**מנוע אחזור מידע – חלק א'**

**דו"ח מסכם חלק א'**

**עיצוב תוכנה:**

במהלך חלק זה בפרויקט בנינו תוכנית אשר מקבלת כקלט קובץ טקסט אשר מהווה כ URL לתיקייה במחשב אשר מכילה 2 סוגי קבצים שונים:

1. קובץ stop-words אשר מכיל בתוכו מילים נפוצות בשפה אשר אינן בעלות ערך עבורנו ולכן נוריד אותן
2. קבצי data אשר מכילים בתוכם מסמכי טקסט רבים וכן מידע היקפי עליהם כגון שם המסמך, תאריך עריכתו והשפה בו הוא רשום.

התוכנית מכילה את המחלקות הבאות:

**מחלקת MainWindow.xaml**

הView של החלון הראשי של התוכנית, מכיל את הפקדים הרלוונטיים לתפקוד החלון

**מחלקת MainWindow**

הController במודל הזה של החלון ה MainWindow.xaml, אחראי על הלוגיקה הבסיסית מאחורי החלון הראשי

מכיל את השיטות הבאות:

* Browse\_Click(object sender, RoutedEventArgs e)- במקרה של לחיצה על הכפתור BROWSE פותח תיבת דיאלוג ומאפשר לבחור לבד את התיקייה הרצויה
* Run\_Click(object sender, RoutedEventArgs e) -במקרה של לחיצה על כפתור הRUN מריץ את התוכנית אך ורק אם הוזן path לקלט ולמיקום תיקיית הפלט
* showDic\_Click(object sender, RoutedEventArgs e)-במקרה של לחיתה על כפתור הshowdictionary מציג את המילון של התוכנית
* BrowseIndex\_Click(object sender, RoutedEventArgs e)- במקרה של לחיצה נפתחת תיבת דיאלוג ודרכה בוחרים באיזה תיקיה לשים את תיבות הIndex וה Posting.

**מחלקת DictionaryList.xaml**

הView של החלון אשר מציג את המילון בסיום ריצת התוכנית, לפי הדרישה בעבודה. מכיל את הפקדים הרלוונטיים לתפקוד החלון

**מחלקת DictionaryList**

הController במודל הזה של החלון DictionaryList.xaml, אחראי על הלוגיקה של החלון.

מכיל את השיטה DictionaryList(Dictionary<string, term> list) אשר מציגה את הDictionary

**מחלקת Document**

המחלקה אשר אחראית על אובייקט document. הוספנו את המחלקה הזאת בכדי להכיל באופן יותר מדויק וקל בשלל התכונות אשר מכיל כל מסמך (כגון מס' המסמך, תאריך, כותרת וכו'), המחלקה כוללת את כלל השדות הנדרשים לכל מסמך וכן בנאים ואת שיטות הget וset לכל שדה.

**מחלקת Indexer**

המחלקה הזאת אחראית על יצירת קובץ האינדקס וקבצי הposting עבור הterms השונים וכן עבור הערים השונות, והתאמת הterms לאינדוקס.

* Dictionary<string, indexTerm> CreateIndex()- הפונקציה מייצרת את האינדקס ומחזירה את המילון של האינדקס טעון לדיסק
* Dictionary<string, indexTerm> LoadIndex()- הפונקציה טוענת את האינדקס מהדיסק ושמה אותו במבנה נתונים מתאים בזכרון.
* Dictionary<string, indexTerm> LoadIndex(string IndexPath)- המחלקה טוענת את האינדקס מהדיסק ושמה אותו במבנה נתונים מתאים בזכרון. בהיתנן path ייעודי.
* void SortPostings() – הפונקציה הזאת אחראית על מיון כלל הקבצים הזמניים, הדבר נעשה במקביליות (כל קובץ ממוין בפני עצמו באופן מקבילי), בעזרת אלגוריתם quicksort
* void sort(int offset, string[] files, string pathToPosting)- הפונקציה אשר ממיינת קובץ יחיד (נקראת על ידי הפונקציה הנקראת למעלה, על בסיס הסדר הלקסיקוגרפי
* void merge(int fileCount, string[] files) – הפונקציה אחראית על המיזוג של הקבצים הזמניים לקובץ ראשי, נעשה על ידי mergesort בין הקבצים הזמניים הממוינים.
* private string GetPhrase(string line)- מחזיר את השדה phrase עבור הterm בשורה הנתונה.
* private term.Type GetType(string line)- הפונקציה מחזירה את השדה type עבור הterm בשורה הנתונה
* Dictionary<string, indexTerm> Load\_Index(string path)-הפונקציה אשר טוענת את הקובץ לזכרון.
* void sortCity(int offset, string[] files, string pathToPosting)- הפונקציה הזאת אחראית על המיון של הערים השונות בקבצי הערים הזמניים על פי סדר לקסיקוגרפי
* void MergeLocations(string path)- פונקציה זאת ממזגת בין הערים השונות בעזרת merge sort

**מחלקת Model**

מחלקה זאת מרכזת את כלל הלוגיקה של התוכנית, שולטת בחלוקת המשאבים ומנהלת את הTasks השונים אשר מבצעים את עיבוד המידע שבמסמכים השונים ושמירתו

כוללת את השיטות הבאות:

* Void index()- הפונקציה אחראית ליצירה של קבצי האינדקס הזמניים, ניהול ה Taskים שקוראים לפרסר, ניקוי הזכרון וכתיבה לדיסק.
* Void manageResources()- הפונקציה אחראית לאיחוד כל מבני הנתונים שכתבו ה Taskים למבנה אחד של Termים והכנתו לכתיבה לקובץ, הפונקציה מנקה את מבני הנתונים של ה Taskים.
* Dictionary<string, term> getDictionary()- הפונקציה מחזירה מילון עם אינדקס מוכן וטעון.
* bool hasIndex()- הפונקציה בודקת אם קיים אינדקס מוכן, מחזירה TRUE אם קיים, FALSE אם לא.
* Void memorydump()- הפונקציה לוקחת מוחקת את כלל הקבצים שייצרנו (זמניים וקבועים) בהינתן הpath וכן מנקה את כלל המילונים ומבני הנתונים שהשתמשנו בהם כדי לשמור על המידע

**מחלקת Parse**

מחלקה זאת אחראית על פירסור כלל הקבצים במאגר. המחלקה מאותחלת ובונה את רשימת הstop words על פי המידע שניתן לה, ולאחר מכן מקבלת מסמך כלשהו ועבורו מייצר רשימה של terms סופיים לפי ההנחיות שניתנו בגוף העבודה, במהלך העבודה הparse שומרת את קבצי הposting הזמניים שנוצרו לה מכל מסמך בתיקייה זמנית, בכדי לא לשמור אותם בזכרון.

כוללת את השיטות הבאות:

* Parse(string path, bool tostem)- הבנאי קורא את רשימת הstopwords
* void Text2list(document document, int queue)- פונקציה מקבל אובייקט מסוג מסמך והופכת אותו לרשימה של מילים המופרדות על ידי מספר דלימטרים
* bool IsNumber(string str)-הפונקציה בודקת האם המילה שקיבלנו היא מספר המכיל נקודה, אם כן מחזירה True
* void parseText(string[] words, int queue)- פונקציה זאת היא הפונקציה הראשית של המחלקה, היא בודקת לכל מילה במערך על איזה חוק היא עונה אם בכלל ומטפלת בה בהתאם (בעזרת כלל הפונקציות הרשומות בהמשך) , את המושג הסופי מוסיפים לDictionary לאינדוקס
* bool isDate(string[] words, int idx)- הפונקציה בודקת האם המילה שקיבלנו היא חלק מterm מסוג תאריך, אם כן מחזירה True
* bool isPercentage(string[] words, int idx)- הפונקציה בודקת האם המילה שקיבלנו היא חלק מterm מסוג אחוז, אם כן מחזירה True
* bool isPrice(string[] words, int idx)-הפונקציה בודקת האם המילה שקיבלנו היא חלק מterm מסוג מחיר, אם כן מחזירה True
* bool containsNumbers(string s)-הפונקציה בודקת האם המילה שקיבלנו כוללת ולו ספרה אחת בתוכו, אם כן מחזירה True
* bool IsRegNumber(string[] words, int idx)-הפונקציה בודקת האם המילה שקיבלנו היא מספר שאיננו אחוז או מחיר או מרחק, אם כן מחזירה True
* bool IsComNum(string input)-פונקציה זאת בודקת האם המילה שהיא קיבלה הוא מספר רגיל המכיל ספרות או נקודה או שבר, אם כן מחזירה True
* string Isprecent(string[] words, int idx, out int j)-פונקציה זאת מטפלת במקרה של אחוז ושומרת אותו כterm לפי החוקים והכללים שניתנו בעבודה
* string NumberSet(string input, int idx, string[] words, out int j)- פונקציה זאת מטפלת במקרה של מספר רגיל ושומרת אותו כterm לפי החוקים והכללים שניתנו בעבודה
* string ToDate(string firstTerm, string secondTerm)- פונקציה זאת מטפלת במקרה של חוקים ושומרת אותם כterm לפי הכללים שנקבעו בעבודה
* string Setprice(int idx, string[] words, out int j)-פונקציה זאת מטפלת במקרה של מחיר ושומרת אותם כterm לפי החוקים והכללים שניתנו בעבודה
* string SetExp(int idx, string[] words, out int j, int queue)- פונקציה זאת מטפלת במקרה של טווח, כשאר היא מוסיפה כterm את הטווח עצמו, וכן בודקת האם כל מילה בטווח (מ2 צדדי ה"-") עובד באחת החוקים של הפרסר בעצמו, אם כן הוא מפעיל את החוק עליהם ומטפל בהם כפי שצריך
* void AddTerm(int queue, term t)- הפונקציה מוסיפה Term למילון, בודקת אם קיים ואם כן מעדכנת את הערכים המתאימים.
* string TrimAll(string word)-פונקציה זאת מקבלת מילה ומעבדת אותה על ידי הורדת כלל הסימנים שאינם אות או מספר וכן מורידה את כלל המקומות הריקים. פונקציה זאת בניגוד לפונקציות עיבוד מטה מורידה גם סימני פיסוק שונים
* string Fixword(string word)- פונקציה זאת מקבל מילה, עושה לה עיבוד על ידי הורדת סימני קצה, בדיקת מקרי קצה ואם היא עומדת בתנאים מסוימים (לא נופלת בתאי קצה) שולחת אותה לפונקציה TrimtoRemove.
* double FormatNumber(string num)- פונקציה זאת מקבל מספר כstring והופכת אותו לdouble על ידי הורדת כלל התווים שאינם ספרה, פסיק או נקודה. באופן יעיל וללא שגיאות (מחליפה ומשפרת את הפונקציה TryPharse של c# כאשר היא תמיד מחזירה את המספר.
* string TrimtoRemove(string word)- פונקציה זאת מורידה מכל מילה שהיא מקבלת את כלל התווים המיותרים בהתאם לכללים (לא מוריד נקודה או פסיק בתוך מספקים, לא מורידה "-" בתחילת מילה ועוד מקרים כאלה אשר עלולים להשפיע על הערך הסימנטי של המילה. פונקציה זאת מהווה תחליף לפונצקיה replace או remove של c# משום שפונקציות אלו מוסיפות זמן ריצה רב לתוכנית.
* bool hasChar(string word, char del)- פונקציה הזאת מהווה תחליף לפונקציה contains של c#, משום שcontains מעלה רבות את זמן הריצה במהלך ריצת התוכנית אז ייצרנו תחליף יעיל יותר.
* public static string replace(string str, char remove) – פונקציה שאנחנו כתבנו שמחליםה את ה replace של המחלקה string, לצרכי ייעול.

**מחלקת ReadFile**

המחלקה הזאת מקבלת את הpath של התיקייה אשר מאכסנת בתוכה את כלל קבצי הdata ומתוך קובץ זה מביאה את רשימת הpathes של כלל הקבצים, קוראת אותם, מייצרת עבור כל שדה text של מסמך מערך של מילים. לסיום היא שולחת את האובייקט document למחלקת הparse לעבור עיבוד.

כוללת את השיטות הבאות:

* int returnSize()- פונקציה זו מחזירה את כמות כקבצים שיש בקורפוס (לא מסמכים, קבצים)
* void readfile(string file2, int queue)-פונקציה זאת מקבל קובץ, מפרידה אותו על בסיס מסמכים ומכל מסמך מייצר אובייקט document על בסיס המידע בקובץ.
* void addLocation(string city)-פונקציה זאת בודקת מול הapiים ומייצרת אובייקט מסוג location על בסיס הנתונים, ומוסיפה אותו למילון הרלוונטי
* string rmvStr(string input)- פונקציה זאת מקבל מילה ומורידה ממנה את כלל הסימנים אשר אינם נדרשים (על בסיס מילון קבוע שקיים).

**מחלקת Stemmer**

המחלקה הזאת מקבלת מילה ומעפילה עליה את חוקי הstemming אשר נלמדו בקורס, לאחר מכן מחזירה מילה מתוקנת לפי חוקים אלו, המחלקה הזאת היא המחלקה שניתנה במודל

**מחלקת Term**

המחלקה אשר אחראית על אובייקט term. הוספנו את המחלקה הזאת בכדי להיות מסוגלים לבצע פעולות שונות על כל term כגון hashing והשוואת, וכן בכדי להחזיק בצורה יותר מסודרת מידע על המושג כגון מהו הterm וכן ,idf,icf,isUpper

המחלקה כוללת את כלל השדות הנדרשים לכל מושג ,בנאים , את שיטות הget וset לכל שדה, וכן override לשלל פונקציות בכדי לאפשר פעולות מגוונות יותר על האובייקט.

**מחלקת Location**

המחלקה שאחראית על אובייקט location. הוספנו את האובייקט הזה כדי שנוכל לשלוט ולרכז יותר טוב את התכונות השונות והמידע שעלינו לשמור לכל עיר אותה אנחנו מוצאים בקורפוס. המחלקה כוללת את כלל השדות country, population, currency, capital, city וכן רשימת מיקומים שבהם מופיע העיר בכל מסמך. כמו כן המחלקה כוללת בנאים , את שיטות הget וset לכל שדה.

**מחלקת IndexTerm**

המחלקה אחראית על אובייקט indexTerm שהוא כמו אובייקט הterm רק בעל שדות מעטים יותר בכדי שיהיה קומפקטי ונוח לשמירה לזיכרון. כולל את השדות ifc.idf.pharse.type וכן בנאים ושיטת tostring

**ממשק IRule**

כדי לשמור על המודולריות, יצרנו את הממשק IRule למימוש תבנית Strategy כך שהפרסר יוכל לבדוק את חוק שאנחנו יצרנו, חוקים שהתבקשנו לממש בעבודה נכתבו ידנית ללא מימוש ה Strategy.

הממשק מגדיר את הפונקציות הבאות:

* bool CheckRule(bool isNum, string[] words, int idx); - הפונקציה בודקת אם להפעיל את החוק על ה term, לצרכי ייעול, isNum זה המשתנה הבוליאני שנבדק מראש האם המשתנה הוא מספר או לא. תחזיר True אם יש להפעיל את החוק, False אם לא.
* string CreatePhrase(string[] words, int idx, out int j, out term.Type type); - הפונקציה שמפעילה את החוק, מחזירה את ה term, הסוג שלו ואת ה token הבא במערך.

את ה Strategy מממשים רק חוקים שאנחנו המצאנו, בזמן התהליך פיתחנו וביטלנו הרבה חוקים, כך עשינו את זה בצורה יעילה.

מחלקות ההאסטרטגיה: **TimeRule, Distance**

**מחלקה סטטית RuleFactory**

מחלקה זו היא מחלקה סטטית המממשת תבנית Factory, כדי לא ליצור את המחלקות ב Parser, אנחנו עורכים את ה Factory כדי שיחזיר את כל החוקים הקיימים.

* public static List<IRule> Factory() – הפונקציה מייצרת רשימה של כל החוקים הפעילים המממשים את IRule ומחזירה אותה.

כדי לשמור על מודולריות, כל החוקים שאנחנו כותבים יממשו את הממשק IRule, כאשר האובייקט לאלגוריתם ייוצר ויוחזר ע"י ה Factory, כך ה parser בכל ריצה מקבל את כל החוקים ללא שום צורך לערוך אותו hard coded.

התוכנית שומרת לאחר סיום העבודה על הקורפוס את המילון של כלל הterms שמצאנו, את קבצי הposting לכל הterms , את המילון עבור כלל הערים מצאנו וכן את קובץ הposting עבור הערים.

שיפור זמן הריצה: בכדי לשפר את זמן הריצה של התוכנית עשינו את הפעולות הבאות:

* שימוש בHashSet וDictionary , לאחסון כלל המשתנים והאיברים שאיתן אנו עובדים וכן לשמור את המילים ה"שמורות" כגון החודשים, גדלים כגון אלפים ומיליונים ומרחקים. השימוש במבני הנתונים הללו אפשר לנו לחפש את אותם המילים ולהשוות אליהם בזמן ריצה של O(1) לכל חיפוש במקום O(n)
* עבודה עם Tasks אשר הם מהווים מעין ט'רדים אשר מנוהלים באופן יעיל יותר על ידי מערכת ההפעלה, דבר אשר אפשר עבודה במקביל על מס' קבצים.
* החלפה של הפונקציות הקיימות של C# (כגון contains) לפונקציות יעילות יותר שאנחנו כתבנו
* הימנעות משימוש מיותר בפונקציות או מעבר מיותר על המסמכים.
* מיון הקבצים ב Indexer ע"י quick sort לצורך ביצוע merge יעיל

הקטנת הזיכרון המשומש בתוכנית: בכדי להקטין את כמות הזיכרון שבו השתמשנו בתוכנית עשינו את הפעולות הבאות:

* שימוש ב Dictionary בכדי להימנע מכפילויות ברשימות.
* שימוש ב Tasks בתהליך הפירסור לצורך הקטנת מספר קבצי הפלט הזמניים עם ה Termים שעברו פירסור
* בכל קבוצת Taskים שמסיימים עבודה, מתבצע תהליך ניקוי זיכרון וכתיבת ה Termים עם ה posting lists הזמניים לקובץ פלט זמני
* לאחר סיום השימוש במבנה נתונים ניקינו אותו.

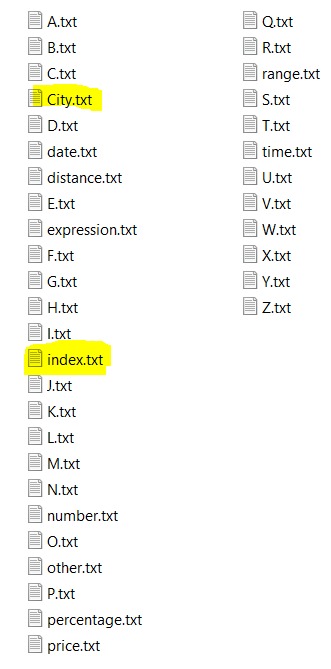
שמירת קבצי הposting:

את קבצי הposting שמרנו על גבי 36 קבצי טקסט.

כאשר שמרנו קובץ לכל אות, קובץ כל type של term לפי חוקי הפירסור שניתנו לו (קובץ לPRICE, וPERCENTAGE ועוד), קובץ לכל המספרים יחדיו ועוד קובץ מיוחד לterms מיוחדים אשר לא הצלחנו לזהות (תווים מיוחדים).

כמובן ביצענו את קבצי הposting ללא כפילויות כאשר term עונה על אחד התנאים הוספנו אותו לקובץ הposting הזה בלבד.

ביצענו בחירה זאת על מנת לצמצם ככל הניתן את קבצי הposting ובו בזמן לשמור עליהם נכונים ומדויקים לדרישה ובכדי לאפשר אחזור מהיר של המופעים בהינתן המושג הרלוונטי.



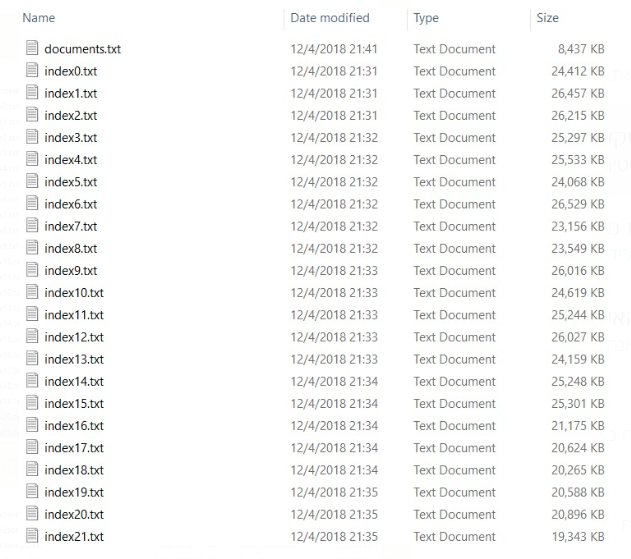
בתמונה כלל קבצי הposting הסופיים וכן קובץ האינדקס, המודגשים אלו קבצי האינדקס ו posting הערים.

קבצי האינדקס של הקבצים ומילון הערים עם הנתונים שלהם.

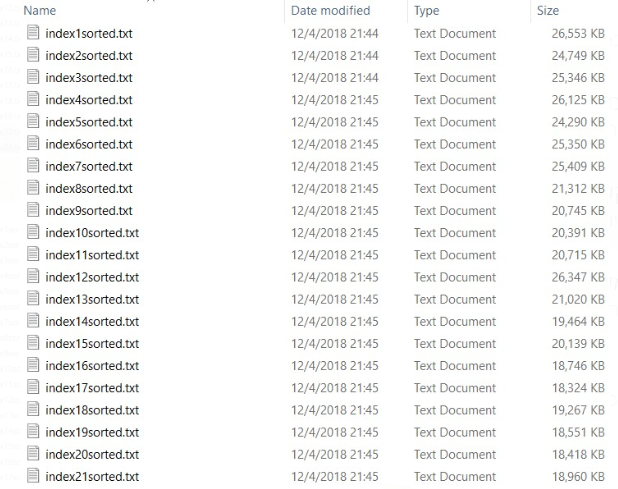
בחירת גודל קבוצת מסמכים חלקית

גודל קבוצת הקבצים עליה עובדים ה Taskים הוא כגודל המשתנה cores המחזיק את כמות המעבדים הלוגיים של המחשב, כל cores קבצים עובדים במקביל, מפורסרים, וה Termים המעובדים מכל קובץ נכנסים למילונים, המילונים מאוחדים ונכתבים בקובץ חלקי כל 5 איטרציות, משמע, Termים מפורסרים מכל cores\*5 קבצים נכתבים לקובץ חלקי אחד. הסיבה לבחירת המספר הזה היא ניצול יעיל של זמן מעבד וזכרון, היה טרייד אוף בין זמן הריצה (שימוש בכמות עצומה של זכרון) ושימוש מינימלי בזכרון (זמן ריצה גבוהה מאוד), אמפירית הגענו למסכנה שהזכרון והמעבד מנוצלים הכי טוב בכמות core\*5 קבצים מפורסרים בזכרון (המילונים מסייעים להורדת האובייקטים החיים אצלנו בתוכנית מעצם מניעת הכפילויות) עם פיק של 1 גיגה זכרון ומצב יציב של 600 מגה.

דבר זה גם מסייע להקטין משמעותית את כמות הקבצים החלקיים וכך למנוע ריצה ארוכה מאוד של האיחוד שמבצע מיון של הקבצים ואיחוד יעיל הקבצים.



בתמונה חלק מהקבצים הזמניים שהשתמשנו בהם, ניתן לראות שכולם בערך באותו הגודל

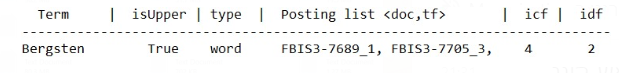


בתמונה הקבצים הזמניים הממויינים (כל קובץ ממוין)

פריטי אינפורמציה נוספים

2 פריטי האינפורמציה על הterms שהוספנו לעבודה זאת הן:

* Icf: הוספנו לכל term את כמות ההופעות הכללית שלו בקורפוס, נתון זה יאפשר לעשות טווח רחב יותר של חישובים בקלות רבה יותר, וכן למדוד באופן מדויק יותר כמה משקל המילה אמורה לקבל וחשיבותה לקורפוס כולו וייתן את האפשרות **לעשות smoothing.**
* Type: הוספנו לכל term את סוג הterm שלו לפי חוקי הפרסר (האם או אחוז או מחיר וכו'), פריט מידע זה יאפשר לנו למיין מידע באופן ממוקד ומדויק יותר וכן לעשות חיפוש אך ורק על סוגים מסוימים של terms. ובכך להאיץ ולייעל את החיפוש



Term: המונח עצמו

IsUpper: האם המונח מופיע בכל הקורפוס עם אות גדולה( במקרה הזה כן)

Type: סוג הterm מבין כל חוקי הparser

Posting list: הקובץ בו נמצא המושג, בצירוף עם כמות המופעים של מושג במסמך

Icf: כמות המופעים הכללית של הterm בקורפוס

Idf: כמות המסמכים שבהם מופיע הterm

חוקים נוספים

2 החוקים שהוספנו לשלב הparser של הפרויקט הם:

זמנים: כאשר יש התייחסות לזמנים או מושג בנוגע לזמן בטקסט נשמור אותו כterm מיוחד, לדוגמה" בהינתן המשפט הבא "we waited for 100 million years"

ישמרו הterms הבאים:

We,waited,for,100 million years. כלומר 4 terms שונים.

(הנחה כי אין stop-words)

מימשנו זאת על ידי שמירת יחידות זמן רבות (שניות,מילישניות וכו') וכאשר קיבלנו מספר ולאחריו הסממן הזה או כמת (אלף,מיליון,מיליארד) ואז הסממן נייצר מזה term ייחודי

הרעיון פה הוא יצירת term מיוחד עבור מקרים של שאילתות הכוללות זמן, ללא טיפול זה לא נוכל להפריד בין מספר רגיל לזמן.

* מרחק: כאשר יש מרחק בטקסט נשמור את המרחק כterm מיוחד

לדוגמה: בהינתן המשפט הבא "every person has about 168 cm colon"

ישמרו הterms הבאים:

Every, person, has, about, 168cm, colon

(הנחה כי אין stop-words)

מימשנו זאת על ידי שמירת הרבה מהסממנים האפשריים למרחק (מטר,סנטימטר,אינצ' וכיוצ"ב). וכאשר קיבלנו מספר (אשר איננו כולל סימן כמו $) ולאחריו הסממן הזה ייצרנו מ2 המילים הנפרדות האלה term.

הרעיון פה הוא יצירת term מיוחד עבור מקרים של שאילתות הכוללות מרחק, ללא טיפול זה לא נוכל להפריד בין מספר רגיל למרחק.

דוגמאות מהקוד:

ציטוט:

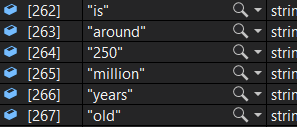
1. ישנו את הטקסט הבא:



הterm שייווצר הוא



ישנות את הטקסט הבא:



הterm שייוצר משורות 264-266 הוא:



מרחק:

1. יש את הטקסט הבא : 

המנוע יבנה מהמילים במקום ה237 ו 238 את הterm הבא:



1. בהינתן הטקסט הבא:



המנוע יבנה מהמילים במקום 245 ו246 את הterm הבא:



OPEN SOURCE:

במהלך התוכנית השתמשנו בpackage NSOUP לקריאה של קבצי JSON וכן בNewsoft.JSON לפרסר את הקבצים, כמו כן השתמשנו ב2 api שונים בכדי לבדוק את המידע על ערים שונות.

<https://restcountries.eu/#sources>

וכן ב <http://geobytes.com/>

בכדי לקבל מידע כמה שיותר נכון.

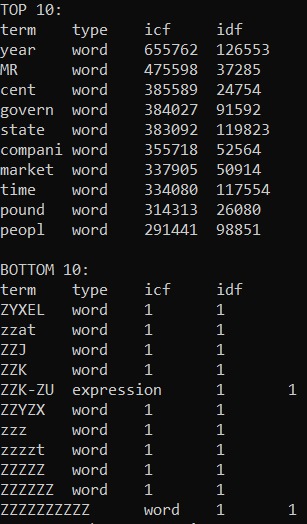
**פלטים:**

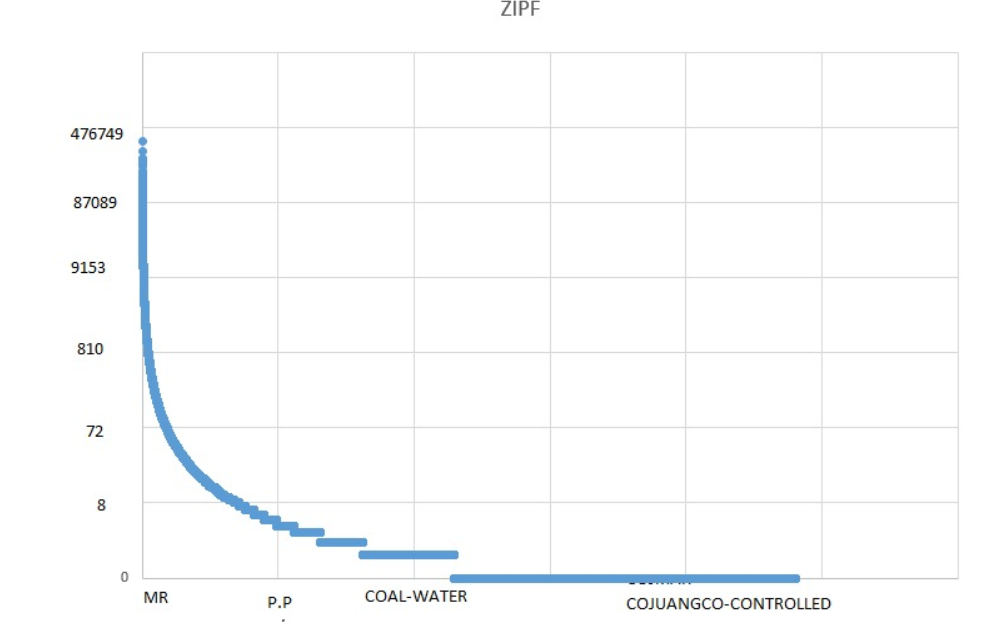
1. ישנם 1,057,302 terms ללא stemming
2. ישנם 958,628 terms עם stemming
3. ישנם 25,296 terms ייחודיים שהם מספרים
4. במאגר ישנם 172 מדינות שונות
5. במאגר ישנם 318 ערים שונות ומתוכן 155 ערים שאינם ערי בירה
6. שם העיר הוא Moscow אשר מופיעה 263 פעמים במסמך FBIS3-60342

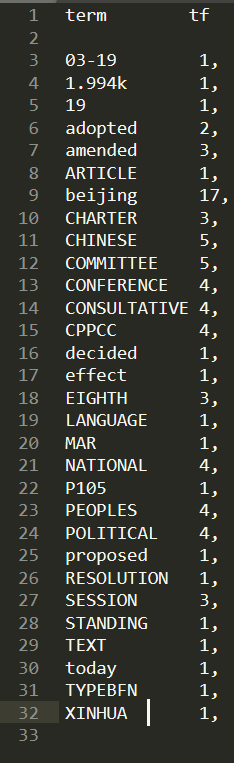
מיקומים במסמך:

36-63-101-143-243-879-910-956-992-993-4030-5040-7366-8449-9257-9504-9554-9697-10009-10377-10382-11546-11554-11902-11953-12016-12026-12272-12326-12385-12461-12488-12592-13360-13389-13761-13915-13931-13944-14245-14371-14917-14970-15124-15503-15634-16576-16932-16955-17040-17198-17383-17791-17979-18205-18208-18252-18447-20236-21132-21565-21667-22105-22235-22612-22637-22981-23038-23313-23549-23757-23779-24192-24219-24445-24464-24472-24683-24724-25023-25276-25288-25635-25666-25688-25739-25750-25757-25769-25780-25785-25809-25815-25836-25841-26731-27099-27393-35410-36514-37426-40442-46059-53520-54394-54859-55153-55417-55607-56045-56085-56160-56945-57299-57423-57973-57988-58001-58088-58126-58159-58184-58236-58304-58376-58412-58432-58441-58574-59112-59125-59211-59415-59464-59465-59468-59489-59533-59655-59760-59785-59862-59878-59896-59940-59958-59999-60174-60312-60543-60587-60625-60661-60680-60683-60711-60726-61160-61380-61385-61462-61524-61547-61566-61589-61604-61641-61647-61712-61728-61818-61844-61867-61885-61905-61986-62181-62219-62285-62312-62374-62575-62701-62713-62953-63069-63118-63152-63217-63237-63350-63456-63467-63537-63933-64035-64042-64296-64457-64602-64694-64728-64959-65000-65048-65078-65109-65249-65259-65339-65485-65563-65628-65670-65794-69212-73525-74269-80749-82385-83522-88043-88145-88355-89280-89339-89390-89535-90836-91311-100727-101632-115077-117296-127565-128094-128827-128878-129266-129507-129521-129524-129534-129537-129540-129544-129552-129557-129765-129978-130172-130388-130815-131248-131276-131776-131968-132027-132148-132918-133250-133502-133595

1. \*\* בicf הערכים נמוכים ב1 מהמציאות (הספירה מתחילה ב0)



1. .
2. מצורף צילום של מסמכים ממוינים לפי term ולידם הtf



1. גודל הposting של כלל הקבצים (כל ה39) הוא 1,147,803,081 בייט לאחסון על הדיסק (כולל מטה דאטא) שזה 1.06 ג'יגה

