דו"ח תרגיל בית 2

מודלי שפה – n-grams

**שלב 1**

**שלב 1.1:**

הפונקציה ' calculate\_prob\_of\_sentence' מומשה. היא מקבלת כקלט משפט, ומחזירה את לוג ההסתברות של המשפט. הפונקציה מחשבת את לוג ההסתברות בעזרת נוסחת ה- Trigram מההרצאה, כאשר ההסתברויות מחושבות באמצעות MLE עם החלקת לפלס כנדרש.

בהינתן המשפט , הפונקציה קודם כל מוסיפה למשפט שני טוקני דמה , כך שהמשפט הופך להיות , ומחשבת את ההסתברות שלה באמצעות הנוסחה הבאה:

כאשר:

כך ש-

המקדמים נבחרו להיות 0.9, 0.0999, 0.0001 בהתאמה. המקדמים נבחרו על סמך ניסוי שהשווה שלשות שונות של מקדמים באחוז הצלחתם בניבוי טוקן חסר במשפט. הניסוי כלל 500 משפטים שנבחרו באופן אקראי מקורפוס הוועדות, ובכל אחד מהם 10% מהטוקנים מוסכו. הניבוי של הטוקנים הממוסכים התבצע ע"י הפונקציה 'generate\_next\_token' שמסתמכת בניבוי שלה על הסתברות המשפט שמחושב ע"י הקריאה ל- ' calculate\_prob\_of\_sentence'. הצעד הזה של בחירת המקדמים נעשה בסוף (בהתחלה השתמשנו במקדמים שרירותיים), ובחירת המקדמים שישתתפו בניסוי נעשתה על סמך אבחנות שהתקבלו במהלך כתיבת הקוד, דבר שצמצם את טווח האופציות של כל מקדם. ניתן לסכם את האבחנות בכך שהמקדם של ה- unigrams צריך להיות קטן מ- 0.05, ושהמקדמים (החל ממקדם ה- trigrams) צריכים להיות גדולים אחד מהשני בהפרש משמעותי. האבחנה הראשונה התקבלה בשל כך ששמנו לב שמקדם גבוה עבור ה- unigrams נתן משקל יתר לטוקנים נפוצים כמו "," ו- "אני", גם כשהמיקום שלהם לא תואם מבחינת תחביר ושפה, ולא מבחינת הסתברות של משפט (למשל, האלגוריתם ניבא את הטוקן "," אחרי הטוקן "עמיר" למרות שהופעת הטוקן "פרץ" אחרי "עמיר" יותר שכיחה בקורפוס). לגבי האבחנה השנייה, היה אפשר לשים לב לכך שיכולת הניבוי של ה- bigrams יותר טובה ביחס ל- unigrams ושל ה- trigrams יותר טובה משניהם. המשקל הגבוה של ה- bigrams יחסית ל- unigram ניתן על סמך כך שהרבה צירופי טוקנים באים דווקא כזוגות, דוגמת: "חוות דעת", "אמר :", ושמות משפחות של אנשים מפורסמים שבאים אחרי שמות פרטיים. מקדם גבוה ל- bigrams נתן לאלגוריתם להצליח לנבא את המילה השנייה בצירופים מסוג זה. מצד שני, מקדם גדול מדי ל- bigrams היה פוגע בצירופים הפחות נפוצים, במיוחד כאשר הטוקן השני פחות מזווג עם הטוקן הראשון מאשר טוקן אחר. למשל, בשל שכיחות הצירוף "במגזר הערבי" בקורפוס, האלגוריתם נכשל בלנבא את המילה "הפרטי" כהמשך למשפט " והן עם העובדים במגזר [\*]", מכאן קיבלנו מסקנה שקונטקסט יותר גדול נדרש על מנת להבדיל בין שני המקרים ("במגזר הערבי" ו- "במגזר הפרטי). בנוסף לכך, ובשל העובדה שמספר המופעים של trigrams קטן משמעותית ממספר ה- bigrams (למשל, "העובדים במגזר הפרטי" לעומת "במגזר הפרטי"), הסקנו שהמשקל של ה- trigrams חשוב שיהיה גבוה משמעותית מה- bigrams.

**שלב 1.2:**

הפונקציה ' generate\_next\_token' מומשה. היא מקבלת כקלט מחרוזת (רצף של טוקנים), ומחזירה tuple בעל שני אלמנטים: הטוקן בעל ההסתברות הכי גבוהה שיהיה ההמשך של המחרוזת, ולוג ההסתברות שלו. לשם כך, הפונקציה עוברת על כל הטוקנים השונים (או באופן יותר מדויק, על ה- unigrams, מכיוון ש- unigrams וטוקנים שקולים במקרה שלנו), מוסיפה את הטוקן למחרוזת ושולחת את התוצאה של החיבור לפונקציה ' calculate\_prob\_of\_sentence ' שבתורה מחזירה את לוג ההסתברות של המשפט. הפונקציה תחזיר את הטוקין בעל לוג ההסתברות הגבוה ביותר ביחד עם לוג ההסתברות שלו.

**שלב 2**

**שלב 2.1:**

הפונקציה get\_k\_n\_t\_collocations, הפונקציה מחזירה אוסף של קולקציות העונות על הדרישות הבאות שמתקבלות כקלט: מספרם הוא k, אורך כל אחת מהן הוא n, ומספר המופעים של כל אחת לא פחות מ- t בקורפוס מסויים, שהוא pandas dataframe. כמו כן, הפונקציה מקבלת כקלט את סוג המדד שלפיו מחליטים איזה k מבין כל הקולקציות העונות על הדרישות צריך לבחור.

חילוץ הקולקציות מתבצע ע"י לעבור על כל המשפטים בקורפוס, ועבור כל אחת שאורכה לפחות n, הוא מחלץ את כל הקולקציות באורך n השונות (כל המחרוזות של טוקנים רצופים בגודל n).

הפונקציה אח"כ מסננת את הקולקציות ומשאירה רק אלה שמספר הופעתם בכלל הפרוטוקולים הוא לפחות t.

הפונקציה אח"כ מסדרת את הקולקציות מהכי נפוצה לפחות לפי המדד הנבחר. שני המדדים הם "Frequency" ו- "Tfidf", כאשר הראשון בוחר לפי כמות ההופעות, והשני לפי הנוסחה הנתונה. ומחזירה את ה- k קולקציות הראשונות בסדר. במקרה שאין k קולקציות כאלה, הפונקציה מחזירה את כל הקולקציות שהתקבלו.

לשם חישוב k- הקולקציות הכי נפוצות לפי מדד TF-IDF. הפונקציה משתמשת בשני מילונים של מילונים (מילונים שה- values שלהם מילונים). כאשר המילון הראשון משמש לשמירת עבור כל קולקציה את הפרוטוקולים (שמיוצגים ע"י מספרם) בהם הופיעה, ועבור כל אחד את כמות הופעותיו (כלומר המפתחות החיצוניות הם קולקציות, הערכים החיצוניים\ המפתחות הפנימיים הם מספרי פרוטוקולים, והערכים הפנימיים הם כמות ההופעות של אותה קולקציה בפרוטוקול). ובשני נשמרים עבור כל פרוטוקול את הקולקציות שהופיעו בו ואת מספר הופעתם (כלומר המפתחות החיצוניות הם מספרי הפרוטוקולים, הערכים החיצוניים\ המפתחות הפנימיים הם הקולקציות, והערכים הפנימיים הם כמות ההופעות של אותה קולקציה בפרוטוקול). המלון הראשון משמש לחישוב ה- Frequency עבור הקולקציה, ששווה לסכום הופעותיו בפרוטוקולים השונים, כמו כן, משתמשים בו בשליפת כמות הופעות הקולקציה בפרוטוקול מסוים בחישוב ה- tf, ובמספר הפרוטוקולים השונים שבהם הקולקציה מופיעה בחישוב ה- idf. בנוסף המלון משמש בשלב סינון הקולקציות. המלון השני משמש לחישוב הקולקציות השונות שבפרוטוקול מסויים בחישוב ה- tf, ובמספר הפרוטוקולים השונים בחישוב ה- idf.

**שלבים 2.2 – 2.4:**

הקוד שמדפיס את הדרוש נמצא ב- main ומשתמש בפונקציה מהשלב 2.1 לשם כך. הפלט מצורף כקובץ txt כנדרש.

**שלב 3**

**שלב 3.1:**

הפונקציה mask\_tokens\_in\_sentences מומשה. הפונקציה מקבלת כקלט רשימה של משפטים ומספר בטווח (0,100). הפונקציה ממסכת x% מהטוקנים במשפט (מחליפה אותם ב- [\*]). ומחזירה משפט חדש עם הטוקנים הממוסכים. במקרה ש- x% לא מספר שלם היא מקרבת אותו לשלם הכי קרוב, ובשל כך, משפט בעל פחות מ- 5 טוקנים (במקרה של המשפטים שלנו, משפט בעל 4 טוקנים) לא ימוסך אף טוקין בו. את הטוקנים שהיא ממסכת הפונקציה בוחרת באקראי.

**שלב 3.2:**

הקוד מומש ב- main, והפלט נמצא בשני קבצי txt מצורפים כנדרש.

**שלב 3.3:**

הקוד מומש ב- main, והפלט נמצא בקובץ txt המצורף כנדרש.

**שלב 3.4:**

הקוד מומש ב- main. להלן התוצאות שקיבלנו, שנמצאות גם כן בקובץ txt המצורף כנדרש.

הנוסחה שהשתמשנו בה לחישוב ה- Perplexity היא אותה נוסחה שהוזכרה בהרצאה עם השינוי שהשתמשנו רק בטוקנים שהחליפו טוקנים ממוסכים. מתמטית, נסמן ב- s את המשפט, וב- את הטוקנים שהחליפו טוקנים ממוסכים. כמו כן, נסמן ב- s[:i] את החלק במשפט עד הטוקן ה- i (לא כולל). אז הנוסחה היא: