דו"ח תרגיל בית 1

קורפוסים

עובידה חטיב , ת.ז: 201278066

מאיה עטואן, ת.ז:314813494.

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.

דוגמה : שם הקובץ : 25_ptv_3841247.docx אז הפונקציה עושה split('_') לשם הקובץ ושומרת את הערכים במערך, הערך בתא 0 זה מספר הכנסת 25 :"knesset_number"

האות האחרונה בתא 1 בודקת את סוג הפרוטוקול בין אם m או v, במקרה שלנו זה v, כלומר שומרת "protocol_type": "committee" .

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.

הפונקציה מחזירה את המופע הראשון של מספר פרוטוקול\ישיבה. במקרים שבהם יש יותר ממספר פרוטוקול אחד (דוגמאות למטה), הפונקציה מחזירה את המספר הראשון המופיע.

דוגמה 1:

פרוטוקול מס' 110

מישיבת ועדת החינוך, התרבות והספורט

ופרוטוקול מס' 33

מישיבת הוועדה המיוחדת לזכויות הילד

הפונקציה מחזירה את המספר 110.

דוגמה 2:

פרוטוקול מס' 141, 144, 145

הפונקציה מחזירה 141 .

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.

בטיפול במספרים המורכבים מאותיות הפונקציה לוקחת בחשבון את האות "ו" המסמלת חיבור, כמו גם את זה שלפעמים המלים שמרכיבות את שם המספר מופרדות ב- "-". הפונקציה יכולה להתמודד עם מספרים עד 1999, הדבר נקבע בהתבסס על העובדה שהמספרים של הפרוטוקולים המצורפים לא עלה על 500.

דוגמה:

הישיבה המאתיים-ושישים-ותשע של הכנסת השש-עשרה

הפונקציה 'extract_metada_from_content', מוצאת את המילה "הישיבה", שולחת את המילה שאחריה לפונקציה 'convertToInt', שבתורה בודקת אם יש אחריה מספר במלים. אם אכן יש, אז מתחילה לחשב אותו בהתחלה split('-') אחר כך מחיקת "ה" ו "ו" . מחשבת את המספר ע"י המרה בעזרת המילון שהגדרנו ומחזירה 269 .

דוגמה 2:

פרוטוקול מס'

הפונקציה מחזירה -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". מההסתכלות בקבצי הפרוטוקולים השונים ניתן לראות שהקטע הרלוונטי נמצא ברצף, לכן נרצה למצוא את תחילת הקטע וסופו.

a. **מציאת תחילת הקטע הרלוונטי** מתבצעת ע"י הקריאה לפונקציית העזר ' find_starting_relevant'. לאחר בדיקה ידנית של הקבצים שיש לנו , זיהינו שהדובר הראשון הרלוונטי מסומן בעיצוב טקסט ייחודי, בדרך כלל שורה מסומנת בקו תחתי underline. ולאחר שם הדובר מופיע הסימן ":" . דוגמה:

<u>היו"ר ע' מאור:</u>

אופן פעולת הפונקציה: שלב ראשון, כדי להימנע מזיהוי טקסטים לא רלוונטיים שמופיעים בקבצים כמו כותרות ומבואות , הפונקציה משתמשת ברשימה ייעודית skip_keywords שכוללת מילים וביטויים נפוצים שאינם מייצגים נמצאים בקטע הרלוונטי שאנחנו מחפשים למשל "סדר היום" , "נוכחים" ועוד. בניית הרשימה הייתה ע"י סקירת הקבצים ולזהות אילו מילים צריך לדלג עליהם. אם הפסקה מתחילה או נמצא בה אחת המילים שנמצאות ברשימה אז עושים דילוג.

שלב שני, אנחנו מנקים את הטקסט מרווחים ומתווים כמו <<...>> ן <..> שעלולים להפריע בזיהוי. אחרי כך אנחנו בודקים עיצוב הטקסט וסימון בסיום הפסקה 1- מעוצבת ב-underline, 2- הפסקה מסתיימת בסימן ":". אם שני התנאים מתקיימים, הפונקציה מזהה את הפסקה כתחילת הטקסט הרלוונטי ומחזירה את המיקום שלה במסמך.

דוגמאות:

1- <u><<יור>> היו"ר יואב קיש: <<יור>></u> (עם הפורמט הזה) אחרי הניקיון יהפוך ל "<u>היו"ר יואב קיש:</u>" שזה אכן הטקסט הרלוונטי הנדרש, שעונה לדרישות

2- <u>בקשת סיעת "מחר" לשינוי שם הסיעה</u> (עם הפורמט הזה) למרות שהיא מעוצבת בעיצוב הנדרש ולא כוללת מילה ממילים שב skip_keyword אך לא מסתיימת ב ":" לכן הפונקציה ממשיכה לפסקה הבאה במצב כזה.

3- מר שלמי גולדברג: (עם הפורמט הזה) הפונקציה מתחילה לעבור על הפסקה הזאת לא נמצא בה אחת המילים מ skip_keyword ומזהה שהוא אכן בעיצוב שאנחנו מחפשים underline ומסתיים ב ":" אז הפונקציה מחזרה את האינדקס של הפסקה הזאת כאינדקס התחלתי.

b. **מציאת סוף הקטע הרלוונטי** מתבצעת ע"י הקריאה לפונקציית העזר 'find_last_relevant'. אופן פעולת הפונקציה נסמכת על כך שכמעט כל פרטוקול נגמר במשפט "הישיבה ננעלה בשעה ..". משפט זה נמצא ישר אחרי המשפטים שנאמרו ע"י הדוברים. כמו כן, הוא לא נאמר ע"י מישהו, לכן ניתן להניח שהוא המשפט הראשון הלא רלוונטי. בנוסף לכך, במקרים שבהם שהקובץ לא נגמר במשפט הזה הדיון נמשך עד הסוף. על סמך המידע הזה הפונקציה מאתרת 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 כסוף הקובץ.

2. **מיון המשפטים בקטע הרלוונטי** לאחד מבין שני הסוגים: משפט שמכיל את שם הדובר, משפט שנאמר ע"י דובר (משפט נאום). הדבר 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 (underlined), 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.

- יוצא מהכלל מבין המשפטים ב- "קטע הרלוונטי" הם המשפטים המתייחסות להצבעות או המשפטים המודיעים על הפסקה באמצע הישיבה. שני הסוגים של המשפטים נמצאים בקטע הרלוונטי, אולם לא נחשבים לשמות דוברים או למשפטים המכילים דיבור. במקרה של משפטים כאלה הם לא נלקחים בחשבון. משפט המתייחס להצבעה הוא בד"כ אחד מבין קבוצה של כ- 5 משפטים כאשר הראשון בהם הוא המספר של ההצבעה או הנושא שלה. המשפטים הבאים אחריו הם לגבי מספר המצביעים בעד או נגד החוק ואלה שנמנעו. המשפט האחרון מתייחס להחלטה שהתקבלה בעקבות ההצלחה. משפטים אלה מזוהים ע"י כך שכאשר המשפט הראשון מכיל את המילה "הצבעה", והשני והשלישי

מכילים את המלים "בעד" ו- "נגד" בהתאמה. המשפטים המודיעים על הפסקה הם משפטים בודדים שמתחילים ב- "הישיבה נפסקה .." ולפי זה הם מזוהים. דוגמה לקבוצת משפטים המתייחסות להצבעה:

הצבעה מס' 2

בעד ההצעה לכלול את הנושא בסדר-היום – 36

נגד – אין

נמנעים – אין

ההצעה לכלול את הנושא שהעלו חברי הכנסת א' דיין, ז' המר,

ש' בניזרי ומ' קצב בסדר-היום נתקבלה.

- יוצא מהכלל של המשפטים העונים על שלושת הדרישות של שמות הדוברים הם המשפטים "קריאה" ו- "קריאות" אשר מקיימים את שלושת התנאים הנ"ל אולם לא נחשבים לשם דובר תקני ועל כך את מה שנקרא אחריהם משויך לשם הדובר האחרון.
- **ניקוי השם** מתבצע ע"י הקריאה לפונקציה 'speakerClean'. מטרת הפונקציה היא "לנקות" את השם, כלומר להסיר כל דבר מלבד השם הפרטי ושם המשפחה, ובמקרים מסוימים גם את שם האב או שם המשפחה של בן הזוג. דברים שמנוקים כוללים תארים (כמו השר, היו"ר, ד"ר וכדו'), שם המפלגה או סימוני ניווט (כמו <, >) שבהרבה פורמטים מופיעים לפני ואחרי השם. מימוש הפונקציה מתבסס על סקירת האופנים השונים שבהם יכול השם להופיע. למשל שם המפלגה תמיד מופיע בין סוגריים, לכן הפונקציה מאתרת סוגריים ואם מוצאת אותם מוחקת אותם מהשם ביחד עם מה שיש בתוכם. הפונקציה מנסה לאתר תארים שונים נפוצים ולמחוק אותם. בתארים של "שר" ו- "שרה" היא בודקת אם אחריהם יש את שם המשרד של השר\ה, לכן היא בודקת אם אחד משמות המשרדים מתוך רשימה מופיעים, אם כן מסירה אותו. דבר נוסף שהפונקציה מטפלת בו הוא ריבוי רווחים, כאשר היא מחליפה אותם ברווח אחד.

3b:

- מבין האתגרים שהתמודדנו איתם במהלך מימוש הפונקציה הוא בלקבוע מתי צירוף האותיות "שר" או "שרה" הם חלק מהשם (כמו במקרה של אושר ואושרת) ומתי הם שם תואר. לאחר ניסוי וטעיה הוסק שכאשר המילה מופיעה בראש המשפט אחרי ניקוי הניווט, ואחריו יש רווח אז הוא מצביע על שם תואר, ורק אז יש להסיר אותו. אתגר נוסף היה בלהחליט על המקרים בהם יש למחוק את

הסימן "-" מהשם, ואחרי סקירה של המקרים השונים בהם השמות הכילו את הסימן הגענו למסקנה שהוא רלוונטי רק כאשר הוא בתוך השם, כלומר כאשר לא נמצא באחד הקצוות.

- למרות הניקוי שביצענו לשמות הדוברים, אחת הבעיות שלא טופלה היא המקרים בהם שמות דוברים מופיעים ביותר מוריאציה אחת, בעיקר בשל כך שבהרבה פרוטוקולים השתמשו בשם המלא של הדובר לפעמים ובקיצור של השם האישי שלו בפעמים אחרות. דוגמאות לכך הם יוסף עזרן, המופיע בשם זה לצד השם י' עזרן, ואסתר סלמוביץ המופיעה גם בשם א' בלמוביץ, שניהם בפרוטוקול ptm_532240.docx_13. במקרים אחרים, השתמשו בכינוי שידוע בו האדם במקום את שמו האמיתי, דוגמת ראובן ריבלין המופיע בהרבה פרוטוקולים כ-"רובי ריבלין". כתוצאה מכך המשפטים שנאמרו ע"י אותו דובר לא מופיעות ברצף תחת אותו שם אלא מחולקים לשני חלקים.

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.
2. 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 (אשר נגמר ב- ... או - -).
   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 הבאים [, " – ( ) ; :], 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.
2. 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.
3. 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.
4. **כאשר [.] היא נקודה עשרונית או נקודה מסדרת**. דוגמאות לכך ודרך הזיהוי זהה למה שהוזכר בשלב 1.4.

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". לכן לדעתנו פיצול מוספיות חשוב מאוד למודל שפה. אולם, מצד שני, יש לקחת בחשבון את העבודה שבעברית 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". בנוסף לכך פיצול מוספיות דורש שימוש בכלים מורפולוגים כמו מנועי ניתוח תחבירי נכון שכלים אלה משפרים את הדיוק אך הם עלולים להאט זמן הריצה יותר.

שאלה 2.2

מורפמה היא היחידה הקטנה ביותר שיש לה משמעות או תפקיד דקדוקי לכן היינו מחלקים אותה ל 5 טוקנים: **ו, כש, י, בוא, ו**.

ו (בהתחלה) – כמורפמה חבורה מציינת יחס חיבור בין המשפט הקודם.

כש – כמורפמה חבורה מציינת זמן שיתרחש בעתיד או תנאי. עוזר בלהבין שמה שאחריה (כלומר, המילה "יבואו") הוא תנאי להתקיימות של משהו שכנראה הוזכר לפני או יוזכר אחרי.

י – כמורפמה חבורה מציינת זמן וכאן בעתיד , משפיעה על הפועל שאחריה.

בוא – כמורפמה פרודה מציינת פועל המרכזי במשפט.

ו (בסוף) – כמורפמה חבורה מציינת גוף רבים , ובמקרה שהוא לא מצוין במפורש אחרי המילה אז ניתן לשייך את הפועל לנושא שבא ברבים בחלק שלפני במשפט.

הסיבה לבחירה הזאת היא שכל מורפמה חבורה היא משנה את המשמעות ואפשר להשתמש במורפמות החוזרות כמו "ו" "כש" עם פעלים שונים מבלי ללמוד כל מילה מורכבת מחדש. זה מפשט את תהליך עיבוד השפה ומייעל את למידת השפה (בשפה העברית היא עשירה במורפולוגיות).

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 וקושי בשחזור רצף הדיון:** 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

היינו בוחרים לשמור את המידע רשומה עבור כל פרוטוקול, מכיוון דרך זו שומרת על הקשר המלא במשפטים ומאפשרת ניתוח מעמיק על רמת פרוטוקול, משפטים וגם כן התנהגות הדוברים.

בצורה הזאת אנחנו מאפשרים למודל להבין את הזרימה הכוללת של השיחה, הקשרים במשפטים כלומר המשפטים לא מאבדים את משמעותם. בנוסף אפשר לנתח וללמוד מבנה הדיונים כלומר כמה זמן דובר מסוים מדבר, מי התפקידים הראשיים בישיבות , מתי נפסקת ישיבה\ וועדה , תגובת דובר מסוים בנושא מסוים ועוד מלא דברים. מעבר לכל זה בשיטה הזאת אנחנו מאבדים פחות מידע מכיוון שכל משפט נשאר בהקשר המקורי שלו וזה מונע בעיות הבנה ממשפטים שמנותקים משיחה .

בהשוואה לאופציות אחרות: אם היינו שומרים כרשומה לכל משפט אז המודל מאבד את המשמעות של המשפטים ולא מצליח להבין את השיחה ומבנה הוועדה\ישיבה.

אם היינו שומרים כל המשפטים שנאמרו על ידי אותו דובר כרשומה אכן היינו יכולים ללמוד את אופן התנהגות הדובר אך רוב המשפטים יאבדו את הקשרם ומתי נאמרו ואז בחינת התנהגותם לא מדויקת.