Deep Learning- Project

הלמידה העמוקה (Deep Learning) היא תחום מחקר בעולם המחשבים שמטרתו לחקות באופן ממוחשב את פעולת המוח האנושי. התחום הוא מרכזי בעולם האינטליגנציה המלאכותית. המיוחד במערכות למידה עמוקה הוא היכולת שלהן ללמוד ולהשתפר כל הזמן. ממש כמו המוח האנושי, ככל שמערכת כזו פועלת וככל שמשתמשים בה - היא משתפרת ו"יודעת" יותר.

כחלק מכין לביצוע הלמידה העמוקה, ביצענו ניתוח לנתונים, תהליך המורכב מבחינה, ניקוי, ביצוע טרנספורמציה ומידול של הנתונים. זאת במטרה של הדגשה והבלטה של המידע השימושי, ושיפור תהליך הלמידה והסקת המסקנות.

בתהליך ה-Deep Learning אנו מבצעים ניתוח נתונים שמתמקדת ביצירת מודל וגילוי ידע למטרות של חיזוי.

בפרויקט שלנו התבקשנו, באמצעות למידה עמוקה (Deep Learning) וחקר נתונים (data science), לחזות ולייצר טקסט הדומה לטקסט שנכתב על ידי אדם. בעבודתנו חקרנו תסריט של הסרט  
 The wolf of wall street"", שנכתב על ידי Terence Winter.

המטרה המרכזית שלנו בתהליך היא יצירת משפטים כמה שיותר אמתיים ואותנטיים.

בפרויקט זה נציג איך על ידי שימוש בתסריט עבר, בנינו מודל שיכול ליצר משפטים חדשים לבד בסגנון המשפטים הכתובים בתסריט. כלומר, לבנות מודל אשר יכול לבנות משפט בעל n מילים אם נביא לו dataset מסויים של משפטים.

הסיכוי למשפט הוא הסיכוי של כל מילה להופיע אחרי כל המילים שהופיעו לפניה.

מודל זה יכול לעזור במגוון רחב של תחומים אשר כוללים למידה של שפה, לדוגמא, בשורת חיפוש של גוגל, הצגת המשך משפט צפוי על פי הטקסט שהמשתמש הזין לשורת החיפוש עד כה. בנוסף, נוכל גם ליצור משפטים חדשים.

כדי לממש זאת, אנו צריכים לזכור עבור כל מילה את הסיכוי שלה להופיע אחרי כל המילים שקיבלנו בdataset . לרוב המודלים יהיה קשה לייצג כזו תלות ארוכה עקב אילוצי חישוב וזכרון, הם מוגבלים להסתכל על כמות מוגבלת של מילים אחורה. RNN, בתיאוריה, יכול להתמודד עם זה.

לרשת נוירונים, יש שכבות נסתרות. לרוב, מצב השכבות החבויות שלנו מתבססות רק על הקלט: input -> hidden -> output.

שימוש ביכולת הזיכרון משנה את זה בכך שכעת השכבה הנסתרת לא תלויה רק בקלט אלא גם בחותמת הזמן שיש לקלט ובשכבה החבויה של חותמת הזמן האחרונה:

(input + prev\_hidden) -> hidden -> output

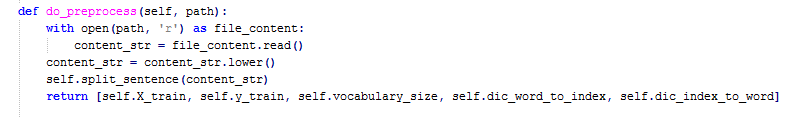
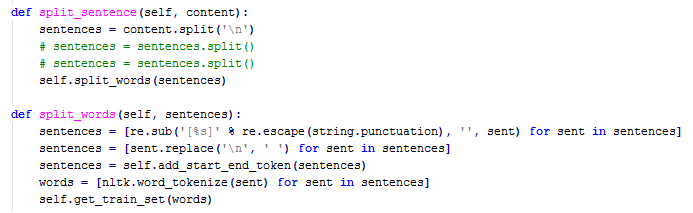
השכבה הנסתרת החוזרת, לומדת מה צריך לזכור בכל פעם.

הפרויקט:

1. Choosing training set- כאשר התחלנו את הפרויקט, חיפשנו טקסט אשר מצד אחד לא יהיה קצר מידי ומצד שני, שיכיל משפטים הגיוניים בשפה האנגלית על מנת שהמודל שלנו יצליח להוציא תוצאות טובות.

כמו כן, היה לנו חשוב לעבוד עם טקסט מעניין ורלוונטי.

לאחר כמה נסיונות, נשארנו עם התסריט לסרט – The wolf of wall street, שנכתב על ידי Terence Winter.

1. Training data and preprocessing – לאחר בחירת הטקסט עליו נרצה לאמן את המודל שלנו, היה צורך לבצע עליו עיבוד מקדים.
   1. Tokenize text- ניקח את התסריט כraw text, אבל אנו רוצים לבצע חיזוי על בסיס מילים, כלומר לחלק את הטקסט שלנו למשפטים וכל משפט למילים.   
      בקוד שלנו נעזרנו בNLTK's, שמפרק את הטקסט למשפטים, וכל משפט למילים. הפונקציות הם sent\_tokenize ו- word\_tokenize.

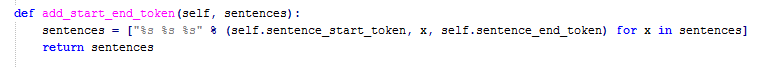
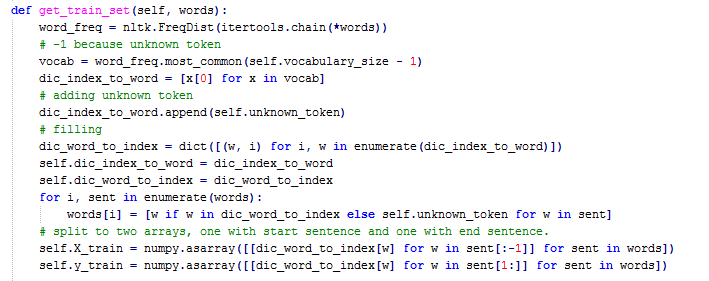
הקטנו את כל האותיות כך שהמילה Hello והמילה hello, יחשבו כאחת, והורדנו כל מיני סימני פיסוק על מנת שלא יפריעו לנו בניתוח. לדוגמא: נגמר! או נגמר. ייוצגו כשני מילים שונות.

* 1. Remove infrequent words- בטקסט רגיל כמו בכל טקסט או שפה, קיימות הרבה מילים אשר מופיעות לעיתים רחוקות בלבד, מילים אלו הם כמו רעש בשביל המודל שלנו ולכן אנו בוחרים להוציא אותם. מילון גדול יכול לגרום למודל שלנו לעבוד לאט יותר, ומכיוון שאין לנו הרבה דוגמאות מהמילים הללו, לא נצליח ללמוד איך משתמשים בהם כמו שצריך, ממש כמו כאשר בני אדם לומדים מילים, צריך לפגוש את המילה כמה פעמים בהקשרים שונים על מנת באמת להבין את המשמעות של המילה.  
     מתוך 3722 מילים שמרנו במילון את החשובות ביותר, ועכשיו הוא מכיל 2000 מילים.

כל מילה שהגדרנו אותה כמילה שמופיעה מעט מידי פעמים, נכנסה למילון בתוך משפט כ- UNKNOW\_TOKEN. המילה UNKNOW\_TOKEN הופכת להיות מילה במילון ונחזה אותה כמו כל מילה אחרת.

המילים השכיחות ביותר בטקסט הם:

* You- מופיע 989 פעמים
* The- מופיע 752 פעמים
* A- מופיע 517 פעמים
  1. Prepend special start and end tokensd- מכיוון שאנו רוצים לא רק לדעת מה המילה הבאה שתבוא אחרי מילה מסויימת, אלא אנו רוצים גם ליצור משפטים חדשים לגמרי, יצרנו 2 token-ים מיוחדים, אשר יסמנו מתי מתחיל משפט ומתי נגמר משפט, זאת על מנת שנוכל לשאול את השאלה, "מה המילה שתגיע אחרי הSENTENCE\_START"". לכן לאחר שפירקנו את הטקסט שלנו למשפטים הוספנו בתחילת כל משפט "SENTENCE\_START" ובסוף כל משפט "SENTENCE\_END".

* 1. Build training data- RNN מקבל כקלט וקטור ולא string ולכן אנו היינו צריכים לסדר את המשפטים שלנו כוקטורים כך שלRNN יהיה יותר קל לעבוד איתם.

הדרך הנוחה ביותר לעשות זאת היא על ידי הפיכת כל מילה לאינדקס. לדוגמא המילה Sleep היא באינדקס 529. 0 מייצג את token ההתחלה ו-1 מסמן את token הסיום.

עבור כל משפט ניתן לRNN 2 משפטים:

* + 1. משפט שמתחיל ב-0 (SENTENCE\_START) ואחריו כל המספרים המייצגים את המילים במשפט.
    2. משפט שמתחיל בלי-0, אלא את כל האינדקסים של המילים במשפט ונוסיף 1 (SENTENCE\_END) בסוף כדי לסמן סיום.

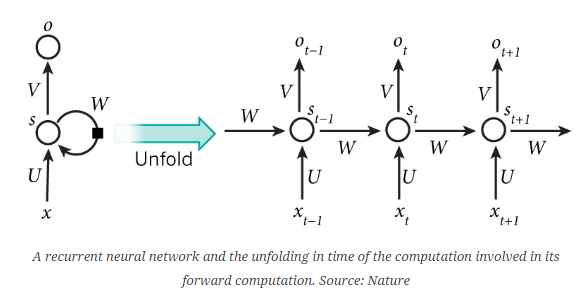
אנו עושים זאת כי המטרה שלנו היא לחזות את המילה הבאה.

1. בניית ה- RNN

דבר ראשון, נרצה לתת הקדמה קצרה על RNN-

היישום של מודל שפה נותן לנו 2 דברים- ראשית, הוא מאפשר לנו לייצר משפטים שרירותיים על סמך הסבירות שלהם להתרחש בעולם האמיתי והוא נותן לנו מידה מסוימת נכונות דקדוקית וסמנטית. מודל זה לרוב משתמש כחלק ממערכת תרגום מכונה. שנית, מודל שפה מאפשר לנו לייצר טקסט חדש לגמרי. אם היינו מאמנים את מודל השפה שלנו על טקסט שייקספירי, היינו יכולים לייצר טקסט שייקספירי מדומה.

הרעיון שמאחורי הRNN, הוא לעשות שימוש במידע רציף. ברשת נורירונים מסורתית יותר, אנו מניחים שכל התשומות והתפוקות הם בת"ל אחד מהשני. אבל רעיון זה לא יכול לעבוד עבור כל משימה, בעיקר לא עבור שפות ללמידת שפה. בשביל זה יותר עדיף לבצע את ה-RNN. יש בו את המילה recurrent כי אנו רוצים לבצע את אותה משימה עבור כל אמנט של רצף. RNN הוא בעל "זכרון" אשר לוכד מידע על מה שחושב עד כה, אך בפועל הוא כן מוגבל לחשוב רק כמה צעדים אחורה.



התרשים הנ"ל מראה איך RNN מתפרש/מתפתח להיות רשת מלאה. על ידי תהליך זה (Unfold) אנו כותבים את הרשת עבור רצף שלם. לדוגמא, אם הרצף שאנו מתייחסים אליו הוא משפט בעל 5 מילים, הרשת תתפתח לרשת של 5 שכבות, מילה עבור כל שכבה. הנוסחאות ששולטות בחישוב שמתבצע בתוך הRNN הם כאלו:

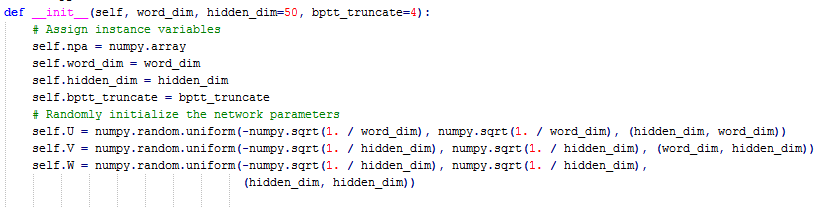
* Xt-הקלט בזמן t.
* St- השכבה הנסתרת בזמן t. זה "הזכרון" של הרשת- הוא מחזיק אינפורמציה על מה שקרה בכל השלבים לפני. הוא מחושב לפי השכבה הנסתרת בשלב הקודם והקלט האחרון. , פונקציה f לרוב היא לא לינארית.
* at- הפלט בשלב t, הוא מחושב לפי St.

עבור כל צעד, RNN מחזיק תמיד 3 פרמטרים U,V,W- זה מראה לנו שאנו מבצעים את אותה משימה כל פעם מחדש. מוריד באופן משמעותי את כמות המשתנים שאני צריכה להחזיק.

עבור המודל שאנו צריכים- מודל שפה ויצירת טקסט, הקלט הוא רצף של מילים (משפט) והפלט הוא רצף של מילים שאנו צפינו.

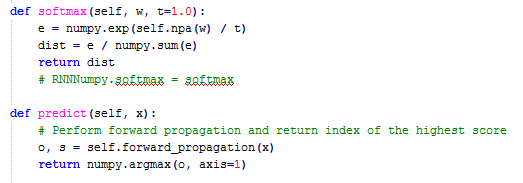
השלבים לבניית ה-RNN:

* 1. כדי שיהיה לנו נוח לבצע כפל מטריצות, אנו מייצגים כל מילה כוקטור באורך של גודל המילון. במקרה שלנו בגודל 2000 וקטור מורכב כולו מ-0 ורק המקום בו האינדקס של המילה נמצא, שם נשים 1.  
     הפלט של הרשת יהיה במבנה דומה, בכל מקום בוקטור יהיה את ההסתברות של המילה באינדקס מסויים להופיע מיד אחרי המילה שנכנסה בפלט.
  2. המימדים של הוקטורים והמטריצות שלנו יהיו כאלו:
     1. – כאשר 2000 גודל המילון שלנו
     2. - גודל השכבה החבויה הוא 50
  3. איתחול- כאשר מאתחלים את המטריצות U, V, W אסור לאתחל את הכל באפסים, זה עלול לגרום לכך שהחישוב בכל שכבה יהיה סימטרי ולכן יש צורך באיתחול רנדומלי של כל מטריצה.

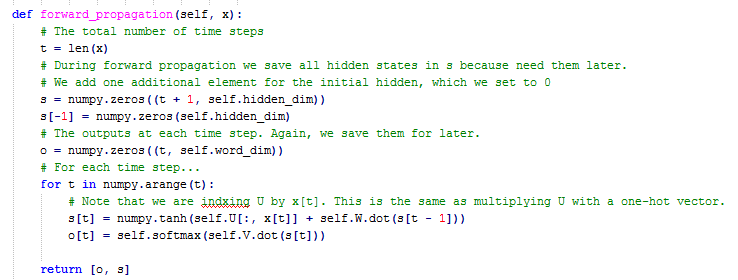
נעשו מחקרים רבים בנושא איתחול המטריצות מכיוון שיש לזה חלק משפיע מאוד על התוצאות. המחקרים גילו שאחת הדרכים הטובות לאיתחול המשנים היא בצורה רנדומלית כך שהתוצאה תהיה בטווח כך שn הוא מספר הכניסות מהשכבה הקודמת.  


1. Forward propagation- חיזוי הסתברות המילים
   1. במהלך הforward propagation, אנו שומרים את כל המצבים הנסתרים במשתנה לשימוש מאוחר יותר.
   2. את S נגדיר כמטריצה בגודל המילון +1 על כמות השכבות החבויות שלנו, נגדיר את זה כך בשביל כל הרצה. נאתחל אותו ב-0.
   3. את O, שהוא הפלט נגדיר כמטריצה בגודל מספר המילים במילון על מספר המילים במילון, כך שעבור כל הרצה, נחזה לכל מילה מה הסיכוי של כל מילה להופיע אחריה.
   4. כדי לחשב את S, אנו צריכים פונקציה לא לינארית, הפונקציה f שלנו היא tanh.

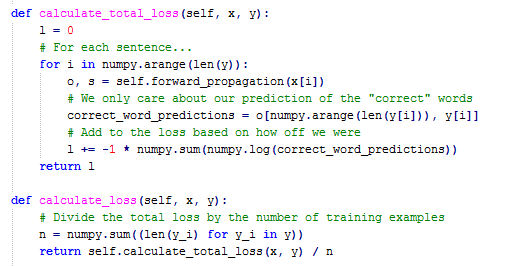
* הפונקציה tanh היא פונקציה מתמטית שמתאימה גם לביצוע מניפולציה סימלית ומספרית.
* אוטומטית הופך להיות tanh[z].
* עבור ארגומנטים ספציפים, tanh הופך אוטומטית להיות אותו ערך.
* ניתן להעריך באופן שרירותי את הדיוק המספרי שלו.
* אחד הייתרונות הגדולים שלו זה שהוא יכול בצורה אוטומטית לבצע thread על ליסטים.
  1. כדי לחשב את O, נשתמש בפונקצית softmax.
     1. נשתמש בפונקציה הזו כדי להקטין את ההשפעה של ערכים קיצוניים או חריגים בנתונים מבלי להסיר אותם.
     2. אנו רוצים לכלול במערך את הנתונים הקיצוניים אך להוריד את המשמעות שלהם.



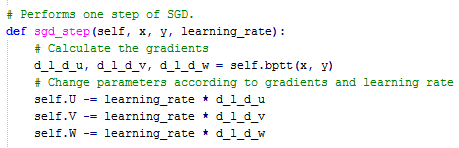
* 1. כל וקטור Ot הוא וקטור הסתברויות המייצג את המילים במילון שלנו, אך כאשר אנו מעריכים את המודל שלנו, כל מה שאנחנו רוצים זה את המילה הבאה עם ההסתברות הגבוהה ביותר. פונקציה שעושה זו היא פונקצית predict.
  2. -Predict הפונקציה קוראת לforward\_propagation, ומחזירה את הערך המקסימלי מהוקטור של הפלט – O.

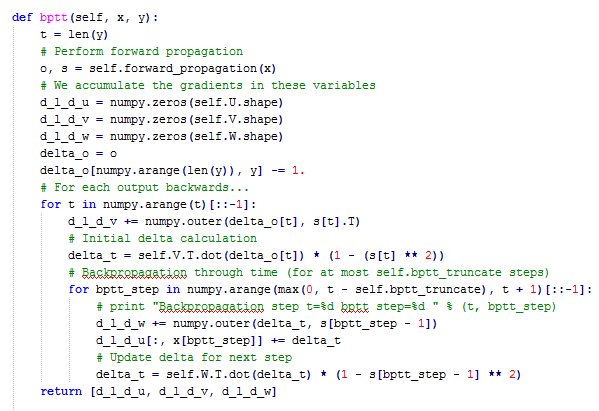


1. Calculating the loss - כדי לאמן את הרשת שלנו, אנו צריכים דרך למדוד את הטעויות של המודל שלנו. המטרה שלנו היא למצוא את הפרמטרים U, V, W אשר ימזערו את את ההפסד עבור ה- training data
   1. הפונקציה משתמשת בנוסחא אשר עוברת על כל דוגמאות האימון שלנו ומוסיפה להם את ההפסד בהתבסס על כמה החיזוי שלנו היה לא מדוייק. כלומר, כמה המילה הנכונה, הייתה רחוקה מהמילה שחזינו.
   2. הפונקציה הראשית היא calculate\_loss תפקידה:
      1. מאתחלת את N במספר דוגמאות האימון שלנו.
      2. קוראת לפונקציה calculate\_total\_loss ומחלקת בN.
   3. הפונקציה calculate\_total\_loss –
      1. עוברת על כל משפט באימון שלנו.
      2. הדבר היחיד שמעניין זה החיזוי שלנו למילה "הנכונה".
      3. נוסיף לתוצאה את המרחק של החיזוי מהמטרה.



1. Training the RNN with SGD
   1. הדרך הכי נפוצה למצוא את הפרמטרים U,V,W על מנת שנצמצם את ההפסד עבור ה- training data שלנו, הוא בעזרת SGD- stochastic gradient descent.
   2. הרעיון מאחורי SGD, הוא לבצע איטרציות על דוגמאות האימון שלנו ועבור כל איטרציה נדחוף את הפרמטרים לכיוון שמפחית את השגיאה, כיוונים אלו ניתנים על ידי שיפוע ההפסד.
   3. SGD צריך גם שיעור למידה, אשר מגדיר כמה גדול הצעד שנרצה לעשות בכל איטרציה.
   4. שיטת אופטימיזציה פופולרית לא רק עבור רשתות נורירונים אלא גם עבור אלגוריתמים למערכות למידה אחרות.
   5. קיימים מחקרים רבים שעבדו על דרכים לייעל את הפונקציה.
   6. מכיוון שהפרמטרים משותפים על ידי כל הצעדים ברשת, הSGD בכל פלט תלוי לא רק בחישוב של ההרצה הזו, אלא גם בהרצות הקודמות.
   7. המימוש של SGD נעשה ב-2 שלבים.
      1. Sdg\_step- מחשב את הgradient ומבצע את העדכון עבור אצווה אחת.
         * מחשב את הgradient בעזרת BPTT (הסבר בהמשך)
         * משנה את הפרמטרים של U, V, W בהתאם לשיעור הלמידה ושיפוע ההפסד.
      2. Train\_with\_sgd - מבצע לולאה חיצונית שמבצעת איטרציה על סט האימון ומשנה בהתאם את שיעור הלמידה.
         * משתנים:
           1. המודל- שכפול של מודל הRNN.
           2. קיים סט אימון המידע ותגיות של אימון המידע.
           3. שיעור הלמידה מאתחלים אותו לSDG.
           4. ועוד משתנים
         * אנו מבצעים מעקב על הפסדים.
         * מבצעים איטרציות כדי לחשב את ההפסד בצורה אופטימלית ולבצע שינויים בשיעור הלמידה אם ההפסד עולה.  
           בתוך האיטרציה נבצע איטרציה נוספת על כל סט אימון בו נקרא ל sgd\_step.



1. backpropagation through time (BPTT)
   1. כדי לחשב את ה- SGD, נשתמש ב BPTT.
   2. הקלט הוא דוגמאת אימון ומחזיר את gradient-
   3. שלבי האלגוריתם:
      1. מבצעים forward\_propagation.
      2. נאתחל את משתני ה- gradient- באפסים, ודלתא.
      3. ונעדכן אותם בכל ריצה.
      4. בסוף נחזיר את ה- gradient.
2. Generating Text – יצירת המשפטים
   1. לאחר ביצוע השלבים הקודמים, קיבלנו מודל אשר יכול לייצר משפטים חדשים עבורנו.
      1. תחילה, נציב בתחילת המשפט את תו "התחל משפט".

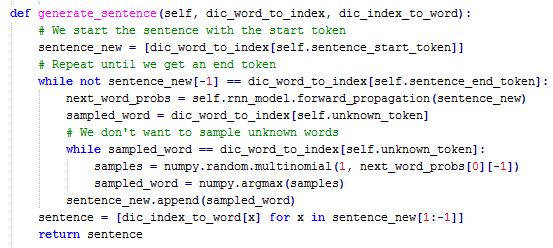


* + 1. נשרשר מילים חדשות למשפט, עד אשר נקבל תו "סיום משפט".

שרשור המילים החדשות מתבצע בעזרת לולאה, אשר בכל איטרציה שולחת את המשפט שנוצר עד כה לפונקציית forward\_propagation, אשר מחזירה, כזכור, וקטור עם משקלי הסיכויים למילה הבאה במשפט.

מכיוון שמודל זה תומך ב-2,000 מילים, ישנו מצב שנקבל מילה unknown\_token המייצגת מילה שלא נמצאת במילון. במצב זה, אנו נבחר מילה חדשה, עד אשר נקבל מילה השונה מ- unknown\_token.

לבסוף, לאחר שקיבלנו תו "סיום משפט" נחזיר את המשפט החדש.



את התוצאה אנחנו שומרים בקובץ חיצוני.

אלו דוגמאות למשפטים שקבלנו:

what the fuck is saying you.

what the fuck is wrong jordan legal got money.

get the fuck you.

want you look.

by the lollipop club week trust years something.

i know i will to get you fucked.

oh my god i had that had out right guy i cant wanted the questions clients cracked must they bunch do.

the only must alone of his pops and for it goes mom talk to the phon a u expensiv.

what do you gon to take it in youll maybe right jordan italian can think you all the time in both the architect shame justice thats jordan be so.

150 faggo my door and let that something okay wants to take a fucking picked.

כאשר מסתכלים על המשפטים שנוצרו, ניתן לשים לב כי המודל יצר משפטים טובים.

ניתן לראות כי ישנם שגיאות דקדוקיות.

ייתכן שזה קרה משום שלא אימנו את המודל מספיק זמן, כלומר יש לבצע עוד חזרות.

סיבה נוספת, ואנו חושבים בעלת משקל גבוהה יותר, היא שמודל RNN מסוגל ללמוד תלות בין מילים במרחק מספר צעדים זה מזה.

בחלק מסקנות ציינו מספר דרכים אשר יכולים לשפר את המודל.

בדיקת הפרויקט:

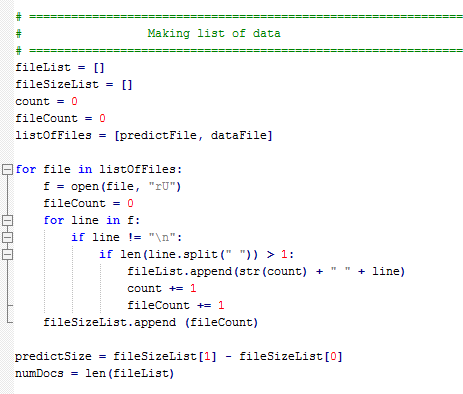
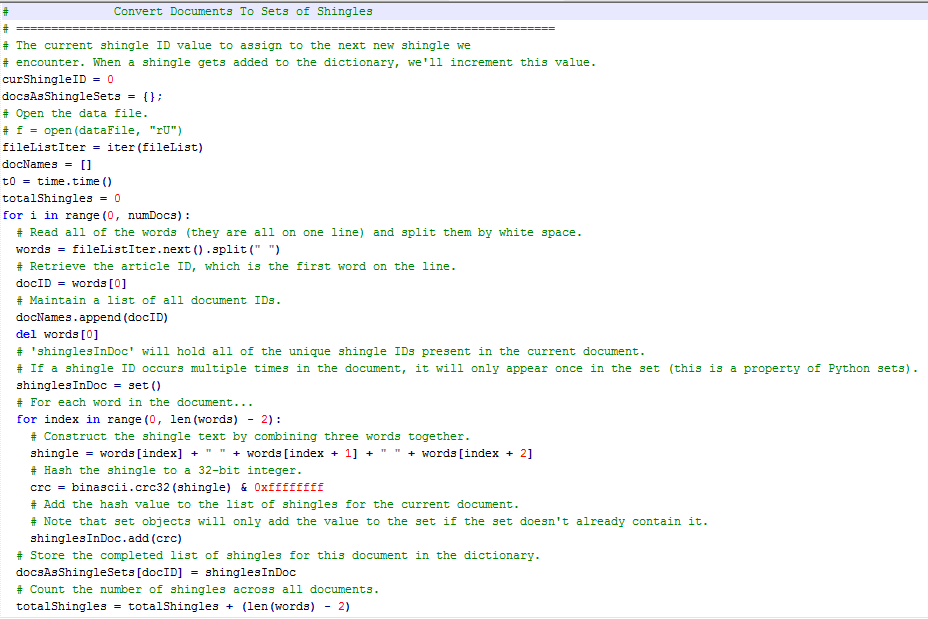
על מנת שנדע שהמודל שלנו אכן מצליח לייצר משפטים שתואמים את הטקסט שנתנו למודל כקלט, היינו צריכים לבצע בדיקות דמיון על הפלט.

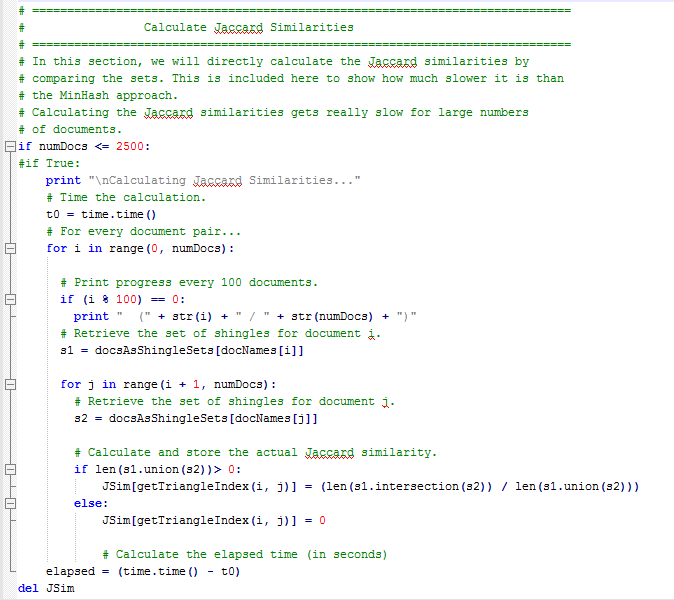
בחרנו לבצע את ההשוואה בעזרת גישת MinHash.

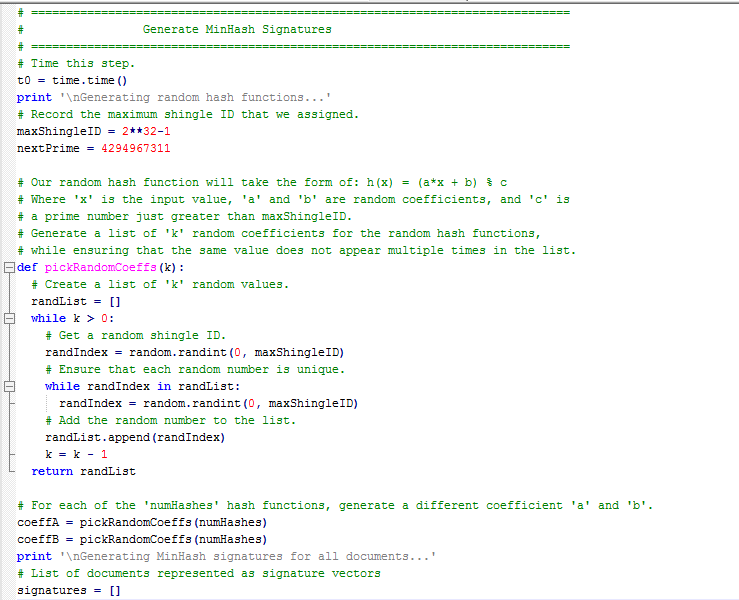
כדי שנוכל לעבוד באלגוריתמים של MinHasah בצורה הטובה ביותר היה צורך בניקוי של הטקסט ועיבודו.

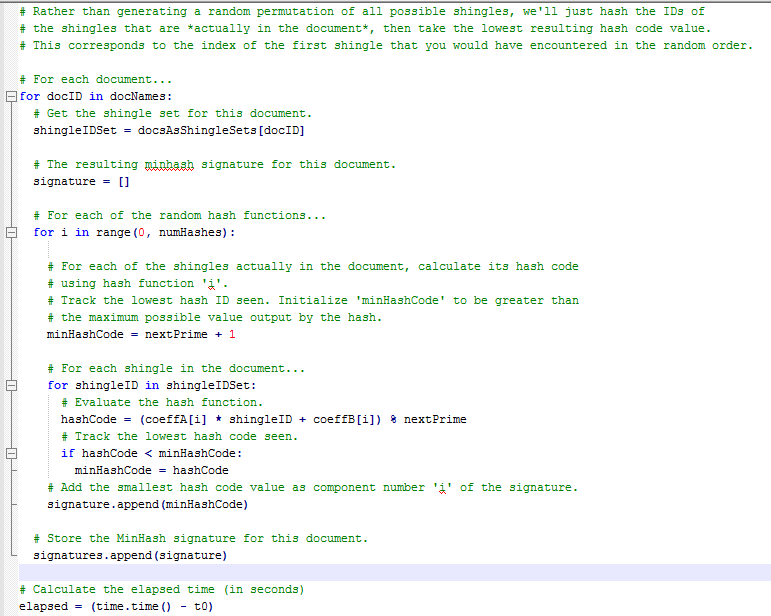
* כל סיומת של נקודה,סימן קריאה, סימן שאלה וסוגריים החלפנו "/n".

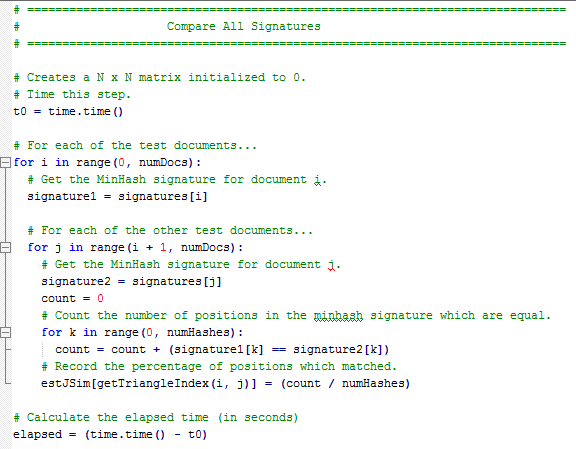
האלגוריתם שעשינו הוא:

1. נשלוף את הנתונים מ- 2 קבצים. קובץ ראשון הקלט של המודל וקובץ שני זה הפלט של המודל תוצאת החיזוי.
2. נייצר רשימה של שורות
   1. ננקה שורות ריקות ושורות בעלות מילה אחת.
3. נייצג כל מסמך כסט של "shingles"
   1. ניקח כל משפט ונחלק אותו לשלושה מילים רצופות. לדוגמא המשפט-
      1. **get out of the fucking car**
      2. get out of
      3. out of the
      4. of the fucking
      5. the fucking car
   2. בעזרת hash נייצג כל מחרוזת כinteger.
   3. כך אנו שומרים על מבנה המסמך – יותר מאשר אם היינו מחלקים למילים בודדות.
   4. הטכניקה נקראת shingling וכל מחרוזת נקראת shingle.
4. בעזרת jaccard similarity, נוכל להשוות את הshingles של מסמך הקלט מול מסמך הפלט (שיטה זו טובה כאשר המסמך קטן יחסית).
   1. נחשב זאת בצורה הבאה: כאשר אנו מסתכלים על שני מסמכים, הקלט והפלט, נבדוק כמה משפטים מוגדים כדומים.
   2. נחלק את מספר המשפטים הדומים באיחוד שני המשפטים, כך שמשפטים דומים לא נספרים פעמיים.

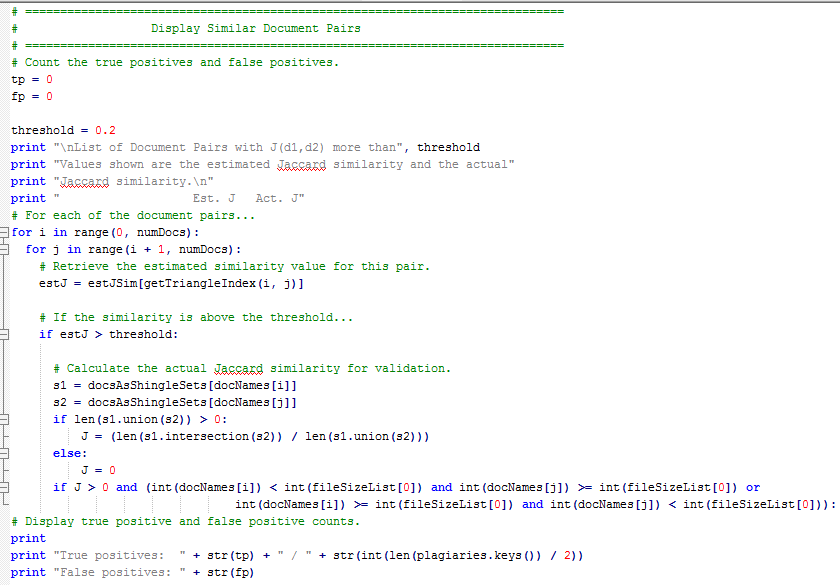


1. נחשב את חתימת ה- MinHash של כל משפט.
   1. נממש את זאת בעזרת פונקציית hash רנדומאלית.
   2. זה עוזר לנו לא לחשב במפורשות פרמוטציות אקראיות של כל מזהי ה- shingles.



1. נשווה את כל חתימות ה- MinHash זה לזה.
   1. נשווה את החתימות על ידי ספירת מספר המרכיבים בהם החתימות שוות.
   2. נחלק את מספר המרכיבים התואמים באורך החתימה של הhash כדי לקבל את ערך ההתאמה.
   3. נספור את החתימות אשר מידת ההתאמה גדולה מהסף שקבענו – threshold = 0.2

1. אנו משווים כל שורה מול כל שורה על מנת לבדוק האם אכן יש שורות זהות בין הטקסט המקורי לטקסט שחזינו.
   1. אם הניקוד של דמיון עולה מעל משתנה סף שהגדרנו כ-0.2, נגדיר את השורות כדומות.



מסקנות והמשך מחקר:

לאחר שהרצנו את החלק האחרון, קיבלנו כפלט את תוצאות המודל.

חלק מהתצאות שיצאו:

|  |  |  |  |
| --- | --- | --- | --- |
| **מדד דמיון J** | **actual J** | **אנחנו חזינו** | **קיים בטקסט המקורי** |
| 0.7 | 0.33 | what the fuck is saying you. | what the fuck is ej entertainment |
| 0.4 | 0.33 | what the fuck is wrong jordan legal got money. | what the fuck is wrong with you |
| 0.3 | 0.33 | get the fuck you. | get the fuck out |
| 0.2 | 0.17 | want you look. | no i want you to cooperate |
| 0.2 | 0.08 | by the lollipop club week trust years something. | oh he wa at the lollipop club for sure |
| 0.2 | 0.07 | i know i will to get you fucked. | did you try to get you broker's license at one time |
| 0.2 | 0.05 | oh my god i had that had out right guy i cant wanted the questions clients cracked must they bunch do. | oh my god baby |
| 0.2 | 0.04 | the only must alone of his pops and for it goes mom talk to the phon a u expensiv. | i mean you could talk to the sec |
| 0.2 | 0.04 | what do you gon to take it in youll maybe right jordan italian can think you all the time in both the architect shame justice thats jordan be so. | what do you mean |
| 0.2 | 0.04 | 150 faggo my door and let that something okay wants to take a fucking picked. | did i hear that right were you tryin to take a stab at wall street |

\*actual J - חיתוך המשפטים בשני הקבצים חלקי האיחוד של המשפטים בשני הקבצים

ניתן לראות כי יש משפטים אשר דומים מאוד למשפטים המקוריים וישנם משפטים שפחות.

כאשר עבדנו על שיפור המודל, ניסינו להוריד סימני פיסוק או סיומת של שורות (/n) ובדקנו אם התבצע שיפור, דבר אשר אכן תרם.

לדעתנו אם נמשיך לעבוד ולחקור עוד שיטות נוכל לייצר מודל מדוייק יותר.

שיטות שבדקנו ובמידה ונרצה להמשיך את הפרויקט נשתמש הם-

1. תיקון שגיאות כתיב. להשוות מילים מהטקסט למילים מהמילון האנגלי.
2. לבדוק עם או בלי סימני פיסוק.
3. הפרדה לפי תחביר- כלומר, הפרדה של פועל, שם עצם, שם תואר וכו'.
4. קלט ארוך יותר.
5. השתמשנו במילים השכיחות ביותר. שמרנו 2000 מילים, ניתן לשנות את המספר הזה ולבדוק האם זה עוזר.
6. עם ובלי /n/r.

בפרויקט זה, ניתן להשתמש במגוון תחומים, ולשנות את הקלט לפי הצורך, כמו טקסט מדעי, מאמרים וכו'.