השלמת משפטים אוטומטית

הערה

לכֿל אורך התרגיל אסור להשתמש בכלי Generative AI (לדוג' ChatGֿPT). מטרת התרגיל היא פיתוח חשיבה אלגוריתמית, ושימוש ב chat מוציא מהתרגיל את הערך שלו.

מבוא

מזל טוב! גויסתם לעבוד ב shmoogle, סטרטאפ חדש שמפתח מנוע חיפוש מתקדם. אחרי מספר בדיקות מול הלקוחות, איש ה product שם לב שהרבה משתמשים לא מוצאים תוצאות כי יש להם שגיאות כתיב. בנוסף, לא כל הלקוחות יודעים להקליד במהירות, מה שפוגע להם בחווית השימוש במוצר. לצורך הפיכת חווית המשתמשים במנוע החיפוש של החברה לטובה יותר, הוחלט להוסיף פיצ'ר חדש במנוע החיפוש הקיים - **השלמת משפטים.**

כדי לייצר את הצעות ההשלמה קיבלתם מאמרים, תיעוד וקבצי מידע נושאים טכנולוגים שונים, ועל ידי ניתוח שלהם, תצטרכו להציע למשתמש משפטים להשלמה, על פי קלט שהוא מכניס. הפיצ'ר עליו תעבדו יתמוך בשני מקרים שונים של השלמת משפטים, היכולים להתרחש באותו המשפט -

- טעויות הקלדה
- משפטים חלקיים

מבנה התוכנית

עליכם לספק תוכנית אשר רצה בשני שלבים:

- שלב ראשון (offline) - המערכת קוראת את קבצי הטקסט (שתקבלו מאיתנו ותשמרו במקום ידוע מראש) ומכינה אותם לשלב השירות - serving (נשמר כבר במודל הנתונים שבחרתם ומוכן להרצת החיפוש).

  דגשים מרכזיים -
  - תקבלו קבצי טקסט שכתובים בשפה האנגלית בלבד,

  (@!.,$ וכו...) המכילים אוסף משפטים (משפט מוגדר כשורה שלמה בתוך הקובץ.)
  - קבצי הטקסט מאוחסנים במבנה של עץ תיקיות, והם נמצאים בתיקיה Archive.zip
  - הטקסטים ממוקמים בגבהים שונים בעץ, כלומר, קובץ טקסט יכול להיות בתוך תיקיה, בתוך תיקיה בתוך תיקיה, וכדומה.

  פונקצית אתחול - מטרת הפונקציה היא לקבל רשימה של מקורות טקסט שעל בסיסם ירוץ מנוע החיפוש, כל מקור מכיל אוסף של משפטים.

- שלב שני (online) - המערכת מחכה לקלט (דרך ה CLI כמובן).ברגע שהמשתמש מקליד קלט ולוחץ על Enter, המערכת מציגה את חמשת ההשלמות הטובות ביותר (במקרה של שוויון על המערכת למיין את המחרוזות עם ציון זהה לפי אלפבית).
  - לאחר הצגת ההשלמות, המערכת מאפשרת למשתמש להמשיך להקליד מהמקום בו הוא עצר.
  - במידה והמשתמש מקליד "#" המשמעות היא שהמשתמש סיים הקלדה עבור משפט זה וצריך לחזור למצב ההתחלתי.

עליכם לכתוב תוכנית אשר תומכת בשתי פונקציות עיקריות:

פונקציית השלמה - על הפונקציה לקבל מחרוזת אשר מהווה את הטקסט שהמשתמש הקליד, על הפונקציה להחזיר את חמש ההשלמות הטובות ביותר (השלמה טובה תוגדר בהמשך).

חתימת פונקציית ההשלמה - לכל השלמה שנמצאה (סה"כ 5 השלמות):

- מה המשפט שהושלם
- לאיזה מקור המשפט המלא שייך
- באיזה מיקום הטקסט שהמשתמש הזין נמצא ביחס למשפט שהושלם (OFFSET)
- מה הSCORE - של ההשלמה

לנוחיותכם מצורפת החתימה של הפונקציה הנ"ל:

```
get_best_k_completions(prefix: str) -> List[AutoCompleteData]
```

כאשר AutoCompleteData - היא המחלקה הבאה:

```
@dataclass
class AutoCompleteData: completed_sentence: str source_text: str
        offset: int
score: int
# methods that you need to define by yourself
```

ממשק המשתמש

מטרתכם היא להראות את נכונות האלגוריתם שפתחתם, ולכן אין צורך ביצירת ממשק משתמש מורכב - ייצרו CLI (Command Line Interface) באמצעות argparse, שיאפשר להריץ את מנוע החיפוש, תוך טעינת קבצי הטקסט וניתוח שלהם, ולאחר מכן ימתין לקלט של משפט מהמשתמש. רק כאשר המשתמש לוחץ על Enter, לוגיקת ההשלמה תופעל ויוחזרו למשתמש ההצעות.

מספר דגשים -

- הדריכו את המשתמש שלא להקפיד על אותיות גדולות\קטנות וסימני פיסוק בהזנת הקלט, וטפלו בקלט, ובטקסט בקבצים המצורפים, באופן מתאים לפני החיפוש - במידה וישנם סימני פיסוק, התעלמו מהם זמן החיפוש.
- אין הגבלה על מספר הרווחים בין המילים. כלומר אם המשפט המקורי הוא "להיות או לא להיות, זאת היא השאלה" - אז גם אם המשתמש הקליד "להיות זאת," "להיות, זאת" או "להיות זאת" אזי על המערכת להתייחס לשלוש תתי מחרוזות כשקולות.
- הפלט של המערכת אמור להיות שורה מתוך קבצי המקור בצורתו המקורית (כלומר כולל פיסוקים), ואת הנתיב של הקובץ.

```
Loading the files and prepariong the system...
The system is ready. Entar your text:
this is
Here are 5 suggestions
1. Store a Python object in a C object pointer.  This is similar to (arg 332)
2. like objects.  **This is the recommended way to accept binary (arg 121)
3. "PyObject_NewVar()".  This is normally called from the "tp_dealloc" (allocation 49)
4. "tp_itemsize" field of *type*.  This is useful for implementing (allocation 40)
5. Information about writing a good bug report.  Some of this is (bugs 87)
this is
```

פעולת ההשלמה

מטרת פעולת ההשלמה היא למצוא את המשפט המתאים ביותר לקלט שהמשתמש שהזין. נגדיר התאמה בין פסוק (משפט) לבין טקסט שהמשתמש הקליד אם אחד משני התנאים מתקיים:

- הטקסט הוא תת מחרוזת של הפסוק (תחילת, אמצע או סוף הפסוק.)
- טקסט בו נדרש תיקון יחיד כדי להפכו לתת מחרוזת של הפסוק.

תיקון מוגדר כאחד משלושת הפעולות הבאות -

- החלפת תו.
  - **דוגמה**: המחרוזת **"להיות או לו"** נחשבת כתת מחרוזת של הטקסט **"להיות או לא להיות** זאת היא השאלה," עם החלפת התוו "ו" במילה "לו" ל- "א".
- מחיקת תו.
  - דוגמה: המחרוזת **"להות או לא"** נחשבת כתת מחרוזת של הטקסט "**להיות או לא להיות** זאת היא השאלה," עם מחיקת התוו "י" במילה "להיות."
- הוספת תו.
  - דוגמה: המחרוזת **"להיות אוו לא"** נחשבת כתת מחרוזת של הטקסט **"להיות או לא להיות** זאת היא השאלה," עם הוספת התוו "ו" במילה "או."

ציון התאמה

במקרה של מספר תיקונים אפשריים לטקסט שהוקלד, נגדיר ציון (score) לכל התאמה כדי למצוא את ההתאמה המוצלחת ביותר:

- הניקוד הבסיסי הוא כפול ממספר התווים (לא כולל פיסוקים (@!.,$ וכו...) שהוקלדו ועבורם נמצאה התאמה.
- החלפת תו מורידה נקודות לפי מיקומו במחרוזת:
  - טעות בתו הראשון - 5 נק'
  - טעות בתו שני - 4 נק'
  - טעות בתו השלישי - 3 נק'
  - טעות בתו הרביעי - 2 נק'
  - טעות בתו בהמשך המחרוזת (חמישי והלאה) - נק' אחת
- מחיקה או הוספת תו מורידה נקודות לפי מיקום השינוי במחרוזת -
  - תו ראשון - 10 נק'
  - תו שני - 8 נק'
  - תו שלישי - 6 נק'
  - תו רביעי - 4 נק'
  - תו בהמשך המחרוזת (חמישי והלאה) - 2 נק'

מצורפות מספר דוגמאות לציון להשלמת משפטים, אשר מניחות שהמשפט "להיות או לא להיות, זאת היא השאלה" נמצא במידע שנטען למנוע החיפוש

- הטקסט "להיות או לא" מקבל 22 נק' - מתאים למחרוזת של הטקסט, ומכיל 11 תווים.
- הטקסט "להיות או לו" מקבל 19 נק' - 10 תווים נכונים בהתאמה למחרוזת של הטקסט ותו אחד שהוחלף, במיקום חמישי והלאה.
- הטקסט "להיות או לא" מקבל 18 נק' - 10 תווים נכונים בהתאמה למחרוזת של הטקסט ותו אחד שהוחלף במקום הרביעי.
- הטקסט "להייות או לא" מקבל 18 נק' - 11 תווים נכונים בהתאמה למחרוזת של הטקסט ותו אחד שנוסף במקום הרביעי.
- הטקסט "להות או לא" מקבל 14 נק' - 10 תווים נכונים בהתאמה למחרוזת של הטקסט ותו אחד שחסר במקום השלישי.

מדדים להצלחה

- נכונות הפתרון - נכונות ההשלמות שהמשתמש קיבל.
- חווית המשתמש (יעילות) - כמה זמן לקח לאלגוריתם לספק למשתמש את ההשלמות? חשבו על הזמן שלוקח לגוגל להשלים משפטים בחיפוש כדוגמה לחווית משתמש טובה.
- שימוש בזיכרון - נסו להשתמש בכמה שפחות זכרון, מבלי לפגוע יתר על מידה בחווית המשתמש.
- רמת גימור של הפרויקט - עבודה נכונה עם גיט, שימוש בבראנצ'ים עם שמות אינדיקטיבים, main עובד תמיד, PRים ו CRים הדדיים, איכות קוד.
- טסטים ולוגים - כתבו טסטים שבודקים רמות שונות של הקוד שלכם (system tests, integration tests, unit tests) - בדיקות של פונקציות משמעותיות,בדיקות של אינטגרציה בין פונקציות שונות ובדיקה אחת לפחות של כלל המערכת. כמו כן, כתבו לוגים כחלק מהקוד שלכם, הקפידו על חלוקה לרמות לוגים שונות.
    - קראו על מבנה תיקיות נכון של פרויקט פייתון עם test ים
    - אין צורך לייצר טסטים עבור ממשק המשתמש
- Readme אינדיקטיבי - מצורף template של קובץ readme, בו תסבירו במספר משפטים על הקוד ועל האלגוריתם שלכם. שימו לב לתחזק אותו לאורך העבודה על הפרויקט.

הלן המלצה למתודולוגיית עבודה בפרויקט -

1. חשבו על הבעיה והבינו אותה היטב מבלי לכֵתוב קוד, נסו להבין אילו אובייקֵטים\מבני נתונים דרושים וכיצד הרכיבים השונים מתממשקים, ייצרו אפיון של הפרויקט - **כל הקבוצה במשותף.** שימו לב להגביל חלק זה בזמן, ונהלו את הדיון באופן יעיל.
    a. זכרו לעדכן את ה readme
2. השתמשו בכלי ניהול משימות כדי לחלק את העבודה ביניכם.
3. דאגו לכך שחלק מהצוות מממש תוכנית מינימלית ושלמה -
    a. לא לממש את כל הפונקציונליות - כלומר לממש השלמה פשוטה בלבד.
    b. לממש את הקוד כך שיהיה אפשר להרחיבו בעתיד.
4. שלבו את מבנה הנתונים שבחרתם בתוכנית
5. שפרו את ביצועי התוכנית, הן מבחינת זכרון והן מבחינת יעילות, ושפרו את מבנה הנתונים שלכם בהתאם.
    a. שימו לב לשמור על קוד עובד בכל איטרציה ייעול.

בונוס

כתיבת את פונקציית ההשלמה בשפת C++ על מנת לקבל ביצועים טובים יותר.

ממשק בשפת :C++

```
vector<AutoCompleteData> GetBestKCompletions(const string& prefix);
```

כאשר AutoCompleteData - היא המחלקה הבאה:

```
class AutoCompleteData { private:
string completed_sentence; string source_text
int offset; int score;
// methods that you need to define by yourself
};
```