פרויקט באחזור מידע

*Firdos Bobo, Hadil Bdir.*

הפרויקט עוסק באחזור מידע, בהינתן שאילתא x כלשהי (שאלה), צריך לתת המידע הכי רלוונטי מבין כל הדאטה, בפרויקט מתעסקים בניתוח שאלות וחילוץ מידע רלוונטי.

הבעיה:

נתון: yahoo dataset, ושאילתה.

פלט: אחת התשובות הנכונות.

*השלבים החשובים כדי לייצר מערכת QA:*

 Create documents.

 data analysis (tokenization, normalization, indexing)

 Question Analysis.

 Passage Retrieval (extract answer candidates, Search relevant documents).

 Improve  Passage Retrieval.

 Return documents.

יצירת מסמכים - Create documents:

כדי לאפשר ל lucene לחפש מסמכים רלוונטים, צריך ליצור לכל טקסט מסמך (document ) ושדה (named field). מחלקת המסמכים מייצגת מסמך ב- Lucene. המסמכים הם היחידה של אינדקס וחיפוש. מסמך הוא אוסף של שדות. לכל שדה יש ​​שם וערך טקסטואלי.

בפרויקט יצרנו document חדש עבור כל תשובה ב dataset. יצירת מסמכים בצורה זו מאפשרת חיפוש וindexing יעילים, באמצעות Lucene אפשר לייצר אינדקס בשיטת createIndex(), ואובייקט IndexWriter() שמשמש לייצר ולעדכן את האינדקס.

ניתוח נתונים - data analysis:

ניתוח נתונים הוא תהליך של בדיקה, טיהור ושינוי נתונים במטרה לגלות מידע שימושי, לחלץ מסקנות ולתמוך בקבלת החלטות. לניתוח נתונים יש היבטים וגישות מרובים וטכניקות מגוונות.

השיטות שהשתמשנו בהן על מנת לאפשר ***indexing***:

 Tokenization.

 Normalization.

 Stop words

Tokenization

היא המשימה לקצץ רצף של מילים לחתיכות, הנקראות tokens, לזרוק תווים מסוימים, כגון סימני פיסוק.

מנתח Analyzer בונה TokenStreams, ומייצג מדיניות לחילוץ מונחי אינדקס מהטקסט. המנתח משליך טקסט שאינו שימושי עבור יישום חיפוש.

Stop Words

מילים נפוצים בשפה שלא מסייעים בבחירת מסמכים רלוונטים לשאילתה, התאמת הצורך של המשתמש למסמכים אינם נכללים באוצר המילים האלה לחלוטין.

לצורך משימה זו השתמשנו ב מנתח (analyzer) סטנדרטי של Lucene , שתומך ברשימת stop words מוגבלת, והוספנו רשימת stop words חדשה שמסייעת למחיקת כל המילים מצורף קובץ של רשימה זו בשם stopWords.txt.

Tokens normalization

שבירת טקסט לtokens היא רק חצי מהעבודה. כדי להפוך את ה tokens הללו לקלים יותר לחיפוש, נעבור תהליך נורמליזציה כדי להסיר הבדלים משמעותיים בין מילים זהות זה לזה, כגון אותיות גדולות לעומת אותיות קטנות.

Stemming

הינו התהליך של העברת מילים לצורת הבסיס שלהם, לדוגמא:

.car, cars, car's, cars' → car

.the boy's cars are different colors → the boy car are different color

בכדי לבצע lemmatization אנו צריכים מנגנון שמכיר את השפה שלנו. Stemming גם הוא התהליך של העברת(terms) מושגים/תארים לצורת הבסיס. הפעולה נעשית ע"י הסרה של תחליות או סיומות מה-terms שלנו. לדוגמא: automate(s), automatic, automation → automat.

לצורך משימה זו השתמשנו ב EnglishStemmer.

פעולת ה-stemming אינה, בהכרח, מייצרת לנו מילים "חוקיות", ולכן משתמשים במילון השפה.

Dictionary

משום שה dataset היא non-factoid, ומהעובדה שיש בה מילים לא נמצאים בשפה (אין להן משמעות ואינם קשורות לאף נושא), וכתוצאה מהפעלת תהליך ה stemming, השתמשנו במילון השפה האנגלית כדי לוודא שמילה היא כן נמצאת לפני ה indexing, דבר שעלול לפגוע במילים שמאויתות שגוי למשל (problem, prublem ) ולכן משתמשים ב spelling corrector.

Spelling correction

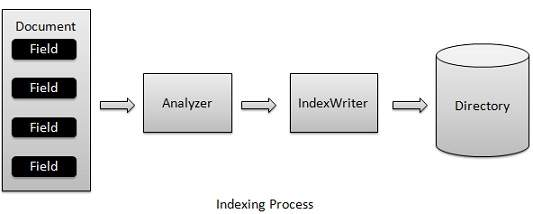
מילים ב DATA שלא נמצאים במילון : או שהם מילים ללא משמעות (למשל eurhfn), או שהם מילים עם שגיאות כתיב.

תיקון שגיאות כתיב של מילה בודדת. במקרים כאלו, לא נזהה שגיאות אם האיות של המילה תקין (from במקום form)

Indexing

תהליך האינדקס הוא אחד הפונקציונליות הליבה שסופק על ידי Lucene’ רכיב שמפרק את תוכני המסמכים ויוצר מהם אינדקס לפי המילים.

התרשים הבא ממחיש את תהליך יצירת האינדקס ושימוש במחלקות:



ה index מאחסן נתונים סטטיסטיים על מונחים כדי להפוך את החיפוש based-term ליעיל יותר.

ה index של Lucene נחשב כ inverted index. זאת משום שהוא יכול לשמור, עבור term, את המסמכים המכילים אותו. זהו ההופך של הקשר הטבעי.

Inverted index שהשתמשנו בו בפרויקט הוא אינדקס מסוג *word-level inverted index* שמאחסן לכל term רשימת מיפוי למסמכים שמכילים אותו ומקומו של כל term במסמך.(posting list) .

Question Analysis

כדי לייצר מערכת אחזור מידע נכונה צריך לנתח השאילתה באותם כלים של המידע, ולכן משתמשים בשיטות קודמות של data processing עבור כל קלט (שאלה) שמקבלים.

Passage Retrieval

כדי לשחזר מסמכים רלוונטים עבור כל שאלה שמקבלים, צריך להשתמש במודל של  similarity שקובע עבור כל מסמך מידת ההתאמה שלו בהינתן השאילתה הנתונה.

ראשית השתמשנו בשיטת Term Frequency שמדרגת כל מסמך (תשובה).

המשקל של המסמך נקבע על ידי התדירות של ה terms ביחס לשאילתא (השאלה), ככל שיש יותר terms שנמצאים גם בשאלה וגם בתשובה המשקל גדל.

בשיטה זו כל ה terms מקבלים חשיבות זהה. אבל קיימים terms מסוימים שמתרחשים בתדירות גבוהה שהם לא רלוונטיים לנושא השאלה  .ודווקא ל terms אלה אנו צריכים להקטין את המשקל שלהם, ולתת חשיבות יותר ל terms שמופעים בתדירות יותר נמוכה אבל יותר קשורים וחשובים לרלוונטיות של השאלה. ולכן משתמשים ב IDF.

לפי התוצאות, קבלנו ש BM25 היא ה similarity הכי טובה למערכת שלנו,

https://lh3.googleusercontent.com/ES6Lgho7CpvyQOKIjfczr6UKwoI1qyiY87eRujw0JJFQRsP1mMm0P-9YVcWVabL21P7mU-j-wgBhsZOrD1p-661SUfE_IKg-sV2lj4hl2MtY2kjQVPTVVrIvx3YA1qgRCBumhcdQ

where *ni* is the frequency of term *i* in document *d*.

Typical values were chosen for the constants *k1* and *b* (2.0 and 0.75, respectively).

Document length *|D|* was estimated by adding the term frequencies *ni* per document.

Average document length https://lh3.googleusercontent.com/VXcY5IuqLORF8RMYuemj_u_68lbqa97R1ggSY4YumigPqK9T2KDpIEXi2aQkmSesB0xHavuHYPoS1IaT6CVY1SNJ0RS_5H2ON5BERA65Kl0zuVX4HN_6axvEMf-H1bO3Wgro-rcx is computed over the entire document set.

The IDF value for a particular term *i* was computed as:

https://lh6.googleusercontent.com/xTDhRLEKHu5nL6JTntI0fL3W_9A7RH-R0OR8oU3jFB6C3clledeDuMKKl__c2C6GRxE2Asnr1yeM_h6Ju-Lv6xs4hiJoDMWKcfAhbGPaAZjWWJE9pD-9bLkDf5GqgbRdtuMAp52_

here *N* is the total number of documents in the dataset

*di* is the number of documents containing term *i*.

פונקצית search ב Lucene  שמחפשת מסמכים רלוונטים בהתאם ל similarity שקבענו(BM25), מחזירה מערך של topDocs שמכיל המסמכים בעלי ה score הכי גבוהה לפי השאלה.

אחרי שקיבלנו עבור כל תשובה משקל לפי שיטת BM25 מבצעים similarity נוספת.

Improve Passage Retrieval

כדי לשפר את הביצועים שלנו ולקבל תשובות יותר מדויקות השתמשנו בשיטות הבאות :

 דמיון של wordNet::Similarity, שמודד את הדמיון הסמנטי ואת הקשר בין זוג מושגים. הוא מספק מדדים שמבוססים על מסד הנתונים WordNet. אשר לוקחים שני מושגים כקלט, ומחזירים ערך מספרי המייצג את המידה שבה הם דומים או קשורים.

 פונקציה כזו לא מזהה קיצורים (למשל PC שהיא קיצור של Personal Computer ) היא נותנת score אפס עם כל המילים אפילו אלה הקשורים בה, דבר שפוגע ברליוונטית כי יתכן שמילה זו היא הנושא של התשובה (המילה שמדבר עליה המשפט!)

לכן מילים שתמיד מחזירים similarity שווה אפס, נבדקים על ידי אלגוריתם אחר ומוחלפים ב פירושם, מסמך בשם AbbreviationsWords.txt מכיל קיצורים נפוצים בשפה.

חישוב score:

מחשבים את ה score מחדש עבור התשובות שהתקבלו בשלב הראשון, כך ש: מילים שקבלו similarity>0.8, מוכפלים פי 2 (כנראה שאלה מילים חשובות לשאלה ויש בינים לבין השאלה קשר חזק), בסוף מנרמילים את ה score.

**שיטות נוספות למערכת QA: להשתמש ב machine learning**

לסווג הקלט (השאלה) לאחת הקטגוריות ולחפש רק בתשובות ששיכות לאותה כטיגוריה, במקרה שהסווג הראשוני לא נכון, כל התשובות וכל החיפושים יתנו תשובות לא נכונות (אפילו לא מאותה קטיגוריה) .

*דרכים לממש שיטה כזו:*

א) machine learning: לבצע אימון (training) על שאלות הדאטא, ללמוד מהמודל מאפינים מיוחדים עבור כל כטגוריה בהתאם לשאלות (למשל מילים חוזרים בשאלות של הקטגוריה x).

בהינתן שאלה כלשהי לסווג אותה לקטגוריה נכונה לפי ה training.

**References:**

**https://etd.ohiolink.edu/rws\_etd/document/get/akron1253659613/inline**