דו"ח חלק א'

1. עיצוב התוכנה:
2. תחילה, אנו מבצעים קריאה של המידע אותו אנו רוצים לעבד. בצורה איטרטיבית אנו עוברים כל פעם על קובץ המכיל מידע על טוויטים.

לאחר מכן מפרסרים את המידע על הטוויט ומסווגים אותו לשדות שונים כאשר ישנם שדות לURL, לTWEETID לFULL TEXT ועוד מידע נחוץ שיכול לעזור לנו בהמשך. במהלך הפירסור על הטוויט, אנו מסווגים כל מילה לפי הדרישות ובנוסף מורידים ממנו מידע אשר בעניינו נחשב כלא נחוץ (לדוגמה סמיילים, טקסט בערבית וסינית וכו').

לאחר שאנו סיימנו לפרסר את כל הטוויט, אנו מעבירים את הDOCUMENT שיצרנו לפי נושאים לשלב האינדוקס, בו אנו מחלקים את המידע (מילים שמצאנו לנכון והעברנו מהפרסר לDOCUMENT) בצורה שתהיה לנו נוחה לאחזר את המידע כאשר נדרש לעשות זאת. אנו שומרים עבור כל TERM בכל DOCUMENT בקובץ חיצוני לפי האות הראשונה (כאשר כל 25 אלף טוויטים, כך מצאנו לנכון בגלל אילוצי זמן ומקום לאחר), את כמות הפעמים שהיא מופיעה בקורפוס, ובאילו טוויטים היא הופיע.

לאחר שביצענו איטרטיבי את פעולות הקריאה-פרסור-אידוקס את כל הקורפוס, אנו מעבדים את המידע, ועושים איחוד למסמכים על מנת להקל על עצמינו בשלב החיפוש והדירוג, כך שהמידע יהיה נגיש בצורה נוחה יותר.

לאחר מכן, אנו בונים את המודל LDA שלנו, בעזרת כל הנתונים שקיימים, כך שאנו מעלים לפני כן, את כל המידע עבור המסמכים שנמצאים בזיכרון החיצוני, וכך לא תופסים מקום על הRAM.

כעת, אחרי שהמודל עלה, אנו מבצעים בדיקה ושמירה במבנה נתונים עבור כל מסמך את הנושא שהכי דומה לו ואת ההסתברות שלו, כך שבמידה ושאילתה מתאימה לנושא מסוים אזי בשלב הSEARCHER אנו מחזירים את כל המסמכים הקשורים לאותו נושא.

לבסוף, אנו מבצעים דירוג לפי COS SIMILARITY שנלמד בהרצאה ובתרגול וכך אנו מדרגים את הטוויטים.

1. למגבלת הזיכרון היינו צריכים לייעל כל אחד מהבאים:

1. תחילה בשלב הREADER ביצענו קריאה ושמירה של כל המידע (טוויטים) של כל הקורפוס של קבצי הפרקט במערך (LIST) אחד אשר מכיל את כל ה10 מיליון טוויטים, דבר אשר גרם בהמשך העבודה ככל שהתקדמנו לבעיית זיכרון ואיטיות שנבעה מכך, לכן החלטנו לעבור בלולאה ולעשות באופן איטרטיבי כך שקודם מכניסים למערך אחד את כל הטוויטים בקובץ פרקט אחד, עושים פרסור על כל המאגר ואינדקוס ואז מוחקים את המערך ומכניסים למערך קובץ טוויטים חדש מקובץ פרקט אחר.

2. בשלב הפרסר, אנו התחלנו לערוך ניסוי מה השיטה הטובה ביותר לרוקן את הטוויט ממידע לא רלוונטי, כגון, סמיילים, צורות למיניהם, טקסט בסינית וכו', וכל פעם מצאנו דרך טובה יותר באמצעות ביטוי רגקס מתאים כך שהמידע יכיל רק את התווים שאנו דורשים שהוא יכיל, דבר אשר מסייע בנוסף לשלב האינדוקס.

3. בשלב האינדוקס שבו החלק המרכזי של הזיכרון, אנו ערכנו כל פעם ניסוי במטרה להבין כיצד המחשב מתמודד בדרך הטובה ביותר עם המידע, כך שחלוקת המידע בין הRAM לבין הזיכרון החיצוני תהיה אופטימלית, הן עבור בעיית זיכרון והן עבור בעיית זמן. כיוון שאנו שומרים מידע רב, נאלצנו לעבוד ביסודיות, כך שכל פעם שמגיע לי TERM מסוים אני מכניס אותו למילון לפי האות הראשונה שלו, וכך אנחנו יודעים כאשר נצברים מאות מסויימת 500 אלף TERMS, אנו מעלים את המידע הזה של אותו אות לזיכרון הראשי ומוחקים את המידע הקיים מהזיכרון RAM, כך אנו דואגים שהכמות זיכרון של הRAM לא תתמלא.

4. בשלב הMERGE, זהו צעד אמנם יקר מבחינת זמן, אך בהמשך יעזור לנו להחזיר בצורה מהירה יותר את הדירוג של הטוויטים. אנו נאלצים לבצע MERGE של כל הקבצים לפי אותיות, כך שבסוף מתקבל קובץ פוסטינג אחד לכל אות.

5. המודל שקיבלנו היה LSI/LDA, כאשר בחרנו במודל LDA, אנו נאלצנו לשמור את המילון של הקורפוס ואת מאגר הTEXT TOKENS של כל DOCUMENT. התמודדנו עם הבעיה בכך שכל 100 אלף (לאחר ניסויים גילינו שזה המספר הטוב ביותר לשמור גודל של מידע כזה בזיכרון הראשי) אנו מעבירים לקובץ פוסטינג את כל המידע הזה ואז מפנים מחדש את הזיכרון במבנה נתונים הזה. לאחר אגירת כל המידע בקבצי פוסטינג אנו מבצעים MERGE שנועד לאחד את כולם לקובץ אחד אשר עוזר לנו להכניס את כל המידע לקורפוס ולמילון של המודל, כך שהוא לא נשמר על הזיכרון הראשי. לאחר יצירת המודל, אנו מתחילים ישר בביצוע ההסתברויות ושמירת עבור כל מסמך את ההסתברות של הנושא הקרוב ביותר, ולאחר מכן שומרים ישר את המודל בקובץ חיצוני על מנת שלא יישאר על הRAM.

1. במחלקת הINDEXER, אנו מבצעים שמירת עבור כל אות, סימנים מיוחדים כאחד, והספרה הראשונה של מספר, שמירת קובץ פוסטינג כאשר כל קובץ מכיל 500 אלף טוויטים. בסוף תהליך האינדוקס אנו מגיעים למצב של בממוצע (משתנה בין אותיות שכיחות לזניח) 20 קבצי פוסטים ומבצעים להם MERGE לקובץ אחד לכל אות. הקובץ במבנה נתונים של מילון, כך שהמפתח זה הTERM והערך זה הTWEETID שבו הוא מופיע. כמו כן, אנו שומרים גם קובץ מילים נפוצות

בנוסף אנו שומרים עבור המודל שקיבלנו (LDA) ועבור הרנקר (שם ביצענו COS SIMILARITY) את כל הטוקנס מכל מסמך כולל הTWEET ID, אנו שומרים כל 100 אלף DOCUMENTS כיוון שהם שוקלים משמעותית יותר משאר מבני הנתונים.

לאחר תהליך הMERGE אנו מקבלים קובץ מספרים אחד, קובץ אחד לכל אות, קובץ DOCUMENTS, וקובץ מילים נפוצות.

1. לאחר ניסויים של ערכים בהם בחרנו כל פעם להעלות את הנתונים לקובץ פוסטינג הגענו להחלטה משותפת בגלל בעיות מקום (אשר משפיע ישירות על הזמן) שכל קובץ לאות אנו מעלים כאשר נאגרים בו 500 טוויטים שמוזכרים בו. עבור המסמכים החלטנו כל 100 אלף כיוון שכל המילים עבור DOCUMENT מסוים גוזלים יותר מקום. כך הגענו למצב שאנו שולטים בצורה מבוקרת על הזמן ועל המקום ככל שניתן.

להוסיף תיעוד!!! (של קבצי הפוסטינג)

1. אנו בחרנו להוסיף לכל TERM וTWEETID המציין באיזה טוויט הוא הופיע בקורפוס את ה-tf ואת ה-idf שלו במסמך הנתון, דבר אשר מאוד מכל עלינו בתהליך הדירוג, כיוון שבשביל לחשב את הCOS SIMILARITY, אנו זקוקים לנתונים אלו.

בנוסף, עבור המודל שלנו שמרנו גם את כל הTERMS שקיבלנו עבור המסמך בתהליך הפרסר, בשביל ליצור את המודל.

בנוסף לצורך מימוש מודל הLDA אנו נאלצנו לשמור 2 מבני נתונים נוספים, אחד עבור כל רשימת הTERMS שקיימים בכל מסמך (כך שבסופו של דבר מדובר ברשימה של רשימות של סטרינגים) ובנוסף שמרנו מילון שהKEY שלו זה המספר הסדר שבו הוא נכנס לרשימה לפי מיקום והVALUE שלו זה הTWEETID (נאלצנו לעשות זאת כיוון שבמודל LDA המסמכים נקבעים לפי סדר הגעתם ועל מנת לשייך בין הDOCUMENT לבין הTWEETID נאלצנו לבצע זאת, וכך יכולנו לדעת איזה DOCUMENT שייך לאיזה TWEETID).

בנוסף, שמרנו גם את המודל שלנו כיוון שכל יצירה שלו גוררת זמן ארוך מאוד (כמה שעות) + המילון של אותו קורפוס של המודל. לאחר יצירת המודל שמרנו גם במבנה נתונים של מילון שהKEY שלו זה המספר נושא (הנושאים במודל ממוספרים מ0 עד המספר נושאים שהגדרנו לו), והVALUE זה הTWEETID, כך דאגנו לשייך כל מסמך לנושא, דבר אשר הועיל לנו רבות לאיחזור המידע.

1. שני החוקים שהוספנו בשלב בפרסר הם:
2. ציטוטים: תהליך המימוש מתבצע כך שבכל פעם שאנו מגיעים בFULL TEXT לביטוי מהצורה גרשיים (לא משנה אם גרש או שניים) + סטרינג בפנים (לא משנה אם מדובר בסטרינג אחד או כמה) ואז שוב פעם גרשיים, אנו שומרים את כל הביטוי (ציטוט) כאחד, בנוסף לשמירת כל TERM בפני עצמו. הבחנו שפעמים לא מעטות ישנם ציטוטים, ולכן החלטנו שלא ניתן להתעלם מהם, ושהם במידה מסויימת בהחלט יכולים לעזור לנו לגבש מידע יותר מדויק עבור הטוויט.

להוסיף דוגמה, איך ב2 טוויטים שונים מגיעים התוצאות

1. הורדנו מידע מיותר: (ניתן לומר שפונקציה זו דאגה לנקות לנו המון המון מידע מיותר ולהשאיר את הTOKENS נקיים מג'יבריש ומידע שלא נחוץ לטעמינו). תהליך המימוש מתבצע כך שביצענו ביטוי רגקס עבור על מילה בטוויט שבו החלטנו בדיוק מה התווים אותם אנו מעוניינים שיכלול (כמו מספרים תווים באותיות גדולות וקטנות, נקודה, סלש, גרשיים, סימן קריאה ושאלה וסימן דולר), כך שאם ישנם תווים/סימנים/שפות זרות וכו' הוא לא יבין בהם (לדוגמה מילה בערבית, כל מיני טעויות כתיב בהוספת סימנים בטעות באמצע מילה, סמיילים, טקסט בסינית וערבית, וסימנים אשר נמצאים בשפות זרות שאינם נמצאים במקלדת שלנו ונדרש להביאם ממקלדת זרה). במצב זה ניקינו המון "מידע מלוכלך" אשר אינו תורם לנו כלל אם היינו משאירים אותו במאגר הTERMS עבור הטוויט, ומנפח את המילון במקום. תהליך זה שיפר לנו גם את זמני הריצה כיוון שניקינו מידע שלטעמינו לא היה קריטי בעת ביצוע שלב האיחזור.

לתקן את סעיף 2 כי אולי לא חכם לרשום שלנקות מידע לא שימושי זה כולל לנקות ערבי וסינית למרות שעשינו את זה כי יש מצב שאם מישהו ירשום נגיד שאילתה בסינית אז זה יעזור לו להגיע לטוויטים המיועדים, צריך לחשוב איך לנסח טוב.

להוסיף דוגמה, איך ב2 טוויטים שונים מגיעים התוצאות

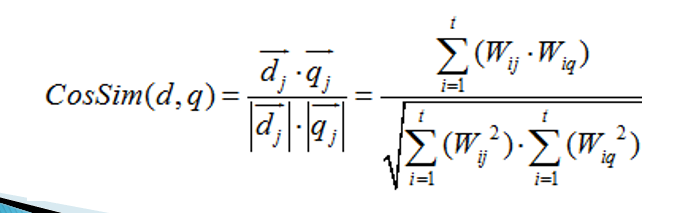
1. כללים וחוקים שהוספנו בשלב הפרסר הם:
2. הוספנו אופציה למקרים של מילה מחוברת לסימן קריאה, סימן שאלה, נקודה או פסיק ואז מיד עוד מילה, וזה יפרש אותן כ2 מילים שונות, כיוון שהבחנו שיש לא מעט מקרים בהם קיים מצב כזה אשר במקום שייכנס כמילה אחת למילון אנו דאגנו לפצל אותם ולדאוג לכך שהמילים יפוצלו כראוי.

כללים וחוקים שהוספנו בשלב האינדקסר הם:

1. דאגנו כבר בשלב האינדוקס לאותיות גדולות וקטנות על ידי כך שכל מילה באינברטד אינדקס שמרנו שדה, בנוסף לכמות המופעים שלה במסמכים, שמייצג האם הTERM הנל הוא אות גדולה או קטנה. כלומר במידה והגיע מילה באות קטנה היא נשמרה בקטן בשדה המיועד, אך במידה והמילה הגיעה בפעם הראשונה באות גדולה שהגיע, שמרנו אותה תחילה כאות גדולה, ובמידה והגענו לאותו מילה שוב אף רק באות קטנה אנו משנים את השדה לאות קטנה, וכך דואגים בשלב האינדוקס לתחזוקה של אותיות גדולות וקטנות.
2. בנוסף, אנו שומרים מונה שכל DOCUMENT שנכנס לאינדקסר אנו מכניסים למילון שהKEY שלו זה המונה, והVALUE שלו זה הTWEETID. לטובת מודל הLDA, כאשר אנו מכניסים למודל רשימה של רשימות (שכל רשימה זה רשימת הTOKENS של אותו מסמך), הדרך לזהות את הקשר בין רשימה ספציפית היא לפי סדר הכניסה שלה לשלב האינדוקס וסדר הכניסה (שזהה) ביצירת המודל.

בנוסף אנו שומרים במבנה נתונים מסוג רשימה של רשימות את כל הTOKENS של כל טוויט, וכך אנו דואגים להעביר את המידע הנל למודל על מנת שייצור את המודל. לאחר שלב האינדוקס והעברת המידע (רשימת רשימות של TOKENS של כל DOCUMENT), אנו מוחקים את הרשימה כיוון שלא נזדקק לה עוד.

1. אנו בחרנו להשתמש בשיטת הדירוג cosSim. תחילה דאגנו לשמור מראש במחלקת האינדקסר את כל המידע הרלוונטי עוברינו, כמו רשימת הTOKENS של כל מסמך ואת הTF שמתאים לאותו TERM, וכמו כן במילון האינברטד אינדקס את הIDF עבור כל TERM שמופיע (כמובן דאגנו לעדכן בכל פעם את הערכים האלו). בנוסף שמרנו באינברטד אינדקס את הIDF של כל TERM.

כעת במחלקת הרנקר אנו עוברים בצורה איטרטיבית עבור כל TERM במסמך ומחשבים עבורו את הWij כלומר (IDF-TF) בהתאם לנוסחה הבאה (מתרגול 3):

את המידע עבור הidf ועבור ה הtf אנו מביאים מקובץ הפוסטינג של האות שבה מתחיל הTERM. את הTF עבור כל טוויט אנו שומרים בשבל האינדוקס, ולאחר שלב האינדוקס בשלב הMERGE FILES אנו מוסיפים לכל TERM את הIDF בעזרת המידע של כמות המסמכים הכוללת בקורפוס.

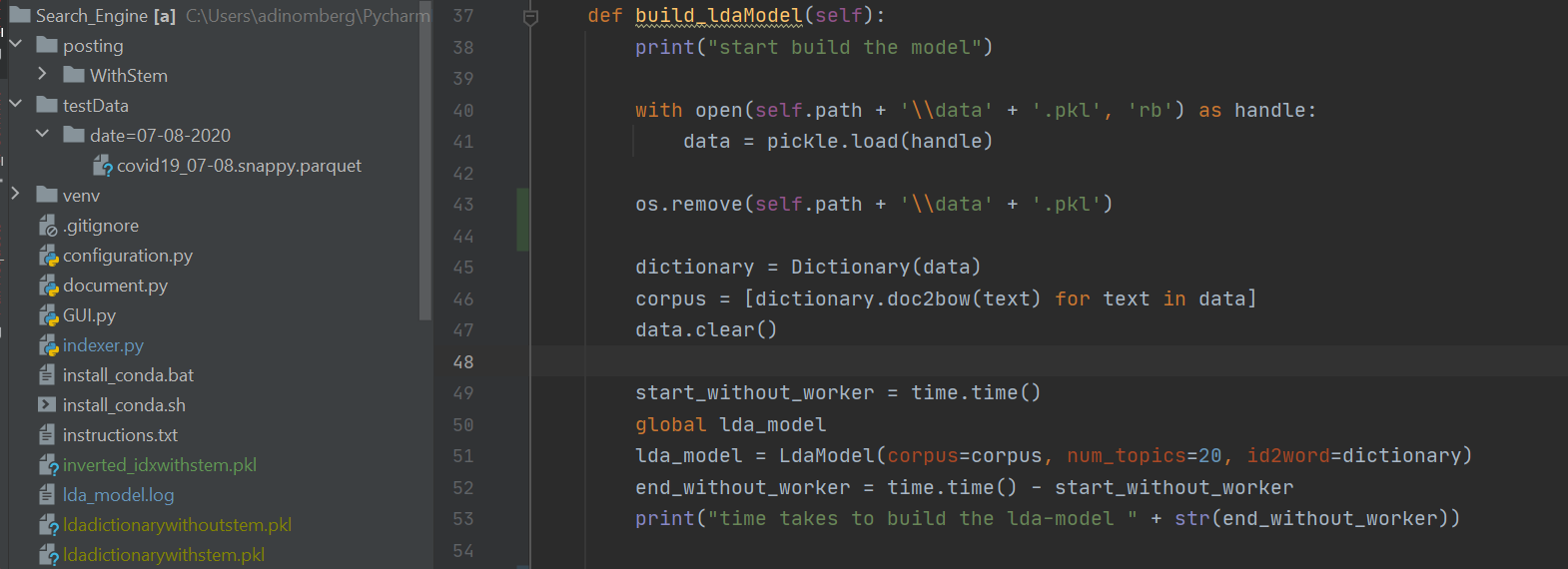
כל קובץ פוסטינג כזה מכיל את הטוויטים שבהם הוא הופיע (TWEETID) ואת הTF שלו באותו מסמך, ובנוסך לזה אנו שומרים בשדה נוסף את הIDF של אותו TERM.

בצורה כזו אנו עוברים על כל המסמכים שהגיעו לשם ומחשבים את הנוסחה עבור על TERM.

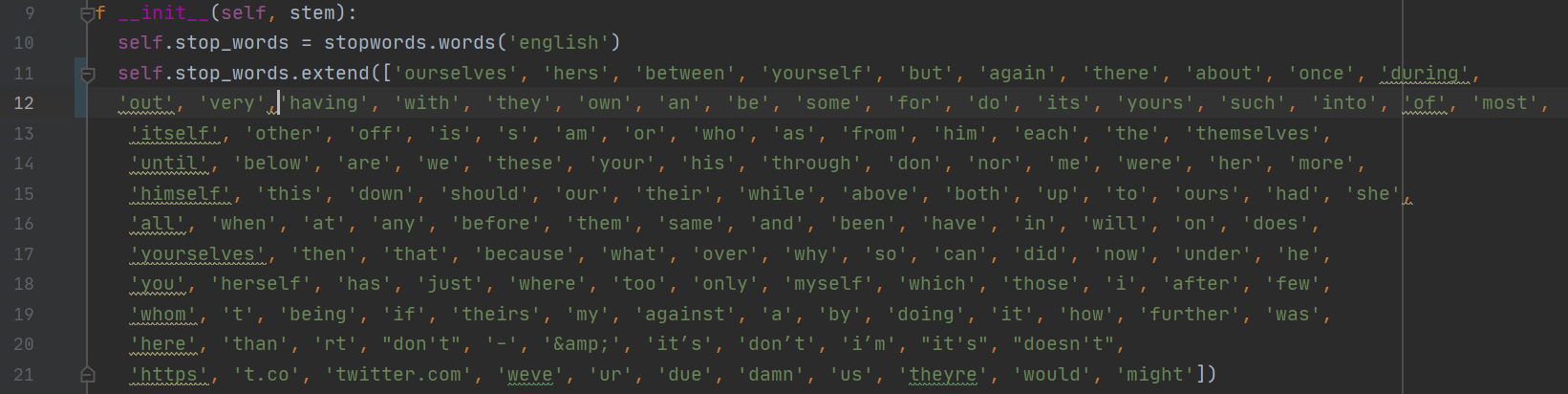
לבסוף, אנו יוצרים מערך שכולל את הTWEETID ואת הSCORE שזהו ציון הCOSSIM של אותו מסמך, ומחזירים רשימה ממוינת מהSCORE הגבוה שיהיה במקום ה0 לSCORE הנמוך, שיהיה אחרון ברשימה (כמובן נחזיר את וארך הרשימה לפי הקלט שנקבל של כמות האיברים במערך אותו נרצה להחזיר ברנקר).

1. החבילה העיקרית שאותה התבקשנו לממש בשביל המודל LDA הוא חבילת GENSIM.

על מנת ליצור את המודל כראוי, תחילה אני צריך להגדיר לו את כל המילים שקיימים במאגר הקורפוס שזהו הDICTIONARY של המודל. במילון של המודל הוא מקבל כביכול את העולם שממנו הוא מפיק את רשימת הנושאים לשנו. כלומר לא ייתכן מילה שלא תופיע במילון והיא תייצג או תהיה חלק (בדרך כלל נושא מורכב מכמה מילים שונות). לכן לפני שאני יוצר את המודל בעזרת חבילת GENSIM, אנו שולחים את כל המילים הייחודיות למאגר הDICTIONARY. כאשר DATA מייצג לי רשימת רשימות, כאשר כל DOCUMENT מייצג לי רשימה שבתוכה נמצאת רשימת הTOKENS של כל מסמך. בתהליך הDICTIONARY החבילה יודעת לזהות מילים שחוזרות על עצמן וכך לא מכניסות אותן למילון אם יש חזרה. בנוסף הDICTIONARY הנל מוכנס לתוך הCORPUS שמייצג את הקורפוס של המודל כך שמיוצג על ידי רשימה של רשימות, כאשר כל רשימה מייצגת מסמך וכל מערך של מסמך מיוצג בצורה הבאה: הופך כל TERM ייחודי במסמך לייצוג של TUPPLE כאשר במקום ה0 זה מציין את מספר הTERM (ייחודי במסמך), ובמקום ה1 מייצג את הTF כלומר כמות הפעמים שהTERM מופיע במסמך. כעת אנו יכולים להכניס את כל המידע הרצוי עבור יצירת המודל (הקורפוס של החבילה, הDICTIONARY של החבילה, ואת כמות הנושאים שייווצרו מהמודל.



נוסף במחלקת הPARSER, אנו משתמשים בחבילת nltk.corpus אשר בעזרתה אנו מסננים מילים נפוצות שלתעמינו לא יועילו בשלב האיחזור. כיוון שחשבנו על מילים נוספות, אנו הוספנו לחבילה זו מילים נוספות שלטעמינו לא נמצא בהם שימוש שיועיל לנו בתהליך האיחזור וכי הם לא מוסיפים מידע משמעותי לתוכן המסמך.



1. כיוון שהמודל שלנו הוא LDA, ולאחר שקיבלנו אישור להעלות יחד עם העבודה את המודל (פעמיים – אחד עם STEM ואחד בלי), על הבודק לבדוק שיש את הקבצים הבאים:  
   ldamodelwithstem.pkl

ldamodelwithoutstem.pkl

ldadictionarywithstem.pkl

ldadictionarywithstem.pkl

לאחר בדיקה שקבצים אלה אכן קיימים, ניתן להריץ את הפרויקט, כך שהקורפוס שייכנס יתבסס על המודל ששמרנו מבעוד מועד.

1. 3 יתרונות:
2. המודל שלנו מתבסס על אימון בצורה כזו שהוא מעבד את כל ה10 מליון DOCUMENTS ודואג לשייך לכל מסמך את ה3 נושאים, בעלי ההסתברות הגדולה ביותר שהם שייכים אליו. במצב כזה אנו דואגים לכך שלא בהכרח אם יש התאמה גדולה של מילים בין המסמך לבין השאילתה, אנו נחזיר את התוצאה, ייתכן שגם נחזיר מסמכים שמדברים על אותו נושא ויכולים להיות קשורים (כמובן שבשלב הרנקר ככל הנראה שהם ידורגו נמוך יותר ממסמכים בעלי התאמה גבוהה יותר של מילים אך ייתכן מאוד שגם מסמכים אלה יחזרו).
3. אנו שומרים את ציטוטים, כלומר במידה ויש לנו ציטוט אנו מזהים זאת בשלב הפארסר ומלבד פירסור של כל TERM בנפרד בתוך הציטוט, אנו שומרים גם את כל הציטוט כאחד. כתוצאה מזה, בשלב האיחזור במידה וישנו ציטוט בשאילה אשר קיים את אותו ציטוט גם באחד המסמכים, הוא ככל הנראה יקבל דירוג גבוה במיוחד כיוון שישנו התאמה גדולה בין המילים.
4. מלבד עניין המודל LDA שקיבלנו, מימשנו גם את המודל הבוליאני, כך שייצא מצב שאיכות האיחזור תצא טובה יותר, כיוון שאנו בודקים לא רק את התאמת הנושא בין הטוויט לשאילתה, אלא גם מוודאים שיש התאמה (מספיק מילה אחת מותאמת) עלמנת להעביר את המסמך לרנקר. אנו רואים במודל הבוליאני עזרה למודל LDA כיוון שהרגשנו שלהסתמך על המודל בלבד בעייתי כיוון שאין הבטחה כלל שאם ישנו קשר על פי נושא בים טוויט לשאילתה אז, ישנו צורך להחזיר את המסמך בהרכח (כיוון שלפעמים במודל אינו עובד בצורה יעילה ומייטבית על טוויטים ולכן החלטנו להוסיף "עזרה" כדוגמת המודל הבוליאני).

3 חסרונות:

1. בעיית מקום – כיוון שאנו משתמשים במודל LDA, אנו צריכים לשמור עבורו על פי המודל בDICTIONARY ייעודי (של חבילת GENSIN) את כל המילים של כך מסמך וכך הוא מבצע עיבוד כך שביצירת המודל הוא נעזר במילון זה, (מבחינתו כאשר אני מביא לו את המילון זה כל המילים במאגר שיקבל אותו מודל).

בנוסף אנו שומרים כפי שציינתי בסעיף g עבור האינדקסר סעיף 2, את כל רשימת הTOKENS של אותו מסמך (כל כמות של 50 אלף ) לקובץ פוסטינג ולאחר מכן, אנו עושים MERGE לקובץ פוסטינג אחד גדול, וכך אנו מעבירים את כל רשימת הרשימות האלה ליצירת המודל (מבחינת המודל מסמך מוגדר כרשימה של TOKENS), לאחר מכן אנו מוחקים את הרשימה מקובץ הפוסטינג אך בעת יצירה המודל אנו נדרשים להרבה מאוד מקום כדי שהמודל יוכל לעבד את המידע בצורה מיטבית.

1. בעיית זמן – יצירת המודל זוהי פעולה שלוקחת המון זמן, ולכן כל פעם שבו נרצה לעבד את המידע מ0, כלומר ליצור מודל חדש עבור קורפוס חדש (ניתן כמובן להתבסס אחרי הרצה אחת על המודל הקיים שנוצר, אך במידה ואנו רוצים ליצור מנוע חיפוש שמבוסס על המודל מחדש אנו נדרשים לשעות רבות של עבודה (כמובן תלוי גודל קורפוס)). חשוב להוסיף שאחרי הפעם הראשונה אנו מתבססים על המודל הקיים ששמרנו בקובץ חיצוני , כך שקורפוס קטן יותר מהקורפוס של 10 מליון טוויטים יוכל לעבוד על אותו מודל מוכן.
2. המודל LDA מבוסס בסופו של דבר על כמות TOKENS גדולה עבור כל מסמך, כך הוא מעבד את המידע עבור מסמך בצורה המייטבית, ומשייך אותו לנושא בהסתברות אמינה יותר. כיוון שטוויטים רובם מייצגים משפט אחד ולפעמים המשפט קצר, קשה מאוד למודל לבוא לידי ביטוי בצורה מייטבית ולכן לפעמים יכול להיווצר מצב שהוא לא משתייך לנושא מסויים אם אין את אותה מילה שמקשרת בין הנושא למסמך. דבר זה יוצר בעיות כאשר ישנם שאילתות בעלי שיכולות לא להחזיר טוויטים מסויימים כיוון שאותו טוויט לא היה מספק מספיק עבור המודל ולכן לא שייך אותו לאותו נושא כראוי (בשלב הSEARCHER אנו מעבירים לרנקר רק מסמכים שבהם הנושא הכי משוייך לשאילתה המירבי או גדול מ0.6, כלומר במידה ושאילתה משוייכת לנושא בהסתברות גבוהה מ0.6 אנו מעבירים לרנקר את כל המסמכים שקשורים לאותו נושא).
3. במחלקת הSEARCHER כאשר המשתמש שואל שאילתה, ישנו השהייה של כדקה וחצי עד קבל התשובות, כיוון שאנו רוצים להביא למצב בו הרנקר אינו מבצע על כמות גדולה של טוויטים את הדירוג. אנו עושים זאת על ידי כך שמלבד החזרת כל הטוויטים בעלי אותו נושא משותף כמו השאילתה, אנו מעבירים לרנקר בנוסף את הטוויטים בעלי מילה אחת דומה לפחות, ולכן על מנת לעשות זאת אנו בודקים עבור כל טוויט, בקבצי הפוסטינג של הDOCUMENTS שם נמצאים כל הTOKENS עבור כל טוויט, ובודקים אם יש התאמה של לפחותמילה אחת בין הטוויט לשאילתה. דבר זה גורר המתנה כיוון שהקבצי DOCUMENT (ישנם 10) כבדים מאוד ומעמיסים על הRAM ולכן לוקח למחשב זמן להחזיר את התשובות.

