקת Term:**

מחלקה שנועדה לשמור פרטים על המושג- פרטים שצריכים להיות במילון. מחלקה זו נועדה לשמור אובייקטים מסוג Term במילון.

שדות המחלקה:

String term – המושג עצמו.

Int df – כמות המסמכים שהמושג מופיע בהם.

Int frequencyInCorpus – כמות המופעים של המושג בכל המאגר.

Long pointerToPosting – מיקום המושג בקובץ הפוסטינג הסופי.

String postingFilePath – הנתיב של קובץ הפוסטינג הסופי.

שיטות המחלקה עם getters ו-setters ובנוסף דרסנו את equals שישווה שני Strings של term ודרסנו את hashCode שיהיה hashCode של string על term. המחלקה גם עושה implement לcomparable ודורסת את השיטה compareTo ככה שתשווה frequencyInCorpus של שני Terms. זאת על מנת שנוכל למיין אחר כך את המושגים לפי תדירויותיהם ונוכל לדעת מה להכניס לcache. שאר הדריסות נועדו להכניס את המושגים לHashMap של המילון בצורה יעילה.

**מחלקת Stemmer:**

מחלקת open source שנלקחה מהאתר: <https://tartarus.org/martin/PorterStemmer/java.txt>

לכל מילה שנרצה לעשות לה stemm ניצור לה Stemmer חדש בparse נשתמש בפונקצייה add כדי להוסיף אותה ל-Stemmer . נשתמש בפונקציית stemm() של המחלקה שעושה Stemm למילה שעשינו לה add. ואז ניקח את המילה מהStemmer באמצעות הפונקציה toString של המחלקה.

**מחלקת Indexer:**

מחלקת הIndexer אחראית לבניית המילון, הcache וקבצי הposting . המחלקה מפעילה את הReadFile על הcorpus, לאחר מכן מפעילה את הParse ושומרת את כל המושגים שפורסרו בתוך HashMap. המחלקה בונה קבצי posting ראשוניים (זמניים) קבוצת מושגים שנמצאים בHashMap. לאחר שקבצי הposting הזמניים מוכנים, נאחד אותם ע"י מיון וכתיבה ל-6 קבצי posting סופיים שמחולקים לפי אותיות או מספרים.

שדות המחלקה:

String corpusPath - הנתיב לקובץ שבו נמצא מאגר הנתונים שממנו נקרא את הקבצים. מקבלים את הנתיב דרך הGUI.

filePath - הנתיב שבו הקבצים הסופיים יימצאו - המילון וקבצי הposting.

tempDictionary - שדה עזר לבניית המילון.

Dictionary - HashSet המכיל אובייקטים מהסוג term.

int counterForFiles - counter שרץ יחד עם הלולאה שעוברת על כל הקבצים בcorpus ולפיו נותנים שם לקובץ posting הזמני.

שיטות המחלקה:

initialize - נכנסת לתיקיית המאגר, ושולחת אותה ל- ReadFile. עבור כל קובץ שנקרא נקבל מבנה נתונים של מסמך מול התוכן שלו - של כל ההמסמכים באותו הקובץ. נשלח לפונקצית parse שמקבלת את המסמכים ועוברת על התוכן ומפרסרת. בסוף הפונקציה תחזיר לנו מאגר נתונים של כל המושגים מהקובץ שפורסרו. את מבנה הנתונים של כל המושגים נשלח לפונקציה constructPosting אשר יוצרת קבצי פוסטינג זמניים לכל קובץ במאגר ולאחר שנסיים לרוץ על כל הקבצים במאגר וליצור מכל אחד קובץ פוסינג, נקרא לפונקציה mergePosting אשר יוצרת קבצי פוסטינג סופיים. כעת נקרא לפונקציה createCache אשר יוצרת את מבנה הנתונים של הcache. פונקצית initialize היא הפונקציה הראשית של מחלקה זו והיא זאת שנקרא לה בGUI לבניית כל האינדקס.

constructPosting - פונקציה זו עוברת על כל המושגים במאגר שקיבלה- מכניסה למילון במידה והמושג לא במילון, או מעדכנת את התדירויות של המושג במילון במידה והמושג כן במילון. המילון בנוי כHashMap שהמפתח הוא המושג כString כדי שנוכל בo(1) להגיע אליו במילון, והערך הוא אובייקט Term של אותו מושג, כדי שנוכל לעדכן בקלות את התדירויות שלו ושאר המידע שאנחנו שומרים על המושג. הפונקציה יוצרת ArrayList לצורך שמירת כל פרטי המושגים שנרצה לשמור בקובץ פוסטינג הזמני. לשם נכניס כל String שניצור מהמושג ונרצה שיהיה בקובץ פוסטינג. את הקובץ פוסטינג ניצור ונפתח עם bufferedWriter. את המידע על המושג נשמור בString מתוך האובייקט MetaData בפורמט הבא: "term:d1:23,d2:34," כאשר di הוא המסמך והמספר שלאחריו הוא ה-tf. לאחר שנשמור בArrayList את כל השורות לפי כל המושגים בקובץ שעליו אנחנו עובדים כרגע, נמיין את הרשימה ונכתוב אותה לקובץ שפתחנו שורה אחר שורה כך שכל המושגים ממויינים לפי סדר. במעבר על הקבצים יש פוינטר בשדה המחלקה שנקרא counterForFiles אשר מתחיל מ-0 ועולה ב-1 כל פעם שיוצר פוסטינג ונותן לו את השם של הcounterforFile. כלומר שמות קבצי הפוסטינג הזמניים ממוספרים מ-0 עד מספר כל הקבצים במאגר.

mergePosting - נרוץ על קבצי הפוסטינג שיצרנו באמצעות הcounterForFiles בלולאה ונאחד כל שני קבצים לקובץ אחד ממויין עד שcounterForFiles יהיה גדול מ-2 (עד שיישארו לנו שני קבצים). נפתח שני bufferedReader לכל שני קבצי פוסטינג זמניים וBufferedWriter אחד שיוצר קובץ חדש עם שם זמני. נאחד באמצעות פונקציית helpMerge את שני הקבצים לקובץ החדש ונמחק את שני הקבצים שאיחדנו. לאחר מכן נשנה את שם הקובץ החדש להיות מספר של counter שמאותחל ל-0 וגם כן רץ עד כמות הקבצים בדיוק באותה דרך שעשתה הפונקציה הקודמת. לאחר כל ריצה אנחנו מסיימים עם חצי מכמות הקבצים שהיו לנו ולכן נחלק את counterForFiles ב-2 ונאתחל את counter ל-0. במידה ולאחר איטרציה יש לנו מספר אי זוגי של קבצים - קודם נאחד את הקובץ עם המספר הכי גבוה עם הקובץ עם מספר שקטן ממנו ב-1 וניתן לקובץ המאוחד את המספר של הקובץ ששמו הוא המספר הקטן יותר מביניהם, זאת כדי לשמור על רצף המספור של הקבצים. תמיד תהליך האיחוד יתבצע רק כאשר יש מספר זוגי של קבצים בגלל שאנחנו כל פעם מאחדים שני קבצים אחד עם השני.

helpMerge - מקבלת שני bufferedReader וBufferedWriter אחד ומאחדת בין שני הקבצים לתוך הקובץ השלישי. הפונקציה מאחדת באמצעות פונקציית mergeSort טיפוסיץ. קורא שורה אחת מקובץ אחד ושורה אחת מקובץ שני ומשווה ביניהם (ניתן לראות שלא עברנו את מגבלת שלושת השורות) לפי ההשוואה היא יודעת איזו שורה לכתוב לקובץ השלישי ומהקובץ שממנו קראה - היא תקרא שוב עוד שורה וכך עד ששני הקבצים יתאחדו לקובץ אחד. אין יותר משני שורות (אחת לכל קובץ) שנקראות בו זמנית.

FinalPosting - לאחר שנשארנו עם שני קבצי פוסטינג זמניים נכתוב אותם ל-6 קבצי פוסטינג סופיים בצורה הבאה: קובץ ראשון רק מספרים מתוך שני הקבצים בצורה ממויינת. באותה צורה נכניס לעוד 5 קבצים לפי האות הראשונה של כל מילה: A-E, F-J, K-O, P-T, U-Z. לכל קובץ פוסטינג יש int שהוא פוינטר בתוך הקובץ. בכל פעם שנשמור שורה לקובץ, נעדכן במילון במושג השייך לשורה הזאת, את הפוינטר של אותו הקובץ ואת הנתיב לקובץ הפוסטינג הסופי , ואז נעלה את הפוינטר של אותו קובץ פוסטינג בכמות הcharacters ששמרנו לקובץ.

createCache - ה-cache הוא שדה במחלקה והוא מבנה נתונים HashMap<String,String> כדי שנוכל לחפש בו מושג בצורה מהירה. המפתח הוא מושג והערך הוא שלושה מסמכים ראשונים של אותו מושג עם הTF שלהם. אנו רוצים שהמושגים שנכניס למטמון יהיו הרלוונטיים ביותר, לכן ניקח את כל הערכים במילון (כל אובייקטי הTerm) ונמיין לי התדירות של המילה במאגר. לאחר שנמיין ניקח מהמילון את 10,000 המילים הראשונות אל תוך המטמון.

resetIndex - מקבל את כל הנתיבים של כל התיקיות והקבצים שיצרנו ומוחקת אותם. פונקציה זאת תקרא לאחר לחיצה על כפתור הreset בGUI.

**מחלקת GUI:**

המחלקה שדרכה מריצים את התוכנית. המחלקה כתובה בJavaJX ומכילה את פונקציית הMain שמפעילה פונקציית launch שקוראת לפונקציית Start. בפונקציית start יוצרים את כל הרכיבים שבGUI – כפתורים, gridPane, box למיניהם, תיבות טקסט ועוד. מחלקה זו נעזרת במחלקת Controller כדי לתקשר עם שאר המחלקות של הפרויקט.

שיטות המחלקה:

Start: הפונקצייה הראשונה שרצה, בונה את הממשק ואת הרכיבים בו ויוצרת לכל כפתור או רכיב event לפי הצורך. ה-eventים- במידה ומדובר בפונקציונליות של המנוע, תקרא לפונקציה בController שאחראי להפעיל את פונקציות שאר המחלקות. במידה ומדובר בעדכון דברים בGUI הפונקציונליות תמומש בתוך הevent ישירות. כמו למשל כאשר קורה event על תיבת טקסט – התוכן של התיבה נשמר בשדה String במחלקת הGUI.

**מחלקת Controller:**

מחלקה שמהווה את הפונקציונליות של הGUI ואת הקישור שלה לשאר המחלקות. במחלקה שדה בוליאני שנקרא toStemm שלוקח מידע מהGUI לגבי האם המשתמש רוצה להפעיל את התוכנית עם Stemmer או בלי.

במחלקה זו 5 שיטות:

setToStemm – שיטה זו נקראת מהGUI כאשר קוראת פעולה על ה-checkBox בGUI שנקרא enableStemming במידה והcheckBox מסומן נקבל בשיטה true, אחרת נקבל false.

isToStemm – מחזיר true אם השדה toStemm הוא true.