**שלב 1:**

הסבר כללי על הקוד:

פרטים על class LM\_Trigram :

* הבנאי מקבל שני פרמטרים אחד שזה הקורפוס והשני הסוג שזה plenary או committe.
* בתוך הבנאי:
  + שומרים את שני הפרמטרים שקבלנו
  + עושים את unigrams, bigrams, trigrams (דרך המתודה get\_n\_grams(n) נפרט בהמשך) ושומרים את אלה בהתאים במשתנים unigrams\_set, bigrams\_set, trigrams\_set .
  + מחשבים כמה תוקנים יש בתוך הקורפוס שומרים את המספר במשתנה בשם length\_of\_corpus .
  + ויש לנו רשימה של 3 הλות.
* מתודה calculate\_prob\_of\_sentence מקבלתי שני פרמטרים אחד שהוא מחרוזת של תוקנים והפרמטר השני הוא מחרוזת המתארת סוג ההחלקה ( לינארית או לאלפס) והוא
* בתוך המתודה:
  + לפי הפרמטר שמקבלים עושים החלקה בהתאים:
    - החלקת לאלפס: מוספים 1 ומחלקים בגודל הunigrams
    - משתמשים בלוג כדי שלא יהי לנו overflow .
    - ש V1 הוא מספר המופעים של הטוקן הראשון
    - ש V2 הוא מספר המופעים של שני התוקנים האחרונים
    - ש V3 הוא מספר המופעים של שלוש התוקנים
    - לפי אלה אנחנו את ה MLE עם החלקת לאלפס.
  + ההחלקה הלינארית היא רק שאנחנו P(w1, w2, w3) עושים סכום עבור unigrams, bigrams and trigrams עם הכפלה ב λ ות כך שסכום ה λ ות יהי שווה ל 1 ולפלאס לunigram .
* מתודה generate\_next\_token מקבלת פרמטר אחד שהוא מחרוזת שמכילה משפט
  + בתוך המתודה:
    - עוברים על כל התוקנים השונים
    - מחשבים את ההסתברות של התוקן הבא במשפט בהחלקתליארית כך:
      * .
* מתודה duplications מקבלת עד שלושה פרמטרים שהם תוקנים.
  + מחזירה את כמה התוקנים ב self.unigrams and self.bigrams and self.trigrams .
* מתודה get\_n\_grams מקבלת פרמטר אחד שהוא מספר ה ngrams כדי לבנות אותו.
* מתודה probability\_of\_tokens היא עוזרת לנו כדי למנוע שכפול מחזירה את ההסתברות המותנית של P(Wn|Wn-1Wn-2) .
* ההחלטה בפונקציה זו להשתמש generate\_next\_token בחלקת לינארית היא בגלל שכך נוכל למקסם ביטוי אחד וכך לא נקבל כמה מילים שיש להם אותה הסתברות, וכדי שנוכל לקבל משפטים טובים אנחנו נוכל לבחור ב λ עובר ה trigrams גדול כך ש unigrams לא ישפיעו הרבה כדי שלא נקבל רק סמני פיסוק.

1. בחירת ה λ:

* הבחירה של ה λ הייתה לפי המודל בגלל שאנחנו מעוניינים במודל שהוא trigrams אזי אנחנו נותנים ערך יותר גבוה ל trigrams ואז ל bigrams כי הוא קרוב למודל שלנו ולכן λ1 < λ2 < λ3 .
* ההחלטה הזו גם מושפעת מזה כאשר אנחנו רוצים להשלים משפט אז אנחנו לא רוצים שהכי נפוץ יבחר תמיד ולכן עדיף לתת משקל קטן ל unigrams כדי שיש לנו חזוי יותר טוב.
* הערכים היו אצלנו למשל 0.7 0.2 0.1

1. משפטים עם פחות מ 3 טוקנים:
   * כדי לחשב את ההסתברות בצורה נכונה אזי הוספנו לתחילת המשפט <s> שזה מעיד על תחילת משפט.
   * עוזר לנו לסננן דברים לא רלוונטיים כמו סמני פיסוק ורק מתייחס לדברים שממש באים בתחילת המפשט.
   * משפטים שיש בהם רק טוקן אחד אנחנו נוסף שני <s> כי ב linear interpolation זה ככה מסתדר לא כאשר מוספים סימון לסוף המשפט כי אז לא נחשב את הbigrams and unigrams בצורה הנכונה.

השתמשנו בקורפוס שנתון לנו (כלומר לא לפי מה יש לנו מתרגיל בית קודם).

**שלב 2:**

1. פונקציה get\_k\_n\_collocations :
   * היא חלק מהמחלקה שבנינו כי כך נוכל לגשת בצורה נוחה יותר לקורפוס עצמו לחשב עבור כל קורפוס את קולוקציות שלו.
   * הקלט הוא n ו k כאשר n הוא אורך הקולקציה וk הוא מספר הקולוקציות.
   * והפלט הוא k הקולוקציות הכי נפוצות.
     1. מחשבים את ה PMI עבור n מילים .
     2. לוקחים את ה k הגדולים ביותר.
2. פונקציה get\_PMI:
   * כאשר N הוא גודל הקורפוס (מספר התוקינים)
   * Length\_ngrams מספר המשפטים שבאורך n .
3. פונקציה insert\_collocations :
   * המטרה שלה לשמור את הפלט של פונקציה get\_k\_n\_collocations מ 2 עד 4grams בתוך קובץ טקסט.
   * מקבלת פרמטר את שני המודלים כרשימה.

**שלב 3:**

1. הפונקציה insert\_sentence :
   * המטרה שלה לשמור את הפלט של הפונקציה generate\_next\_token כאשר עושים את שלב 3
   * מקבלת 4 פרמטרים:
     1. את שני המודלים.
     2. את המשפט שצריך להשלים.
     3. את שני המשפטים שהושלמו בהתאמה למודל.
     4. ואת הטוקנים שהשתמשנו בהם כדי להשלים.
2. השתמשנו בהחלקה לינארית.

**שלב 4:**

1. האם שמתם לב להבדל משמעותי בין שני המודלים שבניתם? האם לרוב קיבלתם את אותן תוצאות בשניהם או תוצאות שונות? הסבירו מדוע לדעתכם זה קרה

לרוב קיבלנו תוצאות שונות:

* + וזה מה שצפינו לקבל משום ששני הקורפוסים עוסקים בתחומים שונים , וגם יש להם גדלים שונים כך שבמודל אחד יש לנו לעמות שבמודל השני יש ולכן כאשר אנחנו נעשה מודל trigrams עבור זה שיש לו רק אזי יהיו הרבה שהם OOV יותר מהמודל השני.
  + כפי שצוין בהרצאות ש ngrams הוא תלוי בקורפוס ככל שהקורפוס גדול כך המודל יהי טוב יותר, ולכן אנחנו לא רוצים במודלים כאלה שיהיו דלילים ולא נקבל הרבה OOV כדי שנוכל לקבל תוצאות טובות.
  + כך שגם הנושאים בין הקורפוסים שונים ולכן גם זה משפיע.

1. האם הקולוקציות הנפוצות ביותר בכל קורפוס יכולות לספר לנו משהו על התוכן והנושאים בהם הקורפוס עוסק? האם הופתעתם מהתוצאות שהתקבלו או שהן תאמו לציפיות שלכם? הסבירו.
   * התוצאות לא מפתיעות, מדד בודק כמה פעמים טוקנים הופיעו יחד ביחס לכמות הפעמים שהופיעו באופן כללי כך שאם הפעם היחידה שהן הופיעו זה היה יחד הן יקבלו ציון PMI גבוהה, מדד ה PMI רגיש מאוד לשילוב נדיר של מילים באופן כללי.
   * ולכן רוב הפעמים אנחנו נקבל משפטים שהופיעו רק פעם אחד ולכן הם לא מספרים על מה מדובר בקורפוסים.
   * היינו רוצים למשל בקולקציות בגודל 2 לקבל "יושב ראש" , "חבר כנסת" .. מאשר לקבל "ולהבטיח שהתכנים", היינו רוצים לקבל שמופיעות הרבה פעמים ("יושב ראש" הופיעה 159 פעמים) לעומת "ולהבטיח שהתכנים" שהופיעה רק פעם אחת , אבל כל טוקן בנפרד "יושב", "ראש" הופיעו אלפי פעמים לכן מדד ה PMI שלהם שלילי לעומת ש "ולהבטיח" , "שהתכנים" הופיעו רק פעם אחת ואז מדד ה PMI שלה חיובי. וזאת הבעיה של מדד ה PMI כך שהוא מנפח הערך של אירועים הנדירים.
2. לרוב המשפטים לפחות באחד מהמודלים קיבלנו משפט הגיוני. וזה נותן איזשהו סימן שבחירת פרוטוקולים כדי ליצר את הקורפוס בחירה טובה. כך שמודבר בהרבה נושאים שונים ואז יש צפיה לתפיסת תופעות בשפה יותר. מצד שני על סמך התוצאות אנחנו מסיקים שמודל ה Trigram מודל טוב עבור השפה העברית .
3. היינו מצפים תוצאות יותר גרועות, כך שבשפה העברית משפטים שלמים בדרך כלל מרוכבים מיותר משני טוקנים. במיוחד עבור המקרים שבהם הטוקן הבא שאנחנו רוצים לחזות בא אחרי טוקנים כמן "הוא"(או כל כינוי) , "את" , "עוד", "לא", שלא נותנות ידע על מה יכול להיות הטוקן הבא. האינטואיציה היא שמודל הTrigram יש לו יותר "ידע" על המשפט מהמידע שיש למודל ה bigram. במונחים הסתברותיים הערך שנותן מודל הTrigram ל יותר גדול מהערך שנותן מודל ה bigram ל לכן ה perplexity של Trigram בקורפוס שלנו יותר קטן מ perplexity של bigram.