

פרויקט מנוע אחזור מסמכים

יוגב שלומוביץ 311530018

איתי כרמי 205632003



December 9, 2018

קורס אחזור מידע

המחלקה להנדסת מערכות תוכנה ומידע

**חלק א'**

**עיצוב התוכנה:**

**פירוט הפונקציות והמחלקות בפרויקט**

* **מחלקת Read File:**

מחלקת זו קוראת את כלל הקבצים ממאגר הקבצים.

def \_\_init\_\_(self, folder\_path)

שיטה זו מבצעת את אתחול המחלקה. השיטה מקבלת נתיב של תיקייה אשר מכילה את מאגר הקבצים. מחלקה זו מחלקת את מאגר הקבצים לקבוצות של 50 קבצים וזאת על מנת ליעל זמן ריצה ולאפשר למחלקה הבאה לעבוד על מספר קטן יותר של קבצים.

def read(self):

שיטה זו מבצעת את קריאת הקבצים וקבלת תגיות מכלל מאגר הקבצים. בשיטה זו נעבור על קבוצת הקבצים שקיבלנו לאחר האתחול ועבור כל קובץ נוציא את המידע הרלוונטי אותו אנו מחפשים כגון: שפה, עיר, שם מסמך, וטקסט. בשיטה זו ישנו מבנה נתונים של מילון שבו עבור כל שם קובץ (key) נשמור את הטקסט שלו (value) ונבצע זאת לכל קבוצת קבצים ולבסוף נחזיר את רשימת הטקסטים שתועבר אל המחלקה הבאה ואת המידע עבור תגיות העיר והשפות שקיימות בקבצים אלה.

def read\_city\_language(self, save\_path):

שיטה זו מבצעת את קריאת הערים והשפות מכלל מאגר הקבצים. בשיטה זו נעבור על כלל הקבצים ונאסוף את המידע עבור השפות והערים השונות וזאת על פי התגיות השונות שהוגדרו מראש. בנוסף על כך ניצור קבצים אשר יכילו את המידע עבור השפות והערים שיש במאגר הקבצים. יתר על כן עבור הערים שקיימות בapi- נשמור בקובץ חדש את המידע הרלוונטי: המדינה, גודל האוכלוסייה ואף את מטבע המדינה. כלל המידע יישמר כקובצי טקסט בנתיב שהוזן עבור שמירת הקבצים.

def read\_api(self):

שיטה זו מבצעת את קריאת המידע מכתובת הapi-. בשיטה זו השתמשנו בפלטפורמת JSON שבעזרתו ניתן לשאוב מידע משהשרת(אתר אינטרנט בעל כתובת http) ללקוח. שיטה זו מכניסה את כל המידע עבור הערים אל מילון אשר מייצג את מאפייני הערים ולבסוף נחזיר מילון זה.

* **מחלקת Parser:**

מחלקה זו עוברת על הקבצים ומוציאה את הביטויים הרלוונטיים בכל קובץ.

במחלקה זו נחזיר רשימות עבור ה-terms השונים עבורם אנו נדרשים לסווג למקרים השונים על סמך כללי המנוע.

def \_\_init\_\_(self, index, sw\_path)

שיטה זו מבצעת את אתחול המחלקה. השיטה מקבלת נתיב של תיקייה אשר מכילה את מאגר הstop words עבורן אין צורך לבצע בדיקה והכנסתן למילון. מחלקה זו מקנה מספר סידורי לכל אובייקט של המחלקה שנוצר. בנוסף נוצרת רשימה של stop words ונוצר מילון עבור כלל הביטויים שבמאגר הקבצים.

def parse(self, docs\_texts):

שיטה זו מקבלת כפרמטר את רשימת הטקסטים שהוחזרה ממחלקת הread file. שיטה זו אחראית על מהלך פירסור הטקסטים בכך שהיא מתווה את מהלך הפירסור. עבור כל קובץ ברשימת הטקסטים אנו נקבל את הביטויים הרלוונטיים ונבצע ניתוח עבור המסמך ולבסוף נמיין את קובץ הביטויים. לאחר מכן נבצע שמירה של כלל הביטויים שקיבלנו במשתנה. שיטה זו תיצור רשימת ביטויים חדשה עבור כלל הטקסטים של 50 הקבצים שקראנו ואותם נרשום לקובץ.

@staticmethod  
def get\_tokens(doc\_text):

שיטה זו הינה שיטה סטטית עבור המחלקה הנ"ל. שיטה זו מקבלת את קובץ הטקסט עבור קובץ כלשהו ומבצעת הפרדה split לפי רווח. בנוסף, השיטה מוחקת tokens ריקים (במקרים שבהם יש מספר רווחים צמודים

def parse\_document(self, doc\_name, tokens)

שיטה זו אחראית על ניתוח הטקסט. שיטה זו מקבלת את שם הקובץ והביטויים השונים שהוא מכיל ומסווגת אותם למקרים לפי התו הראשון בכל ביטוי. סיווג הביטויים הינו לשלושה מקרים. האחד הינו לביטוי שמתחיל במספר, השני הינו לביטוי שמתחיל באות והשלישי הינו לביטוי שמתחיל בסימן ("$","%"). עבור כל מקרה נקרא לפונקציות השונות שמטפלות במקרים אלה. שיטה זו מקבצת את כלל הביטויים למילון. נחזיר את רשימת הביטויים העדכנית לאחר ביצוע הפירסור.

def add\_to\_dictionaries(self, terms\_to\_add, doc\_terms, doc\_name, term\_index):

שיטה זו אחראית על הוספת ביטויים למילון. שיטה זו מקבלת ביטוי להוספה, מילון הביטויים עבור הקובץ הנוכחי, שם הקובץ ואת מקומו של הביטוי. בבעת ההוספה סוכמים את כמות הפעמים בהן הופיע הביטוי באותו המסמך וכך יוצרים במילון רשימת מופעים של מיקומיו של אותו הביטוי. במידה והביטוי לא מופיע במילון נשמור את המיקום שלו ותדירותו תהיה 1. שיטה זו אחראית לשמור ביטויים אשר הופיעו תחילה באותיות גדולות ולאחר מכאן באותיות קטנות, באותיות קטנות וזאת תוך איחוד המילונים המכילים את שמות המסמכים והמיקומים של הביטוי. לבסוף נחזיר עבור כל מסמך את רשימת הביטויים שבמסמך המיקום הנוכחי של הביטוי שהתווסף ובנוסף את הביטוי שהופיע הכי הרבה פעמים עד שלב זה ואת כמות מופעיו.

def number\_parser(self, tokens, token, index):

שיטה זו מטפלת בביטויים אשר מתחילים במספר. שיטה זו מקבלת רשימת ביטויים, ביטוי ומקומו. עבור כל מספר נסווג בין המקרים השונים: באים מדובר בטווחים, באחוזים, בתאריכים, במחירים, במספר רגיל או בביטוי סתמי אשר מכיל מספר בתו הראשון שלו. נחזיר את רשימת הביטויים העדכניים.

@staticmethod  
def legal\_number(token):

שיטה זו אחראית לבדוק האם מדובר במספר חוקי. שיטה זו מקבלת את ביטוי ומחזירה ערך בוליאני האם הוא חוקי או לא.

@staticmethod  
def legal\_money(token):

שיטה זו אחראית לבדוק האם מדובר במספר (כסף) חוקי. שיטה זו מקבלת את ביטוי ומחזירה ערך בוליאני האם הוא חוקי או לא.

def number\_parser\_range(self, tokens, token, index):

שיטה זו מטפלת במקרה של טווחים. שיטה זו מקבלת רשימת ביטויים, ביטוי ומיקומו. שיטה זו מכילה רשימת סממנים עבור תאריך אחוז ומחיר וכך מבדילים בין המקרים השונים ואף בין טווחים של מספרים רגילים. עבור כל מקרה תישלח רשימת הביטויים והביטוי החדש שנוצר בשיטה, אל שיטה במחלקה שמטפלת במקרה הספציפי הזה. במקרה שמדובר בטווח מספרים רגיל אנו נטפל בזה כפי שהוגדר, נכניס אל רשימת הביטויים את שני המספרים ואף את המספרים עם "-". נחזיר את הביטוי העדכני ונקדם את האינדקס.

@staticmethod  
def contains\_letter(token):

שיטה זו בודקת האם הביטוי מכיל אותיות. שיטה זו מקבלת ביטוי ועבורו עוברת על כלל התווים שלו ואם ישנה אות כלשהי יוחזר ערך בוליאני של TRUE, אחרת יוחזר FALSE.

@staticmethod  
def contains\_only\_numbers(token):

שיטה זו בודקת האם הביטוי מכיל רק מספרים. שיטה זו מקבלת ביטוי ועבורו עוברת על כלל התווים שלו ואם ישנה אות כלשהי יוחזר ערך בוליאני של FALSE, אחרת יוחזר TRUE.

@staticmethod  
def range\_parts\_are\_numeric(range\_parts):

שיטה זו בודקת בטווחים האם מדובר במספרים רגילים. שיטה זו מקבלת את המקרה שמדובר בטווח מספרים רגיל. שיטה זו תחזיר ערך בוליאני FALSE כאשר אחד מצדדיו של הטווח הוא איננו מספר, אחרת יוחזר TRUE.

def number\_parser\_percentage(self, tokens, token, index):

שיטה זו מטפלת במקרה שמדובר באחוזים. שיטה זו מקבלת את רשימת הביטויים, הביטוי הנוכחי ומיקומו. השיטה מקדמת את מיקומו של האינדקס הנוכחי ברשימה ומוסיפה למספר את הסימן "%" אשר מסמל לנו אחוז. ונחזיר את הביטוי העדכני ונקדם את האינדקס.

def number\_parser\_date(self, tokens, token, index):

שיטה זו מטפלת במקרה שמדובר בתאריכים. שיטה זו מקבלת את רשימת הביטויים, הביטוי הנוכחי ומיקומו. השיטה בודקת את המקרים השונים אשר הוגדרו כתאריכים, באם מופיע חודש או שנה אחרי או לפני הביטוי עבורו הגענו לשיטה זו. לאחר מכן נטפל בסדר הכנסת הביטויים כפי שהוגדר. נחזיר את הביטוי העדכני ונקדם את האינדקס.

def number\_parser\_price(self, tokens, token, index):

שיטה זו מטפלת במקרה שמדובר במחירים. שיטה זו מקבלת את רשימת הביטויים, הביטוי הנוכחי ומיקומו. השיטה בודקת את המקרים השונים אשר הוגדרו כמחירים, באם מופיע מיליון, מיליארד, דולר אמריקאי וכן הלאה אחרי הביטוי עבורו הגענו לשיטה זו. לאחר מכן נטפל בסדר הכנסת הביטויים כפי שהוגדר. נחזיר את הביטוי העדכני ונקדם את האינדקס.

def number\_parser\_regular(self, tokens, token, index):

שיטה זו מטפלת במקרה שמדובר במספרים רגילים. שיטה זו מקבלת את רשימת הביטויים, הביטוי הנוכחי ומיקומו. שיטה זו מטפלת גם במקרים בהם תופיע מילת ייצוג של אלף, מיליון, ביליון או טריליון נוסיף למספר את המילה הרלוונטית ונחזיר את הביטוי העדכני ונקדם את האינדקס.

@staticmethod  
def change\_month(month):

שיטה זו הינה שיטה סטטית אשר מכילה את רשימת החודשים. שיטה זו מקבלת כמחרוזת את החודש ותחזיר את מספרו של החודש הנ"ל כמחרוזת.

@staticmethod  
def term\_number(term):

שיטה זו אחראית על הוספת התו הרלוונטי עבור מספרים רגילים מתוך השיטה number\_parser\_regular. שיטה זו מקבלת מספר ומשרשרת אליו את התו המתאים לו לפי תיאור גודלו. נחזיר את הביטוי החדש עם התו הרלוונטי.

def money\_parser(self, tokens, token, index):

שיטה זו מטפלת במקרה שמדובר במחירים עבורם התו הראשון הינו "$". שיטה זו מקבלת את רשימת הביטויים, הביטוי הנוכחי ומיקומו. השיטה בודקת את המקרים השונים אשר הוגדרו כמחירים, באם מופיע במיליון, ביליון, דולר אמריקאי וכן הלאה אחרי הביטוי עבורו הגענו לשיטה זו. לאחר מכן נטפל בסדר הכנסת הביטויים כפי שהוגדר נוריד את סימן ה-"$". נחזיר את הביטוי העדכני ונקדם את האינדקס.

@staticmethod  
def clean\_number\_name(token):

שיטה זו מקבלת ביטוי מורידה את התו האחרון שאינו רלוונטי בביטוי, ומחזירה ביטוי זה.

@staticmethod  
def percent\_parser(token, index):

שיטה זו מטפלת במקרה שמדובר באחוזים שמתחיל בסימן ה- "%". שיטה זו מקבלת את הביטוי הנוכחי ומיקומו. השיטה מורידה את התו הראשון ומוסיפה בסוף הביטוי את תו זה ובכך מייצגת את האחוז כפי שהוגדר. נחזיר את הביטוי העדכני ונקדם את האינדקס.

@staticmethod  
def term\_word(tokens, token, index):

שיטה זו מטפלת במקרים של מספרים. שיטה זו מקבלת את רשימת הביטויים, הביטוי הנוכחי ומיקומו. השיטה מוסיפה תו רלוונטי על פי ייצוג המספר ("K","M","B"). ונחזיר את הביטוי העדכני.

def word\_parser(self, tokens, token, index):

שיטה זו מטפלת במקרה שמדובר במילים. שיטה זו מקבלת את רשימת הביטויים, הביטוי הנוכחי ומיקומו. השיטה בודקת האם מדובר בטווחים, בביטויים של מילים או מילה רגילה ומטפלת בכל מקרה בהתאם. נחזיר רשימת מילים עדכנית ונקדם את האינדקס.

def save\_doc\_data(self, doc\_name, doc\_terms, doc\_properties, max\_tf, max\_term):

שיטה זו מטפלת בשמירת המידע עבור הביטויים השונים שבמסמך. שיטה זו מקבלת את שם המסמך, מילון הביטויים, מאפייני המסמך, כמות הפעמים המקסימלית של הביטוי הנפוץ ביותר והביטוי הנפוץ במילון הביטויים. הוספת הביטויים מתחשבת בביטויים שכבר בתוך מילון, ועבורם נאחד את המידע הרלוונטי למסמך זה ובכך נשמור עבור כל קבוצה של 50 קבצים את מילון הביטויים. מילון הביטויים יכיל עבור כל ביטוי את המסמכים בהם הוא מופיע ואף את מיקומיו של הביטוי בכל מסמך. שמירת המידע יבוצע בזיכרון, במילון הביטויים.

def save\_parser\_data(self):

שיטה זו מבצעת את שמירת הנתונים בזיכרון עבור קבוצת הקבצים שעברו פירסור. בשיטה זו נעבור על כלל הביטויים ושמות המסמכים, שקיבלנו בקבוצת המסמכים, באופן ממוין ונבצע עבורם כתיבה לזיכרון. עבור כל קבוצת 50 קבצים נשמור שני קבצים: האחד הינו קובץ מידע עבור הביטויים והשני הוא קובץ עבור המסמכים שבקבוצת המסמכים. עבור קובץ הביטויים נשמור את הביטוי כמות המסמכים בהם הוא מופיע ואת מילון במיקומים (מסמך- מיקומים). עבור קובץ המסמכים נשמור את שם המסמך, הביטוי שמופיע הכי הרבה פעמים במסמך זה, כמות מופעיו, מספר הביטויים שבמסמך ואת בעיר המייצגת עבור מסמך זה(אם יש כזו). בנוסף הכתיבה לקבצים מתבצעת באופן ממוין.

* **מחלקת Index:**

המחלקה מקבלת רשימת ביטויים והיא עוברת על כל קבצי הפוסטינג של הביטויים ומאחדת ביניהם ל inverted index.

def merge\_two\_posting\_files(self, file\_1\_path, file\_2\_path, index):

שיטה זו אחראית על מיזוג בין שני קבצי פוסטינג שונים. שיטה זו מקבלת שני קבצים ואינדקס אשר מסמן את מספר המיזוג. שיטה זו עוברת על שני הקבצים שורה אחר שורה וממיינת אותם לתוך קובץ חדש. הקבצים שהשיטה מקבלת הינם ממוינים ולכן כשנעבור עליהם שורה אחר שורה נקבל קובץ ממוזג שממוין. לבסוף שיטה זו מחזירה את הקובץ הממוין וכותבת אותו בזיכרון.

def ser\_stemming\_boll(self, bool):

שיטה זו מעדכנת את הערך של ביצוע ה-stemming.

def build\_index\_dictionary(self, save\_path, final\_merged\_posting\_file\_path):

שיטה זו מקבלת את הקובץ הממוזג האחרון שנוצר ויוצרת ממנו את קובץ הפוסטינג. שיטה זו עוברת על קובץ המיזוג הסופי, עוברת על שורותיו ומוסיפה שורה אחר שורה אל קובץ האינדקס של כלל הביטויים במאגר. לבסוף נכתוב את קובץ האינדקס בזיכרון.

* **מחלקת GUI:**

מחלקה זו אחראית על ממשק המשתמש.

def \_\_init\_\_(self, root):

שיטה זו אחרית על אתחול חלון ממשק המשתמש. שיטה זו מגדירה את הפקדים שבממשק המשתמש ואת השיטות השונות עבור כל פקד.

def start\_work(self):

שיטה זו אחראית על הפעלת התוכנית. השיטה

def browse\_save\_file(self):

שיטה זו אחראית על בחירת נתיב לשמירת התיקייה.

def browse\_folder(self):

שיטה זו אחראית על בחירת נתיב לבחירת מאגר הנתונים.

def reset(self):

שיטה זו אחראית על איפוס המערכת ומחיקת המידע מהזיכרון ומנתיב שמירת הנתונים.

def show\_dictionary(self):

שיטה זו אחראית על הצגת המילון שנוצר.

def load\_dictionary(self):

שיטה זו אחראית על טעינת המילון לזיכרון.

**תהליכים בפרויקט**

* התמודדנו עם מגבלת הזיכרון של המחשב בכך שחילקנו את קבצי ה-Posting ל37 קבצים שונים. כלומר עבור כל 50 קבצים ייצרנו קובץ Posting. בנוסף, פעולה זו הקטינה משמעותית את זמן הגישה אל זיכרון המחשב ובכך הקטינה את זמן הריצה של התוכנית. יתר על כן השתמשנו בתור עבור מיזוג הקבצים וזאת כדי שכל קבצי הפוסטינג יתמזגו בצמדים כ-bubble sort ויתקצר זמן הריצה. על מנת לחסוך במקום לאחר מיזוג כלל הקבצים, מחקנו את 37 הקבצים הזמניים. בנוסף לכך השתמשנו בחלק של ניתוח הביטויים במילונים וזאת על מנת לאפשר גישה ישירה לתכולת המילון ובכך לקצר את זמן הריצה לעומת שימוש ברשימה.
* שמרנו את קבצי ה-Posting כקבציי טקסט רגילים. עבור כל 50 קבצים שמרנו קובץ Posting וזאת על מנת לקצר מזמני הגישה אל הדיסק. שמרנו קבצים אלה כקבציי טקסט בכדי שהגישה אליהם תהיי פשוטה ויעילה יותר. כל קובץ מכיל את ה-terms השונים בכל 50 הקבצים ועבור כל term את כמות מופעיו (tf) ומקומו בכל טקסט. סה"כ שמרנו 37 קבצים אשר מביאים מענה ל1,815 קבציי המאגר. עבור קבצי הפוסטינג של המסמכים שמרנו עבור כל מסמך מתוך 50 הקבצים את הביטוי בעל ה-tf המקסימלי, כמות מופעיו של ביטוי זה, כמות הביטויים השונים במסמך ואת העיר המייצגת עבורו.
* בחרנו במספר קבצים אלה מכיוון שרצינו לבחור בכמות מספקת של קבצים שתאפשר למחשב לעבוד בצורה יעילה ומהירה, לא תכביד על המחשב במספר רב של קבצים ובנוסף שייווצרו מספר הגיוני של קבצי Posting ללא גישות רבות לדיסק.
* פרטי האינפורמציה ששמרנו הם: עבור כל מסמך ברשימת המסמכים שמרנו את הביטוי שמופיע המספר הרב ביותר של פעמים. עבור כל רשימת ביטויים במסמך שמרנו את מיקומו הספציפי של כל ביטוי במסמכים השונים בהם הוא מופיע ואת מספריהם של מסמכים אלה.
* בחרנו להוסיף שני חוקים אשר יקלו על תהליך ה-Parser:
  + החוק הראשון מכיל בתוכו שלושה חוקים והוא בהתייחסות לטווחים של תאריכים, מחירים ואחוזים כלומר אם ישנו מצב שבו נקבל טווח ולאחר מכן מילה המסמנת תאריך, מחיר או אחוז נכניס לרשימת הביטויים את כלל האופציות באופן שבו הוגדרו בעבודה.(לדוגמא: percent 4-6 נכניס למילון את הביטויים הבאים 4%,6%,4,6,4-6 כנ"ל לגבי תאריכים ומחירים).
  + החוק השני הוא שעבור מספרים המייצגים מספרים רגילים סימן המספר (K,M,B) מתווסף למספר עצמו כחלק מהמספר ואילו במספרים המייצגים מחירים סימן המספר הנאמר לעיל מתווסף כ-term נפרד.  
    דוגמאות: 50-50 percent (קובץ FB396002 מסמך FBIS3-52) וישנן עוד דוגמאות עבור המצבים השונים. million dollars $430 M Dollar)430) אל מול million 430 (M430) (קובץ FB396002 מסמך FBIS3-58). במחלקת הread file הוספנו את קריאת ה-api והמידע על השפות והערים טרם ביצוע המעבר על הקבצים.
* מהבנת העבודה מחלקת הפרסר הינה המחלקה המשמעותית ביותר על טיבו של המנוע וזאת מכיוון שבניית המילונים ומיונם, מאגר הקבצים והשאילתה עצמה עוברים את תהליך הפרסור ובכך נוצר אחידות בעיבוד המידע, דבר שיקל על אחזור המסמכים עבור השאילתות השונות.
* מיזוג הקבצים במחלקת האינדקס מבוצעת בעזרת תור וזאת על מנת שנעבור על כלל קבצי הפרסר בצורה מסודרת, דבר שהקל על ביצוע מיזוג הקבצים והבקרה על כך.
* השתמשנו בקוד פתוח במחלקות קריאת הקבצים, הפרסר. במחלקת קריאת הקבצים השתמשנו בפלטפורמת JSON אשר הקלה על קריאת המידע מהרשת. במחלקת הפרסר השתמשנו בStemmer- על מנת לנתח את שורשיי הביטויים השונים. במחלקת הgui- השתמשנו בספרית tkinterובתיקיות השונות שבספריה זו.

URL-API: <https://restcountries.eu/rest/v2/all>

* ביצענו את העבודה כמעין שיטת רכבת, כלומר כל תהליך שנוצר לוקח 50 קבצים מעביר אותם דרך ה-Parser לאחר מכן דרך ה-Index רושם את קבצי ה-Posting בדיסק וממשיך הלאה ל-50 הקבצים הבאים, לאחר סיום תהליך זה עבור כל קבצי המאגר מתבצע תהליך המיזוג. המיזוג מתבצע בצורה מבוקרת כך שבכל פעם נוציא שני קבצים מהתור נמזגם ונחזיר את הקובץ הממוזג החדש אל תוך התור ונבצע זאת עד שיישאר לנו קובץ אחד עבורו נבצע אינדוקס.

**חלק ב'**

1. כמות ה-terms השונים ללא stemming: 1,175,863
2. כמות ה-terms השונים עם stemming: 1,066,638
3. כמות ה-terms השונים שהם מספרים (כלומר terms אשר התו הראשון שלהם הוא ספרה): 295,463
4. כמות המדינות השונות במאגר: 126
5. כמות הערים השונות שיש במאגר: 472, מתוכן 126 ערי בירה ו-346 ערים אשר אינן ערי בירה
6. מספר מופעים רב ביותר של עיר בודדת:

שם המסמך: FBIS3-60342  
שם העיר: Moscow  
מספר מופעים: 295  
אינדקסים של ה-term של העיר במסמך:

16, 36, 62, 85, 148, 452, 472, 500, 525, 526, 595, 755, 2000, 2459, 3530, 4022, 4408, 4537, 4564, 4646, 4808, 5003, 5007, 5599, 5603, 5794, 5822, 5854, 5859, 5994, 6042, 6079, 6120, 6134, 6180, 6582, 6603, 6806, 6877, 6884, 6889, 7041, 7098, 7394, 7421, 7498, 7686, 7763, 8221, 8400, 8415, 8464, 8562, 8653, 8854, 8953, 9079, 9080, 9103, 9201, 10080, 10540, 10809, 10815, 10818, 10824, 10826, 10834, 10842, 10850, 10857, 10861, 10862, 10868, 10875, 10879, 10892, 11201, 11289, 11551, 11569, 11748, 11777, 11922, 12050, 12161, 12177, 12388, 12403, 12504, 12519, 12523, 12650, 12674, 12822, 12944, 12951, 13114, 13132, 13151, 13192, 13197, 13202, 13210, 13215, 13219, 13234, 13238, 13252, 13256, 13721, 13897, 14052, 18416, 18994, 19427, 21037, 24036, 27807, 28260, 28498, 28652, 28796, 28888, 29093, 29122, 29160, 29390, 29571, 29761, 29822, 30124, 30133, 30139, 30177, 30199, 30215, 30228, 30260, 30293, 30331, 30346, 30354, 30359, 30425, 30656, 30659, 30709, 30817, 30848, 30849, 30852, 30864, 30895, 30950, 31012, 31029, 31070, 31078, 31084, 31106, 31113, 31132, 31244, 31317, 31429, 31446, 31469, 31488, 31499, 31500, 31514, 31524, 31728, 31830, 31833, 31876, 31904, 31914, 31923, 31935, 31941, 31959, 31962, 31992, 31997, 32002, 32065, 32081, 32097, 32111, 32126, 32177, 32265, 32286, 32319, 32334, 32367, 32460, 32517, 32524, 32651, 32708, 32736, 32752, 32788, 32796, 32848, 32910, 32916, 32960, 33162, 33226, 33228, 33359, 33432, 33503, 33555, 33573, 33693, 33711, 33743, 33758, 33772, 33863, 33871, 33918, 33998, 34037, 34074, 34106, 34165, 36082, 38419, 38802, 42288, 43199, 43812, 46196, 46238, 46251, 46278, 46336, 46376, 46499, 46561, 46612, 46652, 46729, 46900, 46938, 46960, 47040, 47094, 47159, 47215, 47299, 47415, 47489, 47770, 48019, 53176, 53664, 61269, 62340, 68781, 69224, 69781, 69821, 70116, 70316, 70326, 70329, 70335, 70337, 70340, 70343, 70349, 70353, 70533, 70685, 70842, 71015, 71375, 71720, 71742, 72128, 72270, 72316, 72419, 72559, 72950, 73177, 73374, 73462

1. טבלאות terms:

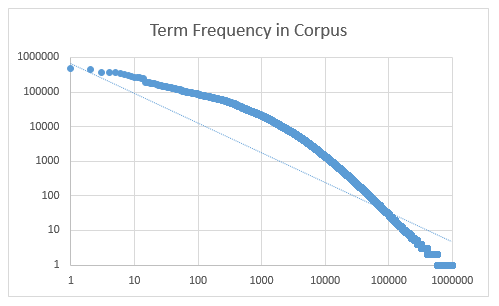
עשרת ה-terms השכיחים במאגר לפי הסדר (1-10) הם:

|  |  |
| --- | --- |
| Term | TF in corpus |
| mr | 470131 |
| year | 440196 |
| dollars | 374442 |
| cent | 364489 |
| said | 335393 |
| government | 320890 |
| pounds | 310397 |
| years | 277350 |
| people | 262530 |
| time | 262261 |

עשרת ה-terms הנדירים במאגר לפי הסדר (1-10) הם:

|  |  |
| --- | --- |
| Term | TF in corpus |
| greenwings | 1 |
| grided | 1 |
| jeans-cut | 1 |
| JECK | 1 |
| JECKLIN | 1 |
| ishai | 1 |
| live-artist | 1 |
| "31-03" | 1 |
| mar-may | 1 |
| naivest | 1 |

1. גרף zipf:



1. רשימת ה-terms של מסמך מספר FBIS3-3366 ממוינת היא:

|  |  |
| --- | --- |
| Term | TF in doc |
| 03-19 | 2 |
| 1994-03 | 1 |
| adopted | 2 |
| amended | 3 |
| ARTICLE | 1 |
| BEIJING | 1 |
| CHARTER | 3 |
| CHINESE | 5 |
| COMMITTEE | 5 |
| CONFERENCE | 4 |
| CONSULTATIVE | 4 |
| CPPCC | 4 |
| decided | 1 |
| effect | 1 |
| EIGHTH | 3 |
| LANGUAGE | 1 |
| NATIONAL | 4 |
| P105 | 1 |
| PEOPLES | 4 |
| POLITICAL | 4 |
| proposed | 1 |
| RESOLUTION | 1 |
| SESSION | 3 |
| STANDING | 1 |
| TEXT | 1 |
| today | 1 |
| TYPEBFN | 1 |
| XINHUA | 1 |

1. נפח האחסון הנדרש לקבצי הPosting- ללא stemming הוא KB1,247,678