שחר טפלר

תרגיל בית 2 – עיבוד שפות טבעיות

חלק 1 – ניבוי משפטים

השתמשתי במחלקה NgramModel כמחלקה אבסטרקטית שממשת פעולות שונות שמשותפות לכל המודלים.
בנוסף, יצרתי גם את המחלקות: UnigramModel, BigramModel, TrigramModel, לכל מודל בהתאמה.

מחלקת ה-NgramModel נראית כך:

```
def __init__(self, n: int, corpus: Corpus, smoothing_type: str = "Laplace"):
    if smoothing_type != "Laplace" and smoothing_type != "Linear interpolation":
        raise Exception("Invalid smoothing type")

    self.n = n
    self.smoothing_type = smoothing_type
    self.corpus = corpus
    self.vocabulary = Counter()

    self.ngrams = Counter()  # Counter for n-grams
    self.contexts = Counter()  # Counter for (n-1)-grams

    self.build_vocabulary()
```

כאשר המודל מחזיק מילון – זהו בעצם מבנה של Counter (בפייתון) כאשר לכל מילה יש את מספר המופעים שלה בקורפוס. בנוסף, למודל יש Counter עבור כל ה-ngrams בגודל n ועבור כל ה-ngrams בגודל n-1 – כך אנחנו יכולים לחשב את ההסתברות:

$$P(w_k \mid w_{k-N+1..k-1}) = \frac{C(w_{k-N+1..k-1} w_k)}{C(w_{k-N+1..k-1})}$$

על מנת לנבא משפט מסוים, עברתי על כל ה-ngrams בגודל n במשפט, וחישבתי לכל אחד את ההסתברות הנ"ל. זה היה יחסית פשוט בגלל ההכנה מראש באמצעות ה-Counters שבנינו.

את החלקת ה-Laplace ביצעתי באופן די פשוט:

```
if self.smoothing_type == "Laplace":
    return (self.get_count(phrases + [token]) + 1) / (self.get_count(phrases) + len(self.vocabulary))
```

ואת ההחלקה של אינטרפולציה לינארית עשיתי באמצעות חישוב מקדמים באופן הבא:
לפי ה-n של המודל, חליקתי את המקדמים כך שהמשקל של הטריגרם הוא הכי גבוה, של הביגרם הוא השני הכי גבוה, וכך הלאה.

שחר טפלר

חלק 2 – יצירת משפטים

קודם כל הכנסתי טוקנים של התחלה (<b>) ושל סיום (<e>) להתחלה ולסוף של כל משפט, כך שהם גם נוספו למילון של הקורפוס. כאשר אני עוברת על כל המילים במשפט, הכנסתי טוקן התחלה לפני הלולאה וטוקן סיום אחרי הלולאה.

לאחר מכן, יצרתי משתנה ששומר את אורך המשפט המקסימלי בקורפוס, ומעדכן אותו לאחר כל הוספה של קובץ XML.

הפונקציה שמייצרת משפטים פועלת כך:

- מגרילים באופן אחיד אורך משפט על הקטע מ-1 עד לאורך המקסימלי.
- בונים משפט ריק ומוסיפים לו טוקן התחלה.
- מגרילים את המילה הראשונה בהתאם לסוג המודל, כך שאם מדובר ב-trigram, קודם כל נשתמש ב-bigram ל-2 המילים הראשונות.

עבור כל מודל מימשתי פונקציה שמגרילה את המילה הבאה בהינתן המילים הקודמות (לפי ה-n של המודל). בהינתן מילים קודמות של המשפט, אנו עוברים על כל המילים במילון שתחזקנו, ומחשבים את ההסתברות להופעת כל מילה כזאת בהינתן המילים הקודמות.

השתמשתי ב-random.choices שמאפשר לתת התפלגות ידועה מראש ומחזיר ערכים בהתאם.

עוצרים כאשר מגיעים לאורך המשפט שהוגרל או לטוקן סיום משפט.

<u>שאלות:</u>

1. לא קיבלתי משפטים הגיוניים לגמרי באף מודל, תמיד יש איזשהי טעות קטנה שפוסלת את המשפט. אבל מודל ה-trigrams מאוד קרוב למשפטים הגיוניים.

2. ניתן להסיק שככל שה-n של המודל גדול יותר, יכולות יצירת המשפטים של המודל משתפרות. זה מאוד הגיוני מכיוון שככל שערך ה-n גדול יותר, כך אנחנו יכולים להסתכל על מילה ועל המילים השכנות לה, ולהגיע למשפטים כך שיש קשר בין מילה ומילה.

3. ל-6grams יהיו ביצועים טובים יותר, בגלל מה שהסברנו בסעיף הקודם, ערך ה-n שלו יותר גדול מ-3, ויש לו אפשרות להסתכל על יותר מילים הקרובות למילה הנוכחית.

נשים לב שזה ישפיע על הרנדומליות של יצירת המילים מכיוון שברגע שמסתכלים על יותר מילים שכנות בכל פעם, יש לנו מעט אפשרויות לבחור את המילה הבאה כך שהיא תתאים לכל n המילים (כאשר n גדול).