בס"ד

פרויקט- מבוא לבינה מלאכותית

מגישים:

ינקל ז'אוי – 324523075

נופר דוכן – 322599424

תוכן עניינים:

בס"ד

הסבר על הבעיה שאנו פותרים

בפרויקט שלנו נרצה לתת מענה לאנשים אשר מחפשים תשובות במכתבים אשר נכתבו על ידי הרבי מליבוביץ' ושמורים במערכת שלנו.

בפרויקט אנחנו מאפשרים למשתמש לקבל מכתב של הרבי מתוך המאגר הנמצא במערכת, כאשר המשתמש מכניס קלט של מילה בודדת באמצעות שני האלגוריתמים שלנו, המערכת שלנו תבצע פעולות על מנת להביא למשתמש את המכתב שהכי רלוונטי אליו וכך המשתמש יוכל לעיין במכתב הקשור ביותר למילת הקלט שהכניס.

המערכת שלנו מתחזקת מאגר של מכתבים מהשנים 1951-1994 שנענו על ידי הרבי מליבוביץ' לשאלותיהם של חסידיו או של אנשים שחיפשו מענה מפיו של הרבי, כאשר המכתבים מסווגים לנושאים שונים.

את המכתבים לקחנו מאתר:
https://www.vaadhatmimim.org/templates/articlecco_cdo/aid/5105295/jewish/Igros-Kodesh.htm

בחרנו בגישה היחידה שיכולה לפתור לנו את הבעיה וזוהי:

גישת ה- **N**euro **L**inguistic **P**rocessing

עליה נרחיב בהמשך וכן על 2 שיטות מתחום זה העוסק בעיבוד שפה טבעית.

לפתרון הבעיה מצאנו שתי שיטות מתחום ה-NLP :

- TF-IDF: **T**erm **F**requency–**I**nverse **D**ocument **F**requency
- BOW: **B**ag **O**f **W**ords

במהלך הפרויקט, נכיר את שתי השיטות באמצעות סקירה ספרותית, נבנה מערכת הפותרת את הבעיה באמצעות שתי שיטות אלו, נשווה בין שתי השיטות ונבין מה השיטה העדיפה ביותר. ולבסוף, נסכם את מה שלמדנו מביצוע הפרויקט על השיטות, על קשרן לתחום הבינה מלאכותית ועל הבינה מלאכותית עצמה.

בס"ד

סקירה ספרותית - מדוע בחרנו בגישות אלו לפתרון הבעיה?

קצת על ה-NLP

התחום הרלוונטי לפתרון הבעיה הוא תחום חשוב ב- AI הנקרא Natural language processing (NLP), זהו תחום שמתמקד ביכולת המחשב לעבד ולהבין שפה טבעית באופן דומה לאדם.

בעשור האחרון אנו עדים להתפתחות משמעותית בתחום ה-NLP הכולל מגוון של משימות כגון: ניתוח והבנה של טקסט הנכתב על ידי אדם, תרגום אוטומטי (כמו: google translate), זיהוי והבנת המשמעות של דיבור אנושי, זיהוי תבניות בטקסט, זיהוי ישות בשם- משמש לחילוץ מידע לישויות שמות מוכרות ולאחר מכן מסווג אותם למחלקות שונות, הבנת קריאת מכונה המבוססת על למידה עמוקה וניתוח נפח עצום של נתונים מצויים ועוד. ה-NLP הוא תחום חשוב ומרכזי בתחום הבינה המלאכותית שמשפר את יכולות התקשורת והשימוש במחשבים. מכיוון שכל המשתמשים אולי אינם בקיאים בשפה ספציפית למכונה, עיבוד שפה טבעית (NLP) נותן מענה למשתמשים שאין להם מספיק זמן ללמוד שפות חדשות או להגיע לשלמות בה, הוא נוצר כדי להקל על המשתמש ולספק את הרצון לתקשר עם המחשב בשפה טבעית. [1]

היכולת של מחשב להבין שפת אדם היא חיונית ומשמשת למנועי חיפוש, מערכות תמיכת לקוחות, בינה עסקית וכו'.

נוכל לראות כי הבעיה שלנו מתעסקת בתחום זה הקשור להבנת טקסט וניתוחו בכך שאנו שומרים מאגר של מכתבים שנכתבו בשפה טבעית ואנושית ומתוכם אנחנו רוצים להוציא למשתמש מכתב אחד שהכי רלוונטי אליו על פי קלט של מילה המבטאת את נושא המכתב שהמשתמש רוצה לייבא. נוכל לראות כי כל מכתב נכתב בשפה אנושית דבר שמתקשר לתחום שלנו הפותר את הצורך של המחשב להבין, לנתח ולעבד שפות אנושיות ולכן נצרך פתרון המשתמש ב- NLP בלבד.

קראנו מאמרים על שיטות מתחום ה-NLP שיהיו הרלוונטיות ביותר לפתירת הבעיה שלנו.

ראינו כי ניתן לפתור את הבעיה באמצעות שיטות שונות כגון:

- Keyword Embedding - זיהוי והבנת הקשרים בין מילים בטקסט על פי ההופעות שלהן ביחס למילים אחרות בטקסט. בעצם, אנו מחפשים לראות אילו מילים מופיעות יחד עם אחרות, ומה הקשרים ביניהן בהקשר של הטקסט. זה יכול לסייע בזיהוי מונחים משותפים, קשרים סמנטיים, או הקשרים פשוטים כמו הכללות, תיאום, או סדר מבני בטקסט.
- N-grams - הטקסט מחולק לסדרות רציפות של N מילים רצופות, הידועות גם כ- grams כל רצף מילים באורך N מיוצג כיחידה נפרדת. המטרה היא ללמוד את התדירויות וההופעות של כל רצף N מילים בטקסט ולהשתמש בזה כדי להבין את המשמעות והקשרים ביניהם.

סיבות לבחירת השיטות שלנו

במהלך הקריאה על עוד שיטות שונות, שמנו לב שיש חשיבות לכך שהמכתבים שאנו שומרים במערכת כתובים בשפה העברית בשילוב מילים חריגות ביידיש, לכן הגענו למסקנה שכדי לפתור את הבעיה שלנו נרצה לחקור שיטות שמתמקדות:

- ב**תדירות** של מילת קלט **בודדת** מהמשתמש במאגר המכתבים שלנו.

- בביצוע השיטה על מילים חריגות ולא נפוצות ללא צורך בהתייחסות למשמעות המילה וללא צורך בקשר סמנטי בין המילים במכתבים הדורשים אימון רב והמון מידע שלא נמצא בידינו.

רצינו לחקור שיטות שיהיו פשוטות להבנה וליישום פרקטי, שיטות שלא דורשות התאמה ולמידה מקדימה של מודלים מורכבים, ואז הגענו לשיטות מתחום ה-NLP בהם נתמקד בהמשך: BOW ו- TF-IDF.

אלו שתי שיטות שמביאות לנו כפלט מכתב מתוך מאגר המכתבים באמצעות ניתוח ועיבוד שפה אנושית ובאמצעות חישובים שונים של תדירות הקלט מהמשתמש על פי מילת קלט בודדת ממנו, כדי להגיע לתוצאה הרצויה - מכתב שנכתב על ידי הרבי הקשור למילת הקלט מהמשתמש (ללא צורך בלמידת מודלים מורכבים מראש ופשוטים להבנה וליישום).

הקשר מתבטא באמצעות תדירות מילת הקלט בקבצים.

שתי השיטות הן חלק מתהליכי עיבוד שפה טבעית, שניהם נפוצים כאמצעים לייצוג טקסט והבינה מלאכותית משתמשת בייצוגים אלו למטרות שונות כמו ניתוח טקסטואלי, קידום תוצאות חיפוש, זיהוי קטגוריות, תרגום, ועוד, דבר שמתקשר לבעיה שאנו רוצים לפתור.

קצת על שיטת TF-IDF (Term Frequency-Inverse Document Frequency)

השיטה הראשונה שבחרנו כדי לפתור את הבעיה שלנו היא TF-IDF הנמצא תחת תחום העיבוד טקסט והבנת שפה טבעית. משמעות השיטה היא שימוש בתדירות מונחים- תדירות מסמכים הפוכה (TF-IDF). שיטה זו היא טכניקה סטטיסטית פופולרית המשתמשת בעיבוד שפה טבעית ואחזור מידע כדי להעריך את המשמעות של מונח במסמך בהשוואה לקבוצת מסמכים, המכונה קורפוס. הטכניקה משתמשת בתהליך וקטגוריזציה של טקסט כדי להפוך מילים במסמך טקסט לערכים מספריים המציינים את חשיבותם. קיימות שיטות ניקוד שונות עבור וקטגוריזציה של טקסט, כאשר TF-IDF הוא בין הנפוצים ביותר. TF-IDF שימושי ביישומי עיבוד שפה טבעית רבים. לדוגמה, מנועי חיפוש משתמשים ב-TF-IDF כדי לדרג את הרלוונטיות של מסמך עבור שאילתה. TF-IDF מועסק גם בסיווג טקסט, סיכום טקסט ומודלים של נושאים. [2]

<u>האלגוריתם מחשב את הניקוד למילה על פי השלבים הבאים:</u>

ראשית, נחשב את תדירות המונח (הקלט) שהמשתמש הכניס -

כמות הפעמים שהמילה מופיעה בקובץ ביחס לכמות המילים הכוללת בקובץ.

$$TF = \frac{number\ of\ times\ the\ term\ appears\ in\ the\ document}{\text{total number of terms in the document}}$$

שנית, נחשב עד כמה מונח חשוב בכל מאגר קבצי ה-txt. הדרך לחישוב נעשית כך שנסתכל על היחס בין כמות המכתבים שבהם נמצא הקלט מהמשתמש לבין הכמות הכוללת של כל הקבצים השמורים במערכת, לבסוף נעשה log על התוצאה מכמה סיבות:

- <u>הפחתת גודל</u>: קנה מידה לוגריתמי מפחית את גודל ערכי ה-IDF. זה מועיל מסיבות חישוביות וגם לפירוש טוב יותר של הערכים. ערכי IDF גדולים במיוחד עלולים להקשות על השוואת חשיבותם של מונחים בין מסמכים.
- <u>העדפה למונחים נדירים:</u> קנה מידה לוגריתמי מדגיש את חשיבותם של מונחים נדירים תוך מתן משקל מסוים למונחים נפוצים. למונחים נדירים, מעצם הגדרתם, יש ערכי IDF גבוהים יותר, אבל קנה מידה לוגריתמי מבטיח שהעלייה לא תהיה בלתי מידתית. זה יוצר איזון בין מונחים נדירים ונפוצים בעת חישוב החשיבות הכוללת של מונח במסמך.

$$IDF = \log\left(\frac{total\ number\ of\ the\ documents\ in\ the\ corpus}{\text{number of documents in the corpus contain the term}}\right)$$

נשים לב שיש כמה גישות שונות לחישוב ציון IDF. לעתים קרובות, לוגריתם הבסיס 10 משומש בחישוב. עם זאת, חלק מהספריות משתמשות בלוגריתם טבעי. בנוסף, ניתן להוסיף למכנה 1 באופן הבא כדי להימנע מחלוקה באפס.

$$IDF = \log\left(\frac{total\ number\ of\ the\ documents\ in\ the\ corpus}{\text{number of documents in the corpus contain the term} + 1}\right)$$

והשלב הסופי נתינת הציון המבטא את ערך החשיבות ע"פ מכפלה של תדירות המילה שהמשתמש הכניס ושל חשיבות המילה בכל מקבץ המכתבים:

$$TF - IDF = TF\ x\ IDF$$

בחרנו באלגוריתם זה כדי לפתור את הבעיה שלנו, מכיוון שהאלגוריתם יבחר לנו את המכתב שבו הקלט יהיה בעל הציון הכי גבוה המדגיש את החשיבות המרבית של המילה במכתב וכך המשתמש יוכל לקבל מכתב הנכתב על ידי הרבי מליבוביץ' שהכי רלוונטי אליו מבין כל המכתבים השמורים במאגר **תוך התחשבות בתדירות הקלט שהמשתמש הכניס ובחשיבותו.**

קצת על שיטת BOW (bag of words)

מודל BOW הוא טכניקת הטמעת מסמכים פשוטה המבוססת על תדירות מילים. מבחינה קונספטואלית, אנו חושבים על המסמך כולו כעל "שקית" של מילים, ולא על רצף. אנו מייצגים את המסמך פשוט לפי התדירות של כל מילה. לדוגמה, אם יש לנו אוצר מילים של 1000 מילים, אז המסמך כולו יוצג על ידי וקטור בעל 1000 ממדים, כאשר הערך ה-i^{th} של הווקטור מייצג את התדירות של מילת אוצר המילים ה-i^{th} במסמך. [3]

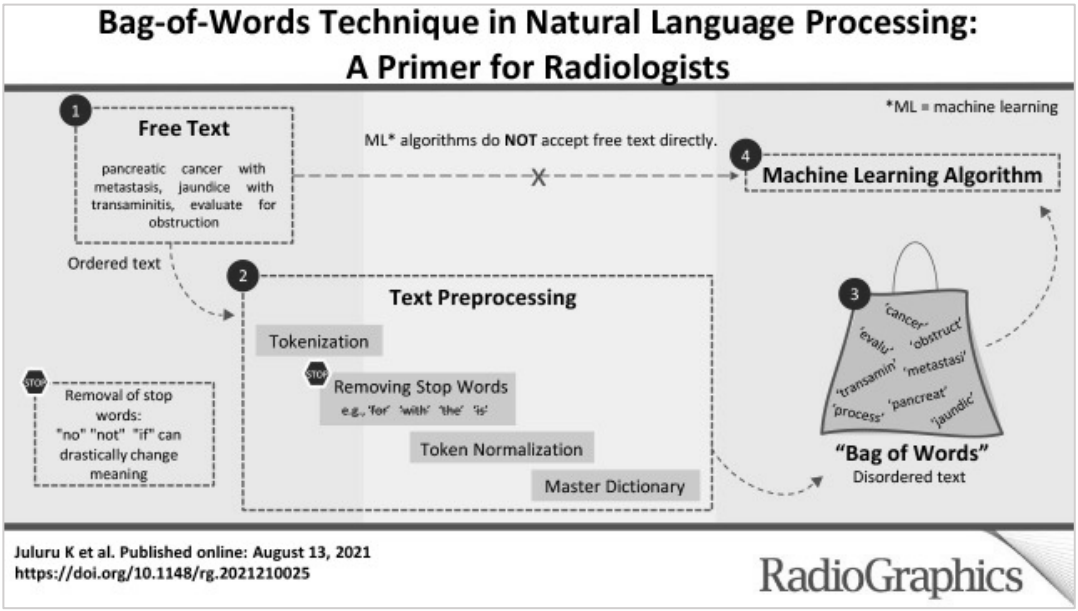
באמצעות טכניקה זו, נוכל להטמיע סט שלם של מסמכים ולהזין אותם במגוון אלגוריתמים שונים של למידה חישובית כגון: סיווג- משימת למידת מכונה שמסווגת מסמך כשייך לאחד מסוגים רבים. המודל אינו מייצר תכונות שימושיות במיוחד עבור משימות אחרות כמו, למשל, מענה לשאלות או סיכום, מכיוון שהן דורשות הבנה סמנטית יותר של המסמך וחייבות לקחת בחשבון את ההקשר. עבור משימות סיווג רבות, לעומת זאת, עצם הנוכחות והתדירות של מילים מסוימות מעידה מאוד על הקטגוריה של המסמך. כמה שימושים נפוצים בשיטת שקית המילים כוללים סינון דואר זבל, ניתוח סנטימנטים וזיהוי שפה. לעתים קרובות אנו יכולים להחליט אם הודעת דואר אלקטרוני היא ספאם או לא לפי התדירות של ביטויי מפתח מסוימים, כגון "פעל עכשיו" ו"תשובה דחופה".

ואם אוצר המילים שלנו כולל מילים שנלקחו משפות רבות, די קל לראות מתי כל המילים במסמך אחד מגיעות משפה אחת בפרט מבלי להזדקק להבנה הקשרית מעמיקה של הטקסט עצמו.

מכיוון שהטמעה זו כל כך בסיסית, היא לא עובדת טוב במיוחד עבור משימות מורכבות. אבל זה מתאים לבעיות סיווג פשוטות. הפשטות וקלות השימוש שלו הופכים אותו לבחירה אטרקטיבית.

שיטה זו מתאימה לפתרון הבעיה שלנו בכך שאנו מסתכלים על התדירות של מילת הקלט, בכל אחד מקבצי הtxt במאגר המכתבים שלנו, ונבחר להציג למשתמש כפלט את המכתב בעל התדירות המקסימאלית של המילה על ידי מודל BOW שעוזר לנו על מנת לבצע זאת.

נמחיש את פעולת השיטה באופן ויזואלי: [4]



נוכל לראות שלשיטה מס' שלבים:

1. קבלת קובץ טקסט
2. הפיכת הקובץ למילון השומר אוצר מילים:
   - הקובץ מתפרק למילים או סימנים בודדים (מחרוזות מינימליות בעלות משמעות)
   - ייצוג מילים שמשמעותיות בקובץ באוצר מילים ע"פ כללים שנקבעו מראש ע"י המתכנת.
   - נרמול – במידה והשפה היא אנגלית, הפיכת כל המילים לאותיות קטנות כדי שלא יהיו מילים עם אותה משמעות אך כתובות שונה ותהיה פגיעה בפלט האלגוריתם.
   - הפיכת מילים לצורת השורש שלהם וכך מילים המעידות על אותה כוונה אך כתובות בבניין שונה עדיין יספרו באותה פעימה, דבר שישפיע על התוצאה של הפלט הסופי, בעצם גורם להתייחסות למספר צורות של מילה כאל אותה ישות.
3. לאחר החלת שלבי העיבוד המקדים הללו, נשארת קבוצה של מילים המייצגות את אוצר המילים של מאגר הקבצים. מילים אלו ישמשו ליצירת ייצוג שקית המילים של המסמכים במאגר, כאשר כל מסמך מיוצג כווקטור של ספירת מילים או תדרים שבהם ניתן להשתמש באלגוריתמים ללמידה חישובית.

סקירת המערכת

שלבים התחלתיים של בניית המערכת

לאחר שמצאנו את השיטות המתאימות לפתרון הבעיה שלנו, עשינו מחקר על אופן תפקודם ולפי איזה מדדים הם מחזירים את הפלט שלהם. התחלנו בשלב הביצוע הפרקטי של המערכת.

1. מציאת data רלוונטי בצורת txt כדי שנוכל להשתמש בפונקציות לקריאת טקסט- לא מצאנו מכתבים של הרבי הכתובים בקבצי txt, לכן היינו צריכים להמיר את קבצי ה-pdf שמצאנו עם המידע הרלוונטי לקבצי txt.
2. תחילת קידוד בסביבת העבודה- PyCharm התומכת בשפת python שמתחזקת ספריות רבות המשתמשות ב-AI.

שלבים לבניית BOW

1. בנינו את פונקציית bow הפעילה את השיטה שלנו, הפונקציה bow מקבלת את המילה **word_to_find** כפרמטר, שהיא המילה שרצינו לחפש בתוך המסמכים.
2. הגדרנו את המשתנה **input_folder** כנתיב לתיקיית המסמכים (בתוך התיקייה יש מסמכי טקסט בפורמט .txt).
3. ייבאנו את הספריות הדרושות: **os, nlt, random, tkinter, scrolledtext,** ו- **CountVectorizer** מתוך **sklearn.feature_extraction.text**. (מחלקת CountVectorizer ב-(sklearn) skit-learn היא כלי המשמש להמרת אוסף מסמכי טקסט למטריצה של ספירת tokens).
4. יצרנו רשימה ריקה **all_content_letters** שבה כל תא יכיל מכתב מתוך מאגר המכתבים הנקראים.
5. הגדרנו את המשתנה **stop_words** כאוסף של מילים בעברית שלא משמשות בדרך כלל כמקור למשמעות המסמך ונהוג להתעלם מהן בעת ניתוח טקסט. בניהן נמצאות מילים כמו "ובכן", "על-ידי", "לפי", "אם" וכו' בעזרת הספרייה **nltk**, ובאמצעות הפונקציה **stopwords.words('hebrew').**
6. עברנו על כל קובץ בתיקייה וביצענו את השלבים הבאים:
   - קריאת תוכן הקובץ באמצעות **open()** והמרתו לטקסט.
   - הפעלת פונקציה לעיבוד טקסט **preprocess_text** שעוברת על כל הטקסט ומבצעת פעולות כמו הסרת תווי רווח כפולים או מילים שלא בשפה העברית(מסננת את הטקסט).
   - פירוק הטקסט למילים בודדות.
   - הוספת הטקסט המעובד לרשימה **all_content_letters**.
7. נשתמש ב־ CountVectorizer מתוך **sklearn** כדי להמיר את הטקסט ל־ Bag of Words model. בנינו מופע השייך למחלקת CountVectorizer כדי שעליו נוכל להפעיל את פונקציית **fit()** המבצעת פעולות ניתוח טקסט. המופע מכיר את כל המילים השונות במאגר המכתבים ובונה מפתח עבור כל מילה, כאשר המפתח הוא המילה עצמה והערך הוא המיקום שלה במילון (זהו מילון של כל המילים שנפלט מהתהליך). כלומר, הפעולה fit מסייעת למופע ללמוד את המילים השונות שקיימות בטקסטים, וליצור את מילון המילים שעליו יבוצעו המדידות והניתוחים הנדרשים.
8. נשתמש בפונקציית **transform** המשמשת להפיכת נתוני טקסט לייצוג מספרי על מנת לבצע למידת מכונה בהמשך. הנתונים שעברו טרנספורמציה הם וקטור שניתן להציג אותו כמטריצה דלילה שבה כל שורה מתאימה למסמך וכל עמודה מתאימה למילה באוצר המילים. הערך בכל צומת שורה-עמודה מייצג את התדירות של מילה זו במסמך המתאים.
9. נמצא את אינדקס המילה **word_to_find** במילון המילים שנוצר על ידי CountVectorizer. אם המילה נמצאת, נשלוף מתוך המטריצה את תדירות המילה בכל קובץ ונמצא את המסמכים בהם היא מופיעה בתדירות הגבוהה ביותר.
10. אם המילה נמצאת במילון, נמצא את המסמך המקורי שבו המילה מופיעה בתדירות הגבוהה ביותר ונציג אותו בחלון חדש (במידה וישר יותר מ-1 נדפיס למשתמש מכתב רנדומלי מבניהם).
11. אם המילה לא נמצאת במילון, נציג הודעה מתאימה.

שלבים לבניית TF-IDF

1. בנינו את פונקציית bow הפעילה את השיטה שלנו, הפונקציה bow מקבלת את המילה **word_to_find** כפרמטר, שהיא המילה שרצינו לחפש בתוך המסמכים.
2. הגדרנו את המשתנה **input_folder** כנתיב לתיקיית המסמכים (בתוך התיקייה יש מסמכי טקסט בפורמט .txt).
3. ייבאנו את הספריות הדרושות: **os, nltk, random, tkinter, scrolledtext,** ו- **TfidfVectorizer** מתוך **sklearn.feature_extraction.text**. (מחלקת CountVectorizer ב-(sklearn) skit-learn היא כלי המשמש להמרת אוסף מסמכי טקסט למטריצה של tf-idf).
4. יצרנו רשימה ריקה **all_content_letters** שבה כל תא יכיל מכתב מתוך מאגר המכתבים הנקראים.
5. הגדרנו את המשתנה **stop_words** כאוסף של מילים בעברית שלא משמשות בדרך כלל כמקור למשמעות המסמך ונהוג להתעלם מהן בעת ניתוח טקסט. בניהן נמצאות מילים כמו "ובכן", "על-ידי", "לפי", "אם" וכו' בעזרת הספרייה **nltk**, ובאמצעות הפונקציה **stopwords.words('hebrew').**
6. עברנו על כל קובץ בתיקייה וביצענו את השלבים הבאים:
   - קריאת תוכן הקובץ באמצעות **open()** והמרתו לטקסט.
   - הפעלת פונקציה לעיבוד טקסט **preprocess_text** שעוברת על כל הטקסט ומבצעת פעולות כמו הסרת תווי רווח כפולים או מילים שלא בשפה העברית(מסננת את הטקסט).
   - פירוק הטקסט למילים בודדות.
   - הוספת הטקסט המעובד לרשימה **all_content_letters**.
7. יצרנו את המחלקה **TfidfVectorizer** שתטפל בהמרת הטקסטים למטריצת TF-IDF.
8. השתמשנו בפעולת **fit_transform** כדי ליצור את המטריצה המתארת את התדירויות והחשיבות של כל מילה בקבצים.
9. לאחר מכן, המרנו את המטריצה למערך באמצעות **toarray()** כדי להשיג תצוגה מוכנה לחישובים נוספים.
10. שמרנו את רשימת שמות המילים במשתנה **word_in_files** כדי שנוכל לדעת האם מילת הקלט קיימת ברשימת המילים הזאת.
11. אם המילה **word_to_find** נמצאת במילים שנמצאו על ידי ה- TfidfVectorizer, נמצא את המסמך שבו התדירות הגבוהה ביותר של המילה נמצאת על ידי האינדקס שלו הנמצא במערך, ונציג אותו בחלון חדש.
12. אם המילה לא נמצאת, נציג הודעה מתאימה.

כיצד ניתן להפעיל את המערכת

- יש להיכנס לקובץ הקוד "BOW_TF-IDF_GUI" ב- PyCharm או בסביבת עבודה התומכת ב-Python ובהורדת ספריות.
- חשוב להוריד את הספריות המיובאות בתחילת הקובץ.
- יש להוריד את תיקיית המכתבים המצורפת לפרויקט.
- ב-2 הפונקציות של שתי השיטות צריך לכתוב את הנתיב שבו נמצאת תיקיית כלל המכתבים.
- יש ללחוץ על הרצה.
- אמור להיפתח חלון GUI, יש להכניס קלט בעברית וללחוץ "process".

#נראה שבסוף קובץ הקוד יש בהערה קוד לבדיקת זמן הריצה של השיטות,

ונשים לב זמן הריצה משתנה מהרצה להרצה.

דילמות תכנוניות:

1. הדילמה: לקח לנו זמן למצוא מאגר מכתבים של הרבי בצורת txt כדי שהמערכת שלנו תוכל לקרוא אותם, כל מה שמצאנו היה רק בצורת pdf
   הפתרון: מצאנו דרך להמיר את קבצי ה-pdf לקבצי txt
2. הדילמה: מחשב MacBook עם מערכת הפעלה Sur Big MacOS, לא תמך בהתקנת ספריות הרלוונטיות אלינו.
   הפתרון: הרצנו את הקוד על מחשב אחר בסביבת עבודה PyCharm
3. הדילמה: המכתבים שלנו כתובים בשפה העברית המשולבת עם יידיש וראשי תיבות, דבר שהעלה חשש שהניקוי של מילים לאוצר מילים רלוונטי לא יעבוד בצורה טובה
   הפתרון: הורדנו ספריה התומכת בשפה העברית ומאפשרת לנקות תווים על פי בקשת המתכנת.
4. הדילמה: מציאת מאגר מכתבים גדול יותר.
   הפתרון: שאלנו את המרצה ואמרה שזה מספיק בשביל פרויקט זה, במקביל להרצות שביצענו כדי לבדוק את הדיוקים של השיטות ואת זמני הריצה, קראנו מאמרים שיתמכו בתוצאות שלנו.

השוואות והבדלים בין השיטות לפתרון הבעיה

השוואה בין שתי השיטות BOW ו- TF-IDF

זמן ריצה:

לאחר שבדנו את זמן הריצה של השיטות שלנו הגענו לתוצאות שהופתענו מהן:

לאחר הרצה של 5 פעמים:

```
The runtime of the BOW function  0.7079585000017412
The runtime of the TF-IDF function  0.5734477000078186
```

לאחר הרצה של 100 פעמים:

```
The runtime of the BOW function  10.6798610999831
The runtime of the TF-IDF function  9.736366100027226
```

לאחר הרצה של 1000 פעמים:

```
The runtime of the BOW function  148.79298110003583
The runtime of the TF-IDF function  115.02990770002361
```

תחילה ראינו שההבדל בזמני הריצה הוא מינורי, דבר לו ציפינו לאחר התעמקות בנושא, אך ההפתעה שלנו נבעה כאשר ראינו שזמן הריצה של TF-IDF קטן יותר מאשר של BOW. ההפתעה שלנו נבעה מכך שחשבנו שהתוצאה תהיה הפוכה, מכיוון ש-TF- IDF מבצע יותר חישובים בהשוואה ל- BOW כגון: קנה מידה לוגריתמי וכפל.

ולאחר מכן הבנו כי:

- תוצאת זמן הריצה יכולה להשתנות בהתאם לגורמים כגון גודל מאגר הקבצים, ומשאבי החומרה הזמינים.
- החישובים המוקדמים של TF ו־IDF יכולים להביא לכך שהפונקציה של TF-IDF תוכל להיות יעילה יותר מבחינת זמן הריצה מאשר פונקציה של BOW, שאינה עושה שימוש בחישובים מוקדמים כמו אלו.

יעילות מקום

גם BOW וגם TF-IDF הם ייצוגים יעילים בזיכרון עבור נתוני טקסט, במיוחד כשהם מיושמים באמצעות מטריצות דלילות (לדוג': שימוש במטריצה דלילה ב-BOW, המשמעות היא שרוב הערכים במטריצה יהיו אפס, מכיוון שרק תת-קבוצה קטנה של אוצר המילים כולו תופיע בכל מסמך נתון). השימוש בפועל בזיכרון יהיה תלוי בגורמים כגון גודל אוצר המילים, הדלילות של מערך הנתונים. בפועל, ההבדל ביעילות המקום בין BOW ל-TF-IDF הוא לרוב מינימלי, ושני הייצוגים מתאימים לטיפול במאגרי טקסט גדולים עם משאבי זיכרון מוגבלים.

עם זאת, TF-IDF עשוי לדרוש מעט יותר זיכרון בהשוואה ל-BOW מכיוון שבנוסף לשימוש במטריצה הדלילה כדי לאחסן תדירות של מונח (TF), הוא צריך לאחסן גם את ערכי תדר המסמך ההפוכה (IDF), אך העלייה בשימוש בזיכרון בדרך כלל אינה משמעותית.

דיוק הפלט:

ראינו בהרצת המערכת שלנו ששיטת TF-IDF מביאה לנו פלט מדויק יותר מאשר אלגוריתם BOW מכיוון שבניגוד לאלגוריתם BOW , אלגוריתם TF-IDF מביא לנו פלט של מכתב שהמילה שהוכנסה כקלט נמצאת בו עם **חשיבות מקסימלית** בכלל מאגר המכתבים שקיים במערכת לעומת BOW שמתייחס לתדירות המקסימאלית של מילת הקלט ללא התחשבות בחשיבותה. והכוונה היא:

- כאשר יש מכתב X שיש בו פעם אחת את המילה word והוא קצר יותר- החשיבות של המילה גבוהה יותר.
- וכאשר יש מכתב Y שיש בו פעמיים את המילה word והוא ארוך בהרבה יותר- החשיבות של המילה היא פחותה במכתב זה.

הפלט על פי TF-IDF יהיה X בעוד שהפלט על פי BOW יהיה Y וזאת משום ש-BOW מתייחס רק לקובץ בעל התדירות המקסימאלית של המילה ו-TF-IDF מתייחס לחשיבות המקסימאלית של המילה המושפע מאורך של הקובץ.

לפי מאמר [5], ראינו הוכחה כי ההבדל בדיוק של שתי השיטות מתבטא גם כאשר נעשתה השוואה בין השיטות עם מודלים שונים של למידת מכונה לזיהוי דברי שטנה מציוצים חיים:

שימוש ב-BOW לזיהוי דיברי שטנה מציוצים חיים:

S/N	Algorithm	Accuracy
1	Naïve Bayes	25.45%
2	KNN	66.21%
3	Logistic Regression	74.79%
4	Decision Tree	67.16%

שימוש ב-TF-IDF לזיהוי דיברי שטנה מציוצים חיים:

S/N	Algorithm	Accuracy
1	Naïve Bayes	75.27%
2	KNN	85.76%
3	Logistic Regression	90.46%
4	Decision Tree	92.43%

נראה כי התוצאות ברורות, ושימוש ב- TF-IDF עדיף על שימוש ב-BOW כאשר אנחנו רוצים פלט מדויק יותר בשביל המשתמש.

במערכת שלנו הדיוק של הפלט הוא חשוב כדי להביא למשתמש את המכתב שהכי מתקשר למילת הקלט אותה מכניס למערכת. חשיבות המילה בקובץ ובכלל מאגר הקבצים מראה על הקשר בין המכתב (הפלט) למילת הקלט של המשתמש. ככל שהחשיבות של המילה בקובץ גבוהה יותר ככה אותו קובץ הוא המדויק ביותר בשביל המשתמש.

יוריסטיקה:

ב-BOW, ניתן לראות יוריסטיקה כאשר אנו מחפשים את המכתב בעל התדירות הגבוהה ביותר של מילה שהמשתמש הכניס כקלט, אנו משתמשים בהנחה כי **למכתבים שבהם המילה מופיעה בכמות גבוהה יותר יש סיכוי גבוה יותר להתאים לנושא המכתב שהמשתמש מחפש.**

ב-TF-IDF, יש פונקציה יוריסטית נוספת שמתייחסת פחות לשכיחות ויותר לחשיבות. במקום פשוט לספור את מספר ההופעות של מילה במסמך (כמו ב-TF), מכניסים כאן את שקלול התדירות ביחס לכל המכתבים במאגר (IDF). כך, באמצעות הפונקציה היוריסטית, ניתן להעריך את **החשיבות של מילה מסוימת במסמך מסוים ביחס לכל המסמכים במאגר המכתבים. בדרך זו, ניתן לקבוע את המכתבים הרלוונטיים ביותר למילת קלט של המשתמש.**

לכן במקרה זה עדיף להשתמש ב-TF-IDF שיקרב אותנו לדיוק מקסימאלי בתוצאת הפלט שהמשתמש מצפה לקבל.

הבדלים בין פתרון Keyword Embeddings לבין פתרון השיטות BOW ו-TF-IDF

בעת החקר על פתרון הבעיה שלנו, מצאנו פתרונות שונים לפתירת הבעיה, אחד מהפתרונות הוא:

שיטת Keyword Embeddings, שיטה זו שונה מ-BOW ו-TF-IDF במספר דרכים:

ייצוג מילים:

BOW ו-TF-IDF מייצגים מילים בתוך מסמך באופן ספרתי או דו-מימדי, כך שכל מילה מיוצגת על ידי ערך במישור מילים ומסמך מסוים. השיטה אינה מתייחסת לקשרים בין המילים.

בניגוד לכך, Keyword Embeddings מייצגת מילים על ידי וקטורים מרוחבים במרחב רב-מימדי, כאשר כל וקטור מייצג את המשמעות והקשר של מילת המפתח בתוך השפה. המילים מיוצגות לפי הקשרים הסמנטיים והדקדוקיים שלהן בתוך השפה, ולא רק על פי מיקומן במסמך או תדירותן.

חישוב מרחקים:

BOW ו-TF-IDF אינן משתמשות בחישובי מרחקים או באלגוריתמים של דמיון סמנטי. הן מתמקדות בתדירויות מילים במסמך ובהתאמתן למסמך המחפש.

Keyword Embeddings משתמשת בחישובי מרחקים ובאלגוריתמים של דמיון סמנטי כדי למצוא את המסמך הכי מתאים.

שימוש:

BOW ו-TF-IDF נמצאות בשימוש נרחב בעיבוד שפה טבעית למשימות כמו סיווג טקסט, תיוג וחיפוש מילות מפתח.

Keyword Embeddings נמצאת בשימוש בעיקר ביישומים בהם חשוב המידע הסמנטי של המילים ולא רק התדירות או המיקום שלהן במסמך. השימוש המקובל ביותר הוא בתחום שיווק באינטרנט, קידום אתרים, ובכלים של חיפוש חכם.

סיכום של השיטות: BOW ו- TF-IDF

בעקבות הכנת הפרויקט והרצת הפתרונות על המערכת שלנו למדנו מספר דברים על BOW ו- TF-IDF

1. שתי שיטות אלו הם מתחום ה-NLP- עיבוד שפה טבעית.
2. שתי השיטות מאפשרות ליצור מודלים פשוטים שמבוססים על הופעת המילים בטקסט, הן מאפשרות לנו לבצע משימות פשוטות ב-NLP כמו סיווג טקסטים או זיהוי נושאים.
3. באמצעות שתי השיטות, ניתן להבין אילו מילים מופיעות במסמך ועל מה המסמך מדבר, הן מאפשרות לנו להבין נושאים ראשוניים בטקסט ולזהות מונחים רלוונטיים.
4. שתי השיטות מהוות חלק מתהליך עיבוד השפה הראשוני, אשר יכול להתבצע בשלב מוקדם של העיבוד, התוצאות של BOW ו-TF-IDF יכולות לשמש כקלט למודלים מורכבים יותר כמו מודלי רשתות נוירונים או סיווגים סטטיסטיים – דבר המתקשר ללמידת מכונה.
5. שתי השיטות הן יעילות מבחינת זמן ומקום, ההבדל ביניהן הוא לא משמעותי בקריטריונים האלה, שניהם יכולים לעבוד מעולה עם מאגרי נתונים גדולים.
6. מה שמייחד את שתי השיטות האלה בדרך פתרונם הוא הסתכלות על תדירות מילה בטקסט או במאגר קבצים. בעוד ש-BOW קובע מה הפלט הרלוונטי למשתמש על פי התדירות המקסימאלית של מילה בקובץ, TF-IDF קובע מה הפלט הרלוונטי למשתמש על פי חשיבות המילה הן בקובץ ספציפי והן מבחינת מאגר הקבצים בכלל- דבר שראינו שמשפיע על דיוק הפלט המוצג למשתמש. שתיהן פחות מסתכלות על משמעות המילה אלא יותר על תדירותה.
7. שתי השיטות בתחילת התהליך שלהן, מנתחות ומעבדות את הטקסט ומנקות מילים הנפוצות בשפה (כגון: "או", "יש", "גם" וכו') כדי להפחית השפעה של המילים הנפוצות האלה על תוצאת הפלט.
8. שתי השיטות משתמשות בווקטורים מספריים על מנת שיהיה ניתן להשתמש בהם בהמשך באלגוריתמים ללמידת מכונה.

תובנות בתחום הבינה המלאכותית

תחום הבינה המלאכותית הוא מאוד מרתק, רחב, מגוון ומסקרן, במהלך חקר על שתי השיטות שלנו שהן תחת תחום ה-NLP הבנו כי הן לא אלגוריתמים ממש של בינה מלאכותית אלא הן שתי שיטות המסייעות לאלגוריתמים מתחום ה-AI-אלגוריתמים של למידת מכונה.

עם חקר הנושא והבנתו גילינו את הקשר בין BOW, IDF-TF, NLP ו-AI.

ראינו כי NLP (Neuro linguistic processing) מספקת את הכלים והטכניקות לעיבוד וניתוח נתוני טקסט, אשר משולבים לאחר מכן במסגרת הרחבה יותר של למידת מכונה כדי לבנות מודלים ולפתור משימות NLP שונות.

- ניתן לבצע עיבוד מקדים של נתוני טקסט באמצעות טכניקות NLP כגון: טוקניזציה (חילוק הטקסט ליחידות קטנות בעלות משמעות), נרמול (הפיכת אותיות גדולות לקטנות) והסרת מילות עצירה. שלב זה עוזר לנקות ולנרמל את נתוני הטקסט לניתוח נוסף.
- לאחר מכן, ניתן להשתמש בשיטות NLP כדי לחלץ תכונות רלוונטיות מנתוני הטקסט המעובדים מראש. וכאן נכנס התפקיד של השיטות שלנו: BOW (bag of words) ו- TF-IDF.
- ולבסוף, לאחר הכנת התכונות של הטקסט, ניתן לבחור באלגוריתם מתאים של למידת מכונה (כגון: סיווג, רגרסיה וכו'). ולאמן את מודל למידת המכונה הנבחר באמצעות התכונות שחולצו. במהלך האימון, המודל לומד את הדפוסים והקשרים בין תכונות טקסט הקלט לבין פלטי היעד המתאימות.

מהתוצאות שצירפנו ממאמר [5] למדנו כי, TF-IDF עובד טוב יותר עם אלגוריתם למידת מכונה "עץ החלטה" בעוד ש- BOW עובד טוב יותר עם אלגוריתם למידת מכונה "רגרסיה לוגיסטית" כאשר TF-IDF תמיד מביא לי תוצאות מדויקות יותר מאשר BOW עבור אלגוריתמי למידת מכונה (בנוסף לשנים הקודמים גם עבור: Naïve Bayes ו- KNN). ומכאן ששיטות שונות על אלגוריתמי למידת מכונה משפיעות באופן ישיר על תוצאת האלגוריתמים.

ביבליוגרפיה:

[1] Khurana, D. K. (2023). Natural language processing: state of the art, current trends and challenges. Multimedia Tools and Applications, 3713-3744.

קישור: https://link.springer.com/article/10.1007/s11042-022-13428-4

[2] TF-IDF (Term Frequency-Inverse Document Frequency). (n.d.). Retrieved from Learn Data Science.

קישור: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/

[3] Bag of Words Model. (n.d.). Retrieved from Built In.

קישור: https://builtin.com/machine-learning/bag-of-words

[4] Xu, Y., Jiang, H., Wang, W., Zhang, C., Cui, Y., & Liu, Y. (2023). A Survey on Text Summarization with Deep Learning. IEEE Access, 11, 11930-12003.

קישור: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8415041/#r10

[5] Zhang, X., Zhao, S., Li, Z., & Yu, W. (2023). A Multi-Head Attention Based Model for Automatic Text Summarization. International Journal of Computational Intelligence and Systems, 20(1), 74-83.

קישור: https://link.springer.com/article/10.1007/s41870-022-01096-4