דו"ח תרגיל בית 1

קורפוסים

**שלב 1**

**שלב 1.1:**

**שליפת הנתונים של מספר הכנסת וסוג הפרוטוקול** ממומשת ע"י הפונקציה 'extract\_metada\_from\_name'. אופן העבודה שלה מתבסס על העובדה שלשמות קבצי הפרוטוקולים יש מבנה מסוים. הפונקציה מקבלת כקלט את שם הקובץ ומחלצת ממנו את הנתונים הדרושים ע"י לחלק אותו ל- 3 חלקים לפי הסימן '\_', ומחזירה את החלק הראשון כמספר הכנסת והאות האחרונה בחלק השני כסוג הפרוטוקול.

**שלב 1.2**:

**שליפת מספר הפרוטוקול** מתממש ע"י הפונקציה 'extract\_metada\_from\_content'. המימוש שלה מתבסס על העובדה שהמספר של הפרוטוקול בא אחרי אחד משני צירופי המלים "הישיבה" או "פרוטוקול מס'". היא מקבלת כקלט את התוכן של הקובץ, ומתחילה בלעבור על כל המופעים של המילה "הישיבה" ובודקת אם המילה שמופיעה אחריה היא מספר, אם לא מצליחה למצוא כזה מופע, היא עושה אותו דבר עבור המופעים של הצירוף "פרוטוקול מס'". הפונקציה עוצרת בעת שהיא מוצאת מספר כמתואר למעלה ומחזירה אותו, או כאשר עוברת על כל המופעים של שני הצירופים ולא מוצאת, ואז מחזירה -1.

**בדיקת האם המילה היא מספר** ואת ערכה במידה שהיא מספר מתבצעת ע"י הקריאה לפונקציה 'convertToInt'. אשר מקבלת כקלט משתנה string, שיכול להיות מורכב מספרות (כמו 73) או מאותיות (כמו: שבעים ושלוש). היא ממירה אותו למקביל שלו ב- int, ואם לא מצליחה היא מניחה שזה לא מספר ומחזירה -1.

שלב 1.3:

הדרוש בסעיף ממומש ע"י הפונקציה 'extract\_relevant\_text', אשר משתמשת גם בפונקציות עזר שונות. היא מקבלת כקלט קובץ ופרוטוקול (מופע של המחלקה Protocol), ומחלצת מהקובץ את המשפטים הרלוונטיים ושומרת אותם לתוך הפרוטוקול. היא מבצעת את העבודה באמצעות השלבים הבאים:

1. למצוא את הטקסט הרלוונטי, אשר הוגדר בנתוני השאלה כ- "הטקסט שנאמר על ידי דוברים בוועדה\מליאה". מהתסתכלות בקבצי הפרוטוקולים השונים ניתן לראות שהקט הרלוונטי נמצא ברצף, לכן נרצה למצוא את תחילת הקטע וסופו.
   1. **מציאת תחילת הקטע הרלוונטי** מתבצעת ע"י הקריאה לפונקציית העזר ' find\_starting\_relevant'. מהידע שלנו על אופן התנהלות הדיונים בכנסת הוועדה\המליאה מתחילה בדבריו של היו"ר של הכנסת או המ"מ שלו. דבר זה משתקף בקבצי הפרוטוקולים. כמו כן בעיון בקבצים השונים נוכל לשים לב שהתואר שלו תמיד נמצא לפני שמו, ו- ":" תמיד מופיעות אחרי שם כל דובר.. לכן המטלה הופכת להיות ללמצוא את המשפט הראשון שמתחיל באחד מהצירופים יו"ר\ היו"ר\ יו"ר הכנסת\ מ"מ היו"ר ונגמר ב- ":", זהו תחילת הקטע הרלוונטי.
   2. **מציאת סוף הקטע הרלוונטי** מתבצעת ע"י הקריאה לפונקציית העזר ' find\_last\_relevant'. ניתן לשים לב שכמעט כל דיון נגמר במשפט "הישיבה ננעלה בשעה .." או לחלופין "הטקס ננעל בשעה ..". משפט זה בא אחרי המשפטים שנאמרו ע"י הדוברים, וגם לא נאמר ע"י מישהו, לכן ניתן להניח שהוא המשפט הראשון הלא רלוונטי. כמו כן, ניתן לשים לב שבמקרים שהקובץ לא נגמר במשפט מהסוג הזה הדיון נמשך עד הסוף. על סמך הזה הפונקציה מאתר את סוף הקטע הרלוונטי ע"י להתחיל לחפש מהסוף את אחד מבין צירופי המלים "הישיבה ננעלה" או "הטקס ננעל", ואם לא מוצאת אותן אז היא מחזירה שהסוף של הקטע הוא בסוף הקובץ.
2. מיון המשפטים בקטע הרלוונטי לאחד מבין שני הסוגים: שם הדובר, משפט שהוא חלק מהנאום. זה מתבצע ע"י הפונקציה הראשית ('extract\_relevant\_text') אשר עוברת על הקטע שסומן כרלוונטי, היא מאתרת את המשפטים מסוג "שם הדובר" ע"י כך שהיא בודקת אם הוא עומד בשלושה תנאים: יש קו מתחתיו, מסתיים ב- ":", ומכיל לכל היותר 5 מילים אחרי הניקיון (על ניקיון השם מפורט אח"כ). אם שלושת התנאים מתקיימים, המשפט נחשב לכזה המכיל שם דובר, וכל המשפטים שמפרים אחד מהתנאים הללו ובאים אחריו משויכים אליו, עד למציאת שם דובר חדש. ברגע שמשפט של נאום נמצא הוא מטופל ע"י פונקציית העזר sentence\_handle, אליה מועבר המשפט, שם הדובר, והמופע של הפרוטוקול. על אופן עבודת הפונקציה מפורט יותר בלשבים 1.4-1.7.

* ניקוי השם: מתבצע ע"י הקריאה לפונקציה 'speakerClean'. מטרת הפונקציה היא "לנקות" את השם, כלומר להסיר כל דבר חוץ שמכיל השם מלבד השם הפרטי ושם המשפחה, ואולי גם שם האב. זה כולל תארים (כמו השר, היו"ר, ד"ר וכדו') או שם המפלגה או סימונים של <> שבהרבה פורמטים מופיעים לפני ואחרי השם. מימוש הפונקציה מתבסס על סקירת האופנים השונים שבהם יכול השם להופיע. למשל שם המפלגה תמיד בא בין סוגריים, לכן הפונקציה מאתרת סוגריים כאלה ואם מוצאת אותם מוחקת אותם מהשם ביחד עם מה שיש בתוכם. הפונקציה מנסה לאתר תארים שונים נפוצים ולמחוק אותם. בתארים של שר ושרה היא מניחה שיבוא אחריהם את שם המשרד של השר, לכן היא בודקת אם אחד משמות המשרדים מתוך רשימה מופיעים אם כן גם אותם מסירה.

**שלב 1.4**:

**המשפטים מחולקים ע"י [. ! ?]**. הדבר מתבצע ע"י הפונקציה 'sentence\_handle', היא עוברת על כל אות במשפט שאותר כמשפט נאום, ובודקת עבור כל אות אם הוא אחד משלושת הסימנים, אם כן הוא חותך את מה שלפניו מזה שאחריו. לסימן "." יש הסתייגויות שבהם הסימן לא נחשב למסיים משפט, והם:

1. כאשר הנקודה היא עשרונית, או כחלק מביטוי של תאריך: למשל,

"העמיתים יקבלו פיצוי בסך של **23.8** מיליוני שקלים"

" אי אפשר להחליף טיוטות ושבחוזה יהיה כתוב שהכל מתחיל מה-**1.1.2014**"

מקרה זה מזוהה כאשר הנקודה באה בין שתי ספרות.

1. כאשר הנקודה היא מסדרת: למשל,

"השינויים העיקריים הם : **1.** הגבלה בחוק של תקופת הזמן לשהייה ..."

" אז **א.** זה לא קורה ; **ב.** בית-המשפט מתנגד לחוקים הדרקוניים ..."

הנקודה מסוג זה מופיעה בד"כ ברשימה ממוספרת, בעלת משמעות דקדוקית, ולא משמשת לסיום משפט.

מקרה זה מזוהה כאשר הנקודה מופיעה אחרי ספרה או אות בודדת (כאשר הספרה\האות באה בראש המשפט או כאשר לפניו רווח).

הייתה התלבטות לגבי אם לכלול את הסימן ":" כמפריד בין משפטים, והוחלט שלא, מכיוון שברוב הפעמים הוא לא מסיים משפט. הדוגמאות הבאות הנלקחות מפרוטוקולים ממחישות את המסקנה:   
"ועדת הכספים תדון בהצעה לסדר-היום בנושא : מאהל המחאה של נפגעי הבנקים והבורסה"

"אני רוצה לומר לך : יישר כוח"

**שלב 1.5**:

המשפטים מנוקים אחרי ביצוע השלב שתואר בסעיף הקודם (1.4), הם נשלחים לפונקציה 'sentence\_validity' שאחראית לכך. הפונקציה פוסלת את המשפט אם הוא עונה על אחד משלושת התנאים הבאים:

1. משפט שלא מכיל אף אות מבין אותיות האלפבית העברי.
2. משפט שמכיל אותיות באנגלית.
3. משפט חתוך.

למשל, "אז באופן פרופורציונאלי פגעתם גם במשטרה, גם ב**---**"

אחרת, הוא המשפט מצליח ב- "מבחן החוקיות" והפונקציה מחזירה True.

**שלב 1.6**:

השלב מתבצע ע"י הפונקציה 'sentence\_tokenize'. הטוקנים מופרדים ע"י כל אחד מבין הרווח והסימנים והבאים [: ; ( ) – " ,], תוך התחשבות במקרים מסוימים בהם חלק מהסימנים נחשבים כחלק מטוקן ולא כטוקן מופרד, והם:

1. כאשר ["] משמשות לציון ראשי תיבות. למשל: ת"א, חב"ד.

מקרה זה מזוהה ע"י כך שהסימן בא בין שתי אותיות.

1. כאשר [,] הוא חלק ממספר, בעיקר כמפריד אלפים. למשל:

"מיליארד , לא מיליון ; כל מיליארד זה **1,000** מיליון"

מקרה זה מזוהה ע"י כך שהסימן בא בין שתי ספרות.

1. כאשר [:] חלק מביטוי זמן, בעיקר מפריד בין שעות לדקות. למשל:

"האופוזיציה היו כאן ב - **08:30** אפילו שהיא לא צריכה להתפלל – אני מתפלל"

""אדוני היושב - ראש , השעה **24:00**"

מקרה זה מזוהה ע"י כך שלפני הסימן ספרה אחת לפחות ואחריו שתי ספרות.

**שלב 1.7**:

השלב מטופל ע"י הפונקציה 'sentence\_tokenize'. אחרי שהפונקציה מבצעת טוקניזציה וכאשר הטוקנים נמצאים בתוך רשימה (לפני חיבורם למשפט שמופרד ברווחים) היא בודקת את אורכה ואם האורך לפחות 4 היא מאחדת את הטוקנים למשפט, אחרת פוסלת את המשפט.

**שלב 1.8**:

הצעד מטופל ע"י הפונקציה ' jsonl\_make' שממירה את הרשימה protocols לפורמט של jsonl. היא פותחת את קובץ הפלט במצב כתיבה עם קידוד UTF-8. ועוברת על כל המופעים של protocol, כאשר בכל אחד מהם היא עוברת על המשפטים (ששמורים בו כ- attribute מסוג מילון כאשר המפתח הוא שם הדובר והערך הוא רשימת המשפטים שהוא אמר). היא יוצרת עבור כל משפט שורה בקובץ ה- jsonl עם התכונות שנדרשו בהוראות.

**שלב 2**

**שאלה 2.1**

מצד אחד, פיצול מוספיות חשוב בתהליך של עיבוד שפות על מנת להיות אפשרי לזהות ששתי מלים הן זהות גם אם הן עם מוספיות שונות. כמו כן, שמירת המוספיות כטוקנים נפרדים מאפשר לזהות את ההקשרים התחביריים בצורה יותר קלה, למשל, במקרה המוספית "כש-" המביעה קשר של זמן, ושקולה למילה "כאשר". מצד שני, בעברית קיימים מקרים שבהם קשה לפעמים לזהות אם אות היא מוספית או שהיא חלק מהמילה, וביצוע פיצול כזה יכול להיות כרוך בטעויות. דוגמה לכך היא האות "ש" במילה רבת המשמעויות "שבתה", בה ה- "ש" יכולה להיות חלק מהמילה כאשר היא באה כנרדפת ל- to strike, ויכולה להיות מוספית כאשר היא בה כנרדפת ל- "that her daughter".

**שאלה 2.2**

ניתן לפצל את המילה ל- 4 טוקנים: **"ו", "כש", "יבוא", "ו"**. כאשר "ו" הראשון מציין חיבור, "כש" מציין זמן או תנאי, "יבוא" היא המילה המרכזית בזמן עתיד, ו- "ו" האחרונה מציינת ריבוי. הפיצול הזה מאפשר לזהות את החלק של המילה, כמו כן מאפשר להבין את ההקשר, כאשר "ו" מחברת את המילה למה שהופיע קודם לה, ו- "כש-" עוזר בלהבין שמה שאחריה (כלומר, המילה "יבואו") הוא תנאי להתקיימות של משהו שכנראה הוזכר לפני או יוזכר אחרי. כמו כן, ה- "ו" בסוף עוזרת בלהבין שהנושא הוא ברבים, ובמקרה שהוא לא מצוין במפורש אחרי המילה אז ניתן לשייך את הפועל לנושא שבא ברבים בחלק שלפני במשפט.

**שאלה 2.3**

היתרונות:

* מאפשר **ניתוח מאפיינים סטטיסטיים של משפטים** באופן יותר פשוט.

למשל, ניתן לסנן את המשפטים לפי הקריטריון "נאמר ע"י ח"כ ערבי", ובכך גם לחשב דברים כמו את ההסתברות של ח"כ ערבי להזכיר את המילה "כיבוש".

החסרונות:

* **סיבוכיות מקום גבוהה**: ה- Metadata כמו מס' הפרוטוקול ומס' הכנסת חוזר על עצמו בכל משפט, דבר המוביל להגדלת נפח קובץ ה- JSON.
* **אובדן ההקשר**: קשה להבין הרבה משפטים בשל היותם מסודרים לפי שם הדובר, ובכך מנותקים מהסדר ומההקשר בהם הם נאמרו. הרבה משפטים, במיוחד הקצרים, למרות היותם תקינים מבחינת הדרישות של המטלה, הם נראים כמו חסרי כל ערך אם לא קוראים אותם בהקשר שלהם. פתרון יותר טוב במקרה הזה הוא רשומה עבור כל פרוטוקול.

**שאלה 2.4**

כ