אחזור מידע

דו"ח מנוע

מגישים:

רועי דור – 301730354

יונתן זקס – 204662779

מגישים: רועי דור 301730354, יונתן זקס - 204662779

דו"ח - מנוע חיפוש חלק ב'

מלבד המחלקות אותן התבקשנו לממש בחרנו להוסיף את המחלקות הבאות המחולקות לחבילות. חלק מהקבצים מכילים בתוכם מספר מחלקות, עליהן נפרט:

- Indexing:
  - CountriesAPI
  - Document:
    - Document
    - TermData
  - FileWriter
  - KWayMerge:
    - MyHeap
    - MergeDataClass
    - Merger
  - MyDictionary:
    - MyDictionary
    - DictionaryData
    - DocumentIndexData
    - CityIndexData
- Parsing:
  - IterativeParsing
- Ranker
- Searching
  - Searcher
  - MyWordEmbedder

ואת המחלקות הבאות מחוץ לחבילות:

- Configurations
- Manager

חלק מקבצי ה-Python שלנו אינם מכילים מחלקות, אלא מטרתם לשמש כפונקציות עזר לכלל חלקי המנוע:

- ConvertMethods
- BasicMethods
- PreRun

מגישים: רועי דור 301730354, יונתן זקס - 204662779

שינויים בחלק א':

Main:

ביצענו שינוי במחלקת Main כדי שתתאים לאסטרטגיה חדשה שנקטנו עבור חלק הפרסור והאינדוקס. שינוי האסטרטגיה נבע כדי לאפשר דחיסה שמות המסמכים הנשמרים בקבצי הפוסטינג. תיאור אופן הדחיסה מתואר תחת אלגוריתמי המנוע, אלגוריתם דחיסה. הדרך שבה אפשרנו דחיסה זו הוא ע"י חלוקת מספר לכל מסמך, ע"י ריצה מהירה לפני תחילת הרצת הבניה – PreRun. כך בעת בניית קבצי הפוסטינג אנו מבצעים דחיסה ע"י gaps.
בנוסף הוספנו חלק שעוזר לנו ליצור קובץ 5 ישויות דומיננטיות עבור על מסמך למהלך הריצה של בניית האינדקס.
בחלק זה אחרי בדיקות רבות החלטנו להגדיל את הסף לסינון מילים הקיימות בכל הקורפוס פחות מ-4 פעמים, משום שמצאנו כי מילים אלה הינם מקריות ולרוב חסרות משמעות.

DocumentIndexData Class:

הוספנו רשימה למחלקה שמחזיק את חמשת הישויות הדומיננטיות במסמך ע"י החישוב שמתבצע בשלב האינדוקס. בהתאם נוספנו פונקציה שמחזירה את הישויות בפורמט מוכן לכתיבה לאינדקס מסמכים-ישויות.

TermData Class:

הוספנו עבור חלק א' פרסור והתייחסות למילים שנמצאות תחת תגית <TI>, שמשמעותה הוא כותרת המסמך. אנו משתמשים במידע זה כדי לשפר את האחזור. לכן הוספנו משתנה ופונקציה שיעזרו לנו לשמור מידע על כל term בכל מסמך המעיד האם המילה הופיעה בכותרת של המסמך. במידה ומילה הופיעה בכותרת המסמך נוסיף למיקומים שלה, שאנחנו שומרים בפוסטינג עבור כל מילה בכל מסמך, בתחילת המיקומים '-'.

Indexer Class:

הוספנו לפונקציה העיקרית של המחלקה, addNewDoc, אשר מקבלת מידע של מסמך שכרגע עבר פרסור ומאנדקסת את כל המילים שהיא מכילה במילונים לפני שמירה לדיסק. הוספנו לפונקציה זו במהלך המעבר על כל ה-terms שבמסמך שבודקת האם ה-term הוא ישויות במסמך (כולו באותיות גדולות) ושומר את כל הישויות הדומיננטיות ביותר. פירוט על פונקציית הדירוג של ישויות נמצא תחת אלגוריתמי המנוע, אלגוריתם דירוג ישויות.

Parse Class:

ע"מ לתמוך בדחיסה ובהוספת מילים הנמצאות בכותרת, שיפרנו את המחלקה כדי שתעבור עם ה-PreRun כדי לקבל מספר עבור כל מסמך.

Configuration Class:

המחלקה נוספו השדות:

- BM25_K – ערך k עבור חישוב BM25 בRanker.
- BM25_B – ערך b עבור חישוב BM25 בRanker.
- Axu_Value– ערך קבוע הנכפל בציון שחוזר מ Axiomatic
- BM25_avgDocLength – אורך מסמך ממוצע בשביל חישוב BM25.
- totalNumberOfDocs– כמות מסמכים בשביל חישוב BM25.
- totalNumberOfTerms– כמו הביטויים בשביל חישוב BM25.

שיפור בהתמודדות עם מגבלות הזיכרון:

באמצעות שימוש במספרים שהקצענו עבור כל מסמך, ושמירת המספרים, תוך שימוש באלגוריתם דחיסה לפי gap הצלחנו להקטין משמעותית את גודל קבצי הפוסטינג שלנו.

כעת אנו שומרים מידע שלא שמרנו בחלק א':

- אינדוקס הכותרת של כל מסמך וסימון המעיד כי המילה מופיע בכותרת המסמך.
- עבור כל מסמך אנו שומרים את חמשת הישויות הדומיננטיות שלו.

ובכל זאת הצלחנו להקטין את קבצי הפוסטינג שלנו ביחס למה שהגשנו בחלק א' בכמעט **45%.**

כך כיווצנו את קבצי הפוסטינג מ-1.3GB עבור קבצי פוסטינג ללא Stemming לכ-**720MB**.

כפי שניתן לחשב, גודל הפוסטינג שלנו כרגע מהווה כ-**55%** ביחס לקבצי הפוסטינג הקודמים.

יישמנו דחיסה זו לאורך כל קבצי הפוסטינג: קבצי האינדקס ההופכי עבור מילים וערים. ומשום שמספר המסמך תואם למיקום שלו בקובץ האינדקס עבור מסמכים לא נאלצנו לשמור את ערך האינדקס שהקצענו למסמך בצורה פרטנית, גם בקובץ המסמכים וגם בקובץ הישויות הדומיננטיות.

למעשה עבור קורפוס כפי שקיבלנו, בו שם מסמך נע בין 7-12 ספרות, וקיימים בו פחות ממיליון מסמכים, כלומר מספר החסום ב-6 ספרות. כך אנו מבטיחים לחסוך באמצעות הדחיסה זיכרון בוודאות עבור כל הופעות מילה במסמך, במיוחד עבור מילים בעלות שכיחות גוהה.

מגישים: רועי דור 301730354, יונתן זקס - 204662779

מחלקות של חלק ב':

Searcher Class:

מחלקה שתפקידה לקבל שאילתה ולהחזיר רשימת מסמכים ממוינים ע"פ דירוג המסכמים.

המחלקה מכילה את השדות:

- config – מייצג את ההגדרות של ריצת התוכנה הנוכחית.
- wordEmbedding – משתנה מסוג MyWordEmbedder הטוען את המידע שיצרנו עבור ה-embedding ומאפשר לנו להכניס מילים או רשימות של מילים ולקבל מילים נרדפות. הסבר מפורט תחת המחלקה **MyWordEmbedder.**
- termDictionaryNoStem – מחזיק את המילון של האינקס ההופכי לפי term ללא Stem.
- termDictionaryWithStem – מחזיק את המילון של האינקס ההופכי לפי term עם Stem.
- iterativeTokenizer – משמש כפארסר עבור השאילתות. נעביר את השאילתה את אותו פרסור שהעברנו את המסמכים במהלך יצירת קבצי הפוסטינג.
- ranker – משתנה מסוג Ranker אשר מקבל מידע של term ומסמך ומחזיר ציון עבורם. מידע מפורט תחת Ranker Class.
- documentsByCitiesSet – רשימת ערים המייצגת משתנה של המחלקה שיחזיק את רשימת המסמכים השייכים לערים לצורכי סינון מסמכים לפי ערים.

המחלקה מכילה את הפונקציות:

- getDocsForQueryWithExpansion(queryString, citiesList, expend, useStem) – הפונקציה העיקרית של המחלקה. תפקידה לקבל שאילתה ולהחזיר רשימת מסמכים ממויינים ע"פ דירוג המתאימים לשאילתה. פונקציה זו משמשת לכל סוג של שאילתה.
  כאשר נרצה לסנן מסמכים לפני ערים נוכל לקבל רשימת ערים ולסנן לפיהם.
  כאשר נרצה לבצע שאילתה עם התייחסות סמנטית נכניס לפוקציה ב-expand, שהוא ערך בוליאני, True.
  כאשר נרצה לחפש מידע במילון עם stem באותו אופן כמו עם expand נכניס True לערך useStem.
  אופן עבודת הפונקציה:
  - הפונקציה תבצע התאמות לפי ערך ה-expand ו-useStem במידע שעליו היא עובדת.
  - במידה ויש לעשות הרחבה של השאילתה לפי סמנטיקה נקרא לפונקציה expandQuery, ונקבל רשימה של מילים נרדפות למילים בשאילתה.
  - באמצעות הפונקציה getDocumentsFromPostingFile נקבל את הנתיב של קובץ הפוסטינג הרלוונטי עבור כל term בשאילתה.
  - עבור כל מסמך שה-term מופיע בו נשתמש בפונקציה getDocumentsScoreFromPostingLine כדי להשיג מילון עם ציון עבור כל מסמך. עבור כל מילה שהופיע בעצמה בשאילתה נוסיף את הציון כמו שקיבלנו אותו מה-Ranker.

- במקרה שעשינו שימוש בסמנטיקה לשאילתה נעבור על כל המילים שחזרו מ-expandQuery . עבורם נבצע את אותו התהליך שעשינו עבור מילים שהופיעו בשאילתה, אלא שאת הדירוג שיחזור מהם ננרמל בצורה הבאה:
  כל מילה שחזרת מ-expandQuery חוזרת עם 2 מספרים נוספים כ-tuple:
  1. הדמיון של המילה למילה שממנה קיבלנו את המילה הזו לפי ה-Embedder.
  2. כמה מילים שחזרו מהEmbedder באמת נמצאות במילון שלנו. זאת משום שה- Embedder שלנו לא אומן רק על הקורפוס שלנו והמילים שלנו, לכן כדי שלא ייצא מצב שעבור 2 מילים שונות בשאילתה למילה אחת יחזרו 5 מילים שנמצאות במילון שלנו ובשנייה 2 מילים, דבר שייתן עדיפות בדירוג עבור המילה שחזרו עבורה 5 מילים.

את הציון שחוזר עבור מילים אלה נכפיל בכמה המילה דומה למילה שבגללה היא חזרה (1) וב-1 חלקי כמות המילים שחזרו עבור המילה שבגללה המילה הגיע דרך ה-Embedding (2).

- נסנן את המילים בפונקציית filterByScores ונחזיר את 50 המסמכים בעלי הדירוג הגבוה ביותר.
- expandQuery(queryList, termDictionary) – הפונקציה מקבלת רשימת מילים השייכות למילים בשאילתה ומילון מילים המתייחס למילים במצב הנוכחי של החיפוש (Stem/No Stem). הפונקציה משתמשת ב-MyWordEmbedder לצורכי תשאול על מילים נרדפות. עבור כל מילה בשאילתה נמשוך 5 מילים נרדפות ע"י getTopNSimilarWords, ובנוסף נמשוך את 10 המילים הדומות ביותר לכלל השאילתה באמצעות פונקציה getTopNSimilarWordsFromList. עבור כל רשימת מילים שחוזרות מה-Embedder נבדוק כמה מילים נמצאות אצלנו ע"י הפונקציה getExistingResults וכך לנרמל את כמות המילים הנרדפות שחוזרות בצורה שווה. הפונקציה מחזירה רשימה של מילים נרדפות למילות השאילתה בצורה הבאה: $(\text{word, similarity, } \frac{1}{len(existing\ words\ in\ dictionary\ term\ in\ query)})$, עבור כל המילים בשאילתה.
- getExistingResults(mostSimilar:list, termDictionary:dict)->list – מסנן מילים המגיעות מה-Embedder עם מילים שנמצאות המילון שלנו ומוסיף לרשימה שקיבל מהצורה: (word, similarity), לצורה בה expandQuery צריך לאחזר.
- filterByScores(doc_Score_list: list)-> list – פונקציה המקבלת רשימת tuple מהצורה: (doc,score), הנקראת בחלק האחרון של אחזור המסמכים ולאחר שחושב עבור כל מסמך הציון הסופי שלו.
  הפונקציה מסננת את הרשימה החל מהמסמך במיקום ה-20, כך שלא נחזיר מסמכים שערכם קטן מ-0.4 מערכו של המסמך עם הדירוג הגבוה ביותר. במידה ורשימת המסמכים קטנה מ-20 נחזיר את כל המסמכים ללא פילטור.
  מצאנו שכך הצלחנו לא לאבד את מספר המסמכים הרלוונטיים שאנו מאחזרים, כלומר ה-Recall לא נפגע ושיפרנו את ה-Precision ע"י אחזור פחות מסמכים לא רלוונטיים.

- getDocumentsFromPostingFile(term:str) -> str – פונקציה המקבלת term ומחזירה את הנתיב לקובץ בו המידע על ה-term נמצא. לפי צורת שמירת קבצי הפוסטינג: חלוקה ל-27 תיקיות שמסמנות תחילית של מילה ושמירת קבצי הפוסטינג לקבצים בגודל חסום של 1MB, נוכל למצוא את הקובץ המתאים במהירות ויעילות. למעשה בגישה בזמן ריצה לקובץ נצטרך לבצע חיפוש רק בין קבצי האות בה המילה מתחילה ולקרוא גודל קבוע של 1MB.
- getDocumentsScoreFromPostingLine(postingFilePath, term, line, useStem) – פונקציה האחראית כל הבאת המידה מקובץ הפוסטינג עצמו. הפונקציה מקבלת את כל המידע שהיא צריכה על השורה שצריך לאחזר: נתיב לקובץ, המילה, מספר שורה בקובץ פוסטינג והאם להשתמש בStem. לפי צורת שמירת קבצי הפוסטינג: חלוקה ל-27 תיקיות שמסמנות תחילית של מילה ושמירת קבצי הפוסטינג לקבצים בגודל חסום של 1MB, נוכל למצוא את הקובץ המתאים במהירות ויעילות. למעשה בגישה בזמן ריצה לקובץ נצטרך לבצע חיפוש רק בין קבצי האות בה המילה מתחילה ולקרוא גודל קבוע של 1MB.

כך עבור כל מילה ניצור רשימת מסמכים בה היא הופיע, עם המידע על איפה היא הופיעה במסמך. נשתמש ב-Ranker כדי לקבל ציון עבור כל מסמך עבור ה-term.

מגישים: רועי דור 301730354, יונתן זקס - 204662779

Ranker Class:

המחלקה מכילה את השדות:

- config – מייצג את ההגדרות של ריצת התוכנה הנוכחית.
- dictionary_document_info_stemmed – מילון: המפתח – DocID
  הValue של המילון – רשימה [max_tf,uniqueTermCount,docLength,city,language]
- dictionary_document_info_withoutStem – כמו המילון הקודם, אבל בלי stem.
- numberOfDoc_Stemmed – כמות המסמכים עם stemming
- averageDocLength_Stemmed – אורך ממוצע של מסמך עם stemming
- numberOfDoc_NotStemmed – כמות המסמכים בלי stemming
- averageDocLength_NotStemmed – אורך ממוצע של מסמך בלי stemming
- dictionary_city_documents_withoutStem – מילון: המפתח – עיר
  הValue של המילון – רשימה של מסמכים עבור העיר
- dictionary_city_documents_Stemmed – כמו המילון הקודם, אבל עם stem.

המחלקה מכילה את הפונקציות:

- __initCitiesIndex(path, dictionary) – פונק עזר לאתחול שדות במחלקה.
- __initNumOfDocs_AvgLength(path,dictionary) – אתחול שדות במחלקה.
- convertDocIndexListToDocID(docNoList : list)-> list – המרת אינדקס לDocName
- getDocumentsFromCityList(citiesList: list)-> set – מחזירה מסמכים עבור עיר
- getScore(docID:str, docDF:int, positionList:list, termDF:int) -> float –
  מחזירה ערך מפונקציית חישוב הציון המרכזית של Ranker, נתונה בביטוי:

$$Score_{Di} = \Sigma(Score_{BM25} + 10 * Score_{Axu})$$

מבדיקות שעשינו גילינו כי ערך המקדם של F2EXP נותן תוצאות טובות עבור הערך 10.

- getBM25Score(docDF,termDF,documentLength, docLengthAvg, numOfDocs) – פונקציה המחשבת ומחזירה את הערך מהפונקציה:

$$\mathrm{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}; \qquad \mathrm{score}(D,Q) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\mathrm{avgdl}}\right)};$$

לאחר הרצה בה בדקנו את כל האפשרויות של ערכי K ו-B, את B בדקנו בערכים שבין 0.2 עד 1 ועבור K מ-0.5 עד ל-2 בקפיצות 0.1, מצאנו כי הערכים שנתנו לנו תוצאת ה-Recall ו-Precision הטובה ביותר הייתה עבור הערכים: K=1.5 B=0.7.

- getAxiomaticScore(docDF,termDF,documentLength,docLengthAvg,numOfDocs)
  פונקציה המחשבת ומחזירה את הערך מהפונקציה:

$$F2EXP(Q,D) = \sum_{t \in Q \cap D} \left(\frac{tf_t^D}{tf_t^D + 0.5 + 0.5 \cdot \frac{dl}{dl_{avg}}} \cdot \frac{N}{df_t}^{0.35} \cdot tf_t^Q \right)$$

מגישים: רועי דור 301730354, יונתן זקס - 204662779

MyWordEmbedder Class:

מחלקה שיצרנו כדי לעזור לתהליך יצירה ושימוש במודל ה-WordEmbedding שבנינו בעצמנו על הקורפוס שלנו בשילוב עם מודל Word2Vec המוכן של GloVe.

ניסינו ליצור מחלקה יחסית גנרית כדי להשתמש במחלקה זאת בהמשך לפרויקטים דומים.

המחלקה מכילה את השדות:

- _modelPath – נתיב למודל שמור שנרצה לטעון או למודל שנרצה ליצור ולאמן.
- _model – מודל Word2Vec הנוצר עם חבילת Gensim.

המחלקה מכילה את הפונקציות:

- setModelPath(self, path:str) – Setter עבור _modelPath
- createModel(self)->bool – פונקציה היוצרת את המודל כפי שצוין באלגוריתם לסמנטיקה, משתמשת בנתיב הנמצא ב-_modelPath. מחזיר ערך בוליאני המשקף האם היצירה עבדה או לא.
- loadModel(self)->bool – פונקציה לטעינת מודל קיים לפי הנתיב הנמצא ב-_modelPath.
- getModel(self)->gensim.models.Word2Vec – פונקציה המחזירה את המודל, _model, כדי לאפשר שימוש יותר גמיש במודל מאשר מה שאנו מאפשרים כרגע. פונקציה למטרה עתידית ונוחות.
- getTopNSimilarWords(self, word:str ,N:int=5)->list or None – פונקציה המקבלת מילה ויכולה לקבל מספר כדי לשנות את מספר המילים הקרובות החוזרות. הפונקציה תחזיר רשימה של מילים בעלות הקרבה המקסימאלית למילה הנתונה לפי המודל, וערך דמיון בין כל מילה שחוזרת למילה הנתונה.
- getTopNSimilarWordsFromList(self, wordList: list ,N:int=10)->list or None – פונקציה המקבלת רשימת מילים ותחזיר רשימת מילים ודמיון עבור כל מילה מוחזרת. הפונקציה בונה וקטור ממוצע מתוך כל הרשימה, וכך המילים המוחזרות מותאמות יותר לקונטקסט של השאילתה כולה.
- visualizeMyModel(self) – פונקציית עזר העוזרת להבין את טיב המודל ע"י יצירת גרף המשקף את המודל.
- otherVisualizationModel(model) – פונקציית עזר נוספת היוצרת גרף המשקף את המודל בצורה טיפה שונה מהקודמת.

מגישים: רועי דור 301730354, יונתן זקס - 204662779

אלגוריתמים במנוע:

- **אלגוריתם למציאת 5 יישויות הדומיננטיות במסמך:**

יישות של מסמך מוגדרת כביטוי שמתחיל באות גדולה בכל ההופעות של במסמך. דירוג יישויות במסמך מתבסס על מספר ההופעות שלנו וגם על המיקומה של כל אחת מההופעות.
תהליך דירוג היישויות מתבצע בזמן האינדוקס של המסמך שניתן לדעת את המיקומים של כל ביטוי במסמך וכן את אורך כל מסמך. לצורך כך יש מבנה נתונים ששומר את ה *Top5* של היישיות במסמך.
האלגוריתם מבצע ריצה איטרטיבית על כל הביטויים(עם אותיות גדולות) במסמך בזמן שה *Indexer* מוסיף ביטויים למיליון הביטויים ונעשית השווה בין הציון הנמוך ב *Top5* והציון שקיבל הביטוי *Ti* .
אם ציונו גובה מהמינימום שברשימה , נכניס אותו לרשימה
אחרת, נכניס את הערך שהוצאנו מהרשימה לצורך ההשווה.
"סכום של אחד פחות המיקום ה-*i* של הביטוי חלקי אורך המסמך D_j. Sigma על כל המיקומים של הביטוי במסמך D_j". נתון בנוסחה:

$$Score_{Ent} = \sum_{i}^{n}\left(1 - \frac{Location(t_i)}{|D_j|}\right)$$

ראינו לנכון להשתמש בנוסחה לעיל, משום שהיא נותנת דירוג גבוהה יותר ככל שמספר ההופעות של ביטוי גדול יותר וגם נותנת ציון טוב יותר עבור הופעות בתחילת המסמך.
דוגמא: שני ביטויים T_1 , T_2. $Loc = \{[2,6] , [7,10,14]\}$ בהתאמה. אורך המסמך 15.
נבחין כי מספר ההופעות של $Df(T_2) > Df(T_1)$

$$Score_{T1} = \left[\left(1 - \frac{2}{15}\right) + \left(1 - \frac{6}{15}\right)\right] = \frac{22}{15}$$

$$Score_{T2} = \left[\left(1 - \frac{7}{15}\right) + \left(1 - \frac{10}{15}\right) + \left(1 - \frac{14}{15}\right)\right] = \frac{16}{15}$$

נראה שלמרות שמספר ההופעות של T_2 גדול ממספר ההופעות של T_1, הוא מקבל ציון גבוה יותר.
כפי שנלמד בהרצאות, יש חשיבות גדולה למיקום הביטויים במסמך.

מגישים: רועי דור 301730354, יונתן זקס - 204662779

- **אלגוריתם דירוג:**

המנוע תומך בשיטות הדירוג הבאות:

1. ***BM25:*** דירוג מסמך לפי:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\text{avgdl}}\right)}; \qquad \text{IDF}(q_i) = \log\frac{N-n(q_i)+0.5}{n(q_i)+0.5};$$

2. ***Axiomatic (F2EXP):*** דירוג מסמך לפי המשוואה, את משוואה זו מצאנו במאמר:

https://pdfs.semanticscholar.org/94c9/30d010c17f3edc0df39ea99fd311d33327c1.pdf

$$F2EXP(Q,D) = \sum_{t\in Q\cap D}\left(\frac{tf_t^D}{tf_t^D+0.5+0.5\cdot\frac{dl}{dl_{avg}}}\cdot\frac{N}{df_t}^{0.35}\cdot tf_t^Q\right)$$

3. ***Locations :*** התחשבות במיקומי הביטויים במסמך עם עדיפות לביטויים מהTitle
4. ***Narrative:*** התחשבות בביטויים שמופיעים בNarrative של השאילתת

אחרי בדיקות רבות מצאנו שהדרך בה מתקבלים ערכי הRecall הגבוהים ביותר, נתונה בביטוי:

$$Score_{(qi,Di)} = Score_{BM25} + 10 * Score_{Axu}$$

עבור B=0.7 , K=1.5

שכן ניסיונות להוסיף את $Score_{Location}$, $Score_{Narrative}$ פגעו משמעותית בערך *# Relevant docs*

במידה ולא משתמשים בהרחבת שאילתה עם סמנטיקה נסכום את ה-*Score* לכל מסמך עבור כל המילים שהופיעו במסמך.

במידה וכן משתמשים בהרחבת שאילתה עם סמנטיקה נתייחס לכל מילה שהופיעה בשאילתה בצורה רגילה ועבור מילים שהגיעו מההרחבה נכפיל את הערך של המסמכים בהם הם הופיעו ב-*Similarity* שקיבלו מה-*Embedding* למילה בכמות המילים שחזרו עבור המילה. כלומר עבור מילים שהגיעו מהרחבת השאילתה נסכום:

$$Score_{(qi,Di)} = (Score_{BM25} + 10 * Score_{Axu}) * similarity(qi, qi') * L$$

$$L = \frac{1}{number\ of\ words\ that\ came\ back\ from\ the\ word\ qi'}$$

- **אלגוריתם לשיפור סמנטי:**

בנינו מודל *Word2Vec* בעזרת חבילת *genism*.

את המודל בנינו באופן הבא:

- העלינו את המילון שלנו לתוך המודל.
- הוספנו למודל את המילים המופיעות במאגר של *GloVe* שנבנה על ויקיפדיה, אותו הורדנו מהאתר הרשמי של *GloVe*, אחרי שעשינו סינון של *StopWords* ועשינו *Stem* למילון כדי לתמוך באחזור עם סמנטיקה ו-*Stemming* לפי הנחיות העבודה.
- לאחר מכן טענו את הוקטורים המוכנים של *GloVe* למודל.
- אימנו את המודל שלנו על הקורפוס שלנו עם הפרמטרים הבאים:
  - איטרציות: 5
  - מספר מימדים: 100
  - מספר הופעות מינימלי: 4
- עבור כל מילה בשאילתה הוצאנו את חמשת המילים הדומות ביותר. מכיוון שהשתמשנו במאגר המכיל מילים שלא בהכרח קיימות אצלנו במילון ביצענו סינון למילים שחזרו כדי לדעת כמה מילים בפועל יש לנו. זאת מתוך כוונה לאפשר תמיכה בחיפוש חופשי של מילים שלא בהכרח מופיעות בקורפוס שלנו ולאפשר למנוע למצוא מילים נרדפות להם וכן לתת להם התייחסות. כאשר אנו מתייחסים למילים שחזרו מההרחבה ננרמל את הציון שהמסמכים שלנו קיבלו לפי הדמיון שהיה למילה שגרמה להחזרת המילה הזאת, במידה ומילה כלשהי חזרה בגלל מספר מילים נבצע ממוצע לדמיון וממוצע לנרמול, ולכמות המילים שבאמת מופיעות אצלנו במילון.
- עבור כל מילה בשאילתה נוציא מהמודל וקטור של המילה וניצור וקטור ממוצע שמייצג ממוצע של כל הוקטורים של כל המילים. נוציא מהמודל 10 מילים הקרובות ביותר לוקטור שמייצג את כלל השאילתה ונבצע נרמול באותו האופן גם למילים אלה. ככה נוכל להוציא מילים שלא היינו מקבלים עבור מילה בודדת, אלא מילים שבאמת קרובות לשאילתה כמכלול.

דוגמה לתוצאות עבור המילה *encrypt* :

(*decod', 0.7813892364501953*)

(*password', 0.7512637376785278*)

(*encod', 0.7155753374099731*)

(*modem', 0.696772038936615*)

(*download', 0.6850541830062866*)

- **אלגוריתמי כיווץ:**

כפי שציינו ב"שיפור התמודדות עם בעיות זיכרון" יישמנו בקבצי הפוסטינג שלנו אלגוריתם דחיסה של *Gap*. יישמנו זאת עבור מיקומים של מילים במסמך ועבור מסמכים להם הקצענו מספר אינדקס והשתמשנו במספר זה כדי לייצג את שם המסמך. למספר זה עשינו *Gap*, ובכך חסכנו כ-45% מכמות הזיכרון שאנו משתמשים כדי לשמור קבצי פוסטינג.

קוד פתוח:

Word2Vec:

עבור יצירה ושימוש במודל *Word2Vec* עשינו שימוש בחבילת *Gensim python*.

את הקוד עבור יצירת המודל, תוך טעינת קורפוס גדול ללא יצירת עומס על הזיכרון והקוד ליצירת ויזואליזציה של המודל לקחנו מהלינק:

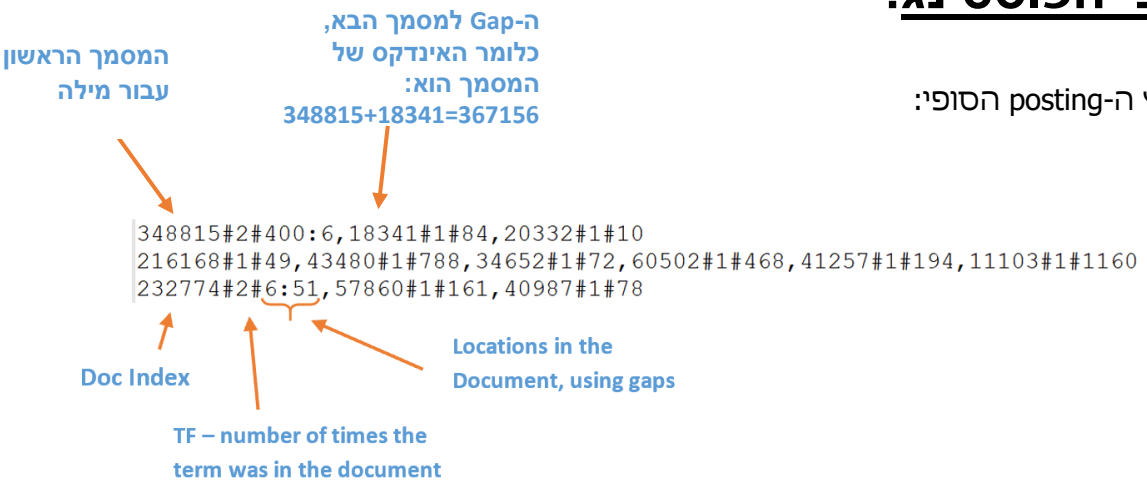
https://rare-technologies.com/word2vec-tutorial

את הקוד עבור טעינת הקורפוס, תוך שילוב מודל קיים לקחנו מהלינק:

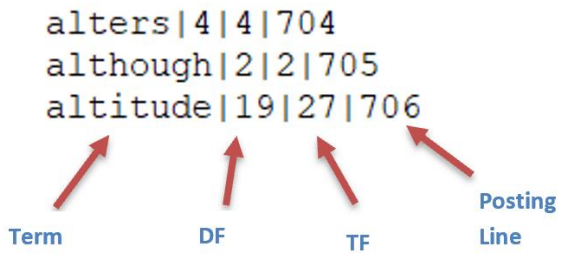
https://gist.github.com/AbhishekAshokDubey/054af6f92d67d5ef8300fac58f59fcc9

נתונים בקבצי הפוסטינג:

דוגמה לשורה בקובץ ה-posting הסופי:



דוגמה לשורה בקובץ המילון:



אחזור מידע – דו"ח מנוע
מגישים: רועי דור 301730354, יונתן זקס - 204662779

דוגמה לשורה בקובץ docIndex:

```
FBIS3-112|5|89|125|LAGOS|English
FBIS3-113|3|38|44|LONDON|English
FBIS3-114|8|95|135|PARIS|French
```

Doc No
Max Freq Term
Number of Unique Terms
Length
City
Language

דוגמה לשורה בקובץ docDominantIndex:

```
(LIBYA:4.05),(HANOI:3.07),(VNA:2.36),(VIETNAM:2.21),(VIETNAM-LIBYA:1.50)
(KOZYREV:14.95),(BEIJING:10.00),(MOSCOW:9.81),(RUSSIAN:7.72),(CHINA:5.18)
(VNA:3.36),(HANOI:2.88),(VIETNAM:2.37),(SUCCESS:1.50),(FIGHT:1.50)
(NORTH:8.34),(PYONGYANG:7.32),(KANG:6.50),(KOREAN:5.86),(SOUTH:4.80)
```

Entity
Score

דוגמה לשורה בקובץ cityIndex:

```
Didcot|United Kingdom|British pound|65.11 M|20888#,64#
Zoetermeer|Netherlands|Euro|17.02 M|20893#,85379#3:9
Karlsruhe|Germany|Euro|81.77 M|20899#23:1599:1215
```

City Name
Country
Currency in the Country
Country Population
Document Index by Condensed Gaps
Positions of City in Document Condensed by Gaps

מגישים: רועי דור 301730354, יונתן זקס - 204662779

דו"ח על ההרצה וביצועים של המנוע:

$Without\ Stemming - 0.4\ (189/691)$						
מס' שאילתא	מילות השאילתא	Precision	Recall	Precision@5	Precision@15	Precision@30
351	Falkland petroleum exploration	$\frac{22}{50} = 0.44$	$\frac{22}{48} = 0.458$	$\frac{5}{5} = 1$	$\frac{12}{15} = 0.80$	$\mathbf{\frac{17}{30} = 0.56}$
352	British Chunnel impact	$\frac{11}{22} = 0.5$	$\frac{11}{246} = 0.04$	$\frac{3}{5} = 0.6$	$\frac{7}{15} = 0.46$	$\mathbf{\frac{8}{22} = 0.36}$
358	blood-alcohol fatalities	$\frac{19}{50} = 0.38$	$\frac{19}{51} = 0.37$	$\frac{4}{5} = 0.8$	$\frac{8}{15} = 0.53$	$\mathbf{\frac{12}{30} = 0.4}$
359	mutual fund predictors	$\frac{0}{20} = 0$	$\frac{0}{20} = 0$	$\frac{0}{5} = 0$	$\frac{0}{15} = 0$	$\mathbf{\frac{0}{30} = 0}$
362	human smuggling	$\frac{9}{50} = 0.18$	$\frac{9}{39} = 0.23$	$\frac{3}{5} = 0.4$	$\frac{5}{15} = 0.33$	$\mathbf{\frac{6}{30} = 0.2}$
367	piracy	$\frac{11}{50} = 0.22$	$\frac{11}{185} = 0.06$	$\frac{2}{5} = 0.4$	$\frac{6}{15} = 0.4$	$\mathbf{\frac{8}{30} = 0.266}$
373	encryption equipment export	$\frac{6}{49} = 0.12$	$\frac{6}{16} = 0.37$	$\frac{4}{5} = 0.8$	$\frac{5}{15} = 0.33$	$\mathbf{\frac{6}{30} = 0.2}$
374	Nobel prize winners	$\frac{29}{50} = 0.58$	$\frac{29}{204} = 0.14$	$\frac{2}{5} = 0.4$	$\frac{8}{15} = 0.53$	$\mathbf{\frac{16}{30} = 0.53}$
377	cigar smoking	$\frac{11}{50} = 0.22$	$\frac{11}{36} = 0.31$	$\frac{4}{5} = 0.8$	$\frac{5}{15} = 0.33$	$\mathbf{\frac{10}{30} = 0.33}$
380	obesity medical treatment	$\frac{7}{50} = 0.14$	$\frac{7}{7} = 1$	$\frac{1}{5} = 0.2$	$\frac{3}{15} = 0.2$	$\mathbf{\frac{5}{30} = 0.16}$
384	space station moon	$\frac{13}{50} = 0.26$	$\frac{13}{51} = 0.25$	$\frac{3}{5} = 0.6$	$\frac{5}{15} = 0.33$	$\mathbf{\frac{7}{30} = 0.23}$
385	hybrid fuel cars	$\frac{23}{50} = 0.46$	$\frac{23}{85} = 0.27$	$\frac{4}{5} = 0.8$	$\frac{12}{15} = 0.8$	$\mathbf{\frac{19}{30} = 0.63}$
387	radioactive waste	$\frac{8}{50} = 0.16$	$\frac{8}{73} = 0.11$	$\frac{2}{5} = 0.4$	$\frac{4}{15} = 0.26$	$\mathbf{\frac{5}{30} = 0.16}$
388	organic soil enhancement	$\frac{8}{50} = 0.16$	$\frac{8}{50} = 0.16$	$\frac{1}{5} = 0.2$	$\frac{4}{15} = 0.26$	$\mathbf{\frac{6}{30} = 0.2}$
390	orphan drugs	$\frac{12}{50} = 0.24$	$\frac{12}{122} = 0.1$	$\frac{5}{5} = 1$	$\frac{8}{15} = 0.53$	$\mathbf{\frac{11}{30} = 0.36}$

Recall	Precision	MAP	R-Precision
189/1241=0.152	189/691=0.273	0.133	0.205

$With\ Stemming - (174/722)$						
מס' שאילתא	מילות השאילתא	Precision	Recall	Precision@5	Precision@15	Precision@30
351	Falkland petroleum exploration	$\frac{23}{50} = 0.46$	$\frac{23}{48} = 0.48$	$\frac{5}{5} = 1$	$\frac{14}{15} = 0.93$	$\mathbf{\frac{20}{30} = 0.66}$
352	British Chunnel impact	$\frac{11}{22} = 0.5$	$\frac{11}{246} = 0.04$	$\frac{3}{5} = 0.6$	$\frac{7}{15} = 0.46$	$\mathbf{\frac{11}{30} = 0.36}$
358	blood-alcohol fatalities	$\frac{19}{50} = 0.38$	$\frac{19}{51} = 0.37$	$\frac{5}{5} = 1$	$\frac{9}{15} = 0.6$	$\mathbf{\frac{14}{30} = 0.46}$
359	mutual fund predictors	$\frac{0}{50} = 0$	$\frac{0}{28} = 0$	$\frac{0}{5} = 0$	$\frac{0}{15} = 0$	$\mathbf{\frac{0}{30} = 0}$
362	human smuggling	$\frac{8}{50} = 0.16$	$\frac{8}{39} = 0.2$	$\frac{2}{5} = 0.4$	$\frac{6}{15} = 0.4$	$\mathbf{\frac{7}{30} = 0.23}$
367	piracy	$\frac{10}{50} = 0.2$	$\frac{10}{185} = 0.05$	$\frac{2}{5} = 0.4$	$\frac{5}{15} = 0.33$	$\mathbf{\frac{8}{30} = 0.266}$
373	encryption equipment export	$\frac{6}{50} = 0.12$	$\frac{6}{16} = 0.37$	$\frac{2}{5} = 0.4$	$\frac{5}{15} = 0.33$	$\mathbf{\frac{6}{30} = 0.2}$
374	Nobel prize winners	$\frac{19}{50} = 0.38$	$\frac{19}{204} = 0.09$	$\frac{3}{5} = 0.6$	$\frac{6}{15} = 0.4$	$\mathbf{\frac{13}{30} = 0.43}$
377	cigar smoking	$\frac{16}{50} = 0.32$	$\frac{16}{36} = 0.44$	$\frac{2}{5} = 0.4$	$\frac{7}{15} = 0.46$	$\mathbf{\frac{10}{30} = 0.33}$
380	obesity medical treatment	$\frac{5}{50} = 0.1$	$\frac{5}{7} = 0.71$	$\frac{2}{5} = 0.4$	$\frac{3}{15} = 0.2$	$\mathbf{\frac{5}{30} = 0.167}$
384	space station moon	$\frac{11}{50} = 0.22$	$\frac{11}{51} = 0.21$	$\frac{3}{5} = 0.6$	$\frac{5}{15} = 0.33$	$\mathbf{\frac{6}{30} = 0.2}$
385	hybrid fuel cars	$\frac{22}{50} = 0.44$	$\frac{22}{85} = 0.25$	$\frac{4}{5} = 0.8$	$\frac{11}{15} = 0.73$	$\mathbf{\frac{20}{30} = 0.66}$
387	radioactive waste	$\frac{7}{50} = 0.14$	$\frac{7}{73} = 0.09$	$\frac{2}{5} = 0.4$	$\frac{3}{15} = 0.2$	$\mathbf{\frac{5}{30} = 0.16}$
388	organic soil enhancement	$\frac{5}{50} = 0.1$	$\frac{5}{50} = 0.1$	$\frac{0}{5} = 0$	$\frac{1}{15} = 0.06$	$\mathbf{\frac{3}{30} = 0.1}$
390	orphan drugs	$\frac{12}{50} = 0.24$	$\frac{12}{122} = 0.09$	$\frac{4}{5} = 0.8$	$\frac{10}{15} = 0.66$	$\mathbf{\frac{11}{30} = 0.36}$

Recall	Precision	MAP	R-Precision
174/1241=0.14	**174/722=0.24**	**0.1332**	**0.203**

מגישים: רועי דור 301730354, יונתן זקס - 204662779

$\boldsymbol{With\ Stemming, with\ Semantic - (173/668)}$						
מס' שאילתא	מילות השאילתא	Precision	Recall	Precision@5	Precision@15	Precision@30
351	Falkland petroleum exploration	$\frac{23}{50} = 0.46$	$\frac{23}{48} = 0.48$	$\frac{0}{5} = 0$	$\frac{6}{15} = 0.4$	$\boldsymbol{\frac{15}{30} = 0.5}$
352	British Chunnel impact	$\frac{11}{22} = 0.5$	$\frac{11}{246} = 0.04$	$\frac{3}{5} = 0.6$	$\frac{7}{15} = 0.46$	$\boldsymbol{\frac{11}{30} = 0.36}$
358	blood-alcohol fatalities	$\frac{19}{50} = 0.38$	$\frac{19}{51} = 0.37$	$\frac{4}{5} = 0.8$	$\frac{10}{15} = 0.66$	$\boldsymbol{\frac{14}{30} = 0.46}$
359	mutual fund predictors	$\frac{0}{50} = 0$	$\frac{0}{28} = 0$	$\frac{0}{5} = 0$	$\frac{0}{15} = 0$	$\boldsymbol{\frac{0}{30} = 0}$
362	human smuggling	$\frac{9}{50} = 0.18$	$\frac{9}{39} = 0.23$	$\frac{4}{5} = 0.8$	$\frac{4}{15} = 0.26$	$\boldsymbol{\frac{7}{30} = 0.23}$
367	piracy	$\frac{8}{50} = 0.16$	$\frac{8}{185} = 0.04$	$\frac{1}{5} = 0.2$	$\frac{2}{15} = 0.13$	$\boldsymbol{\frac{5}{30} = 0.166}$
373	encryption equipment export	$\frac{6}{50} = 0.12$	$\frac{6}{16} = 0.37$	$\frac{3}{5} = 0.6$	$\frac{5}{15} = 0.33$	$\boldsymbol{\frac{6}{30} = 0.2}$
374	Nobel prize winners	$\frac{13}{26} = 0.5$	$\frac{13}{204} = 0.06$	$\frac{1}{5} = 0.2$	$\frac{7}{15} = 0.46$	$\boldsymbol{\frac{13}{30} = 0.43}$
377	cigar smoking	$\frac{16}{50} = 0.32$	$\frac{16}{36} = 0.44$	$\frac{2}{5} = 0.4$	$\frac{8}{15} = 0.53$	$\boldsymbol{\frac{12}{30} = 0.4}$
380	obesity medical treatment	$\frac{5}{50} = 0.1$	$\frac{5}{7} = 0.71$	$\frac{2}{5} = 0.4$	$\frac{3}{15} = 0.2$	$\boldsymbol{\frac{5}{30} = 0.167}$
384	space station moon	$\frac{5}{20} = 0.25$	$\frac{5}{51} = 0.1$	$\frac{2}{5} = 0.4$	$\frac{4}{15} = 0.26$	$\boldsymbol{\frac{5}{30} = 0.16}$
385	hybrid fuel cars	$\frac{27}{50} = 0.54$	$\frac{27}{85} = 0.317$	$\frac{4}{5} = 0.8$	$\frac{13}{15} = 0.86$	$\boldsymbol{\frac{19}{30} = 0.63}$
387	radioactive waste	$\frac{13}{50} = 0.26$	$\frac{13}{73} = 0.178$	$\frac{1}{5} = 0.2$	$\frac{7}{15} = 0.46$	$\boldsymbol{\frac{10}{30} = 0.33}$
388	organic soil enhancement	$\frac{6}{50} = 0.12$	$\frac{6}{50} = 0.12$	$\frac{0}{5} = 0$	$\frac{2}{15} = 0.13$	$\boldsymbol{\frac{4}{30} = 0.13}$
390	orphan drugs	$\frac{12}{50} = 0.24$	$\frac{12}{122} = 0.09$	$\frac{4}{5} = 0.8$	$\frac{7}{15} = 0.46$	$\boldsymbol{\frac{11}{30} = 0.36}$

Recall	Precision	MAP	R-Precision
173/1241=0.14	173/668=0.259	0.1175	0.198

מגישים: רועי דור 301730354, יונתן זקס - 204662779

סיכום:

משום שבחרנו לעבוד בשפה שאין לנו ניסיון רחב בה נתקלנו בהמון בעיות ועיכובים הנובעים מלמידת שפה חדשה והתמודדות עם דרך עבודה שונה ממה שהכרנו. בנוסף נוכחנו לגלות שפייתון היא שפה איטית יותר מ-*Java* לדוגמה, מה שגרם לנו לבעיות ומגבלות רציניות בשלב הפרסור, בו נאלצנו לשקול כל הוספה של בדיקה או שיפור כדי לא לחרוג בזמנים. מגבלה זו גרמה לנו ללמוד ולעבוד עם ספריות פייתון הקשורות במקביליות. בניגוד למה שהכרנו ב-*Java* נאצלנו לעבוד בתהליכים שונים, לא *Thread*, מה שגרם לנו לשנות אסטרטגיה בכל הנוגע לפרסור ואינדוקס המסמכים.
בסופו של דבר אנחנו מרגישים שהבחירה בפייתון הייתה בחירה הנכונה, לא הבחירה הקלה או הנוחה אבל זו בהחלט הייתה החלטה שגרמה לנו לצאת מאזור הנוחות שלנו (*Java*) וללמוד בעצמינו איך ניגשים לעבודה בסדר גודל כזה בשפה חדשה. בנוסף לעבוד בשפה בה מתעסקים בתחום אחזור המידע וניתוח השפה בתעשייה תרם לנו המון והמחיש לנו את כוחה של השפה.

מבחינתנו האתגר הגדול ביותר בפרויקט היה התמודדות עם כמות קבצים גדולה, תוך ניהול הזיכרון בצורה נכונה. האתגר נובע מהצורך לנתח מידע על כמות גדולה מאוד של נתונים והצורך לדגום את המידע בצורה מפוקחת והדרגתית ולבסוף למזג את כל המידע הזה בצורה שתשמור על חוקיות מסוימת לאורך כל המידע שנאסף בצורה חכמה, יעילה וצבירת כמות מקסימאלית של מידע תוך יצירת כמות מינימלית של מידע. אתגר זה גרם לנו לחשוב על כל הוספת פריט מידע וכיצד זה עלול לשנות או לפגוע בתהליך קיים.
זאת בשיתוף האתגר של עמידה בזמנים סבירים עבור כל התהליך בעבודה בפייתון היוו עבורנו את האתגר הגדול ביותר וגרמו לנו לחשוב פעמיים לפני מימוש כל רעיון ומציאת הדרך הנכונה ביותר לממש את אותו רעיון.

כאשר ניגשנו לעבוד על חלק ב' ביצענו כבר הרבה שיפורים שהרגשנו שיכולנו לשפר בחלק א', כמו הגדלת המידע שנאסף, וצמצום משמעותי של גודל קבצי הפוסטינג (ראו הסבר מפורט תחת שיפור בהתמודדות עם מגבלות הזיכרון). כן היינו שמחים לשפר את הפרסור עוד קצת ולהכניס חוקים שחשבנו שיכולים לתרום, אך לא מצאנו את הזמן להתעסק בהם וגם חששנו להאריך עוד את זמן הפרסור והאינדוקס. שכן חלק גדול מאיכות המידע הנאסף קשורה בפרסור נכון ומדויק.
אותנו אישית מאוד עניין להתעסק בבניית מודל ה-*Word2Vec* בעצמינו וללמוד על כלים חזקים שעושים בהם שימוש בתעשייה. היינו שמחים להתעמק בתחום הזה עוד, להמשיך ללמוד עליו ועל האפשרויות הקיימות בו ולשפר את המודל שלנו.