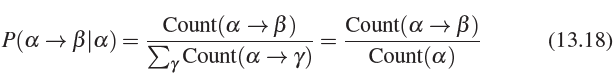
שאלה 1**:** דיוק ה-parsing ה-dummy parser עבור משפטים באורך לכל היותר-40 מילים:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagging accuracy** | **Bracketing FMeasure** | **Bracketing Precision** | **Bracketing Recall** |
| 18.78 | -1.#J | 0.00 | 0.00 |

שאלה 2

שאלה 2.1: הנוסחה לחישוב ההסתברויות לחוק (פרק 13 עמ' 8 בספר הלימוד):



שאלה 2.2: פסאודו קוד לבינרזיציה של עצים: (קוד רקורסיבי, הקריאה הראשונה מעבירה כפרמטר את שורש העץ)

**CFG\_TO\_CNF**(Input: ***node***)

1. If ***node*** has daughters:
   1. Invoke CFG\_TO\_CNF recursively on first (leftmost) daughter
   2. If there are more than two daughters:
      1. Create a new artificial node ***artificialNode*** (for node identifier scheme, see bellow)
      2. Remove redundant daughters from ***node*** and assign all “redundant” daughters to ***artificialNode***
   3. If (at this point) ***node*** has two daughters: (see explanation bellow)
      1. Invoke CFG\_TO\_CNF on second (rightmost) daughter

הסבר לשלב 1.3: בשלב זה באלגוריתם, אחת משתי האפשרויות מתקיימת:

* ל-***node*** היו לכתחילה בת אחת או שתי בנות: במצב הזה, לא רץ שלב 1.2, ובשלב 1.3 המצב נותר כשהיה, ופשוט עובר על הבנים (במידה וקיימים) של הnode ושולח אותם כדי לבצע עליהם המרה ל-CNF.
* במידה ול-***node*** היו לכתחילה יותר משתי בנות: במצב הזה, שלב 1.2 רץ. בשלב 1.2 יוצרים בת "חדשה מלאכותית", וכל שאר הבנות (החל מהאינדקס השני, ועד לסוף רשימת הבנות), הופכים להיות בנות של הצומת המלאכותי. כלומר, שוב ל-***node*** יש שתי בנות - הבת השמאלית המקורית והבת הימנית המלאכותית החדשה.
  + במקרה הזה, נוספים ***n - 2*** בנות ל-***node*** (מספר הבנות הוא ***n***), שכן עבור כל בת (חוץ מהשתיים האחרונות), מוסיפים ***node*** ל-CYK matrix.

שאלה 2.2:

אלגוריתם הבינריזציה נדרש עבור הרצת אלגוריתם ה-CYK, שכן עבור CYK החוקים נדרשים להיות במבנה CNF. (במימוש שלנו, הוספנו לאלגוריתם תמיכה בחוקים אונאריים).

שאלה 2.3:

* כאשר מבצעים בינריזציה של העצים, כל צומת עם יותר מאח אחד מאבד מה"קונטקסט" שלו – הוא לא זוכר אילו אחים היו לו. כיוון שלקונטקסט יש חשיבות בשפה טבעית, איבוד מוחלט של הקונטקסט של כל צומת פוגע ביכולות הזיהוי הנכון והתיוג
* כאשר משתמשים ב-horizontal markovization, בעצם "מפצלים" חוק יחיד לכמה חוקים – חוק נפרד לכל שכנות של non terminals . למשל, אם נשתמש בדוגמה של מטלת הבית: לפני השינוי, ה-non-terminal X@// יכול היה להפיק- Y2, …, Yk . כעת, קיימים חוקים שונים (X@/Y2/, X@/Y3/, …), וכל חוק יכול להפיק תפוקות שונות. כלומר, הוספת הקונטקסט (קידוד האחים השמאליים), משנה את ה-parsing של המשפט.

**שאלה 3**

**Pre-processing**

1. Save map for lexical rules by the product word
2. Save maps of unary and binary grammar rules, by first RHS argument
3. Create the right upper-half of a matrix and populate it with empty maps
4. Create a map for holding the backtrace (for reconstructing the probable tree)
5. Save ***default-prob*** the average probability of NN tag (for assigning to unknown words)

**PROBABILISTIC-CYK**(Input: ***sentence***)[[1]](#footnote-1)

*Part I: Initialization*

1. cyk-matrix ← new “half matrix”[[2]](#footnote-2) of length LEN(***sentence***) and default value ∞
2. cyk-backtrace ← new empty map

*Part II: iterate over words in sentence*[[3]](#footnote-3)

1. For j ← 1 to LEN(***sentence***)

*Part III: Find lexical productions*

* 1. For each LHS of lexical rule {A | A 🡪 ***sentence***[j-1]}
     1. cyk-matrix[j-1, j, A] ← P(A 🡪 ***sentence***[j-1])
     2. cyk-backtrace[j-1, j, A] ← (-1, ***sentence***[j-1], null)[[4]](#footnote-4)

*Part IV: iterate over unary rules after lexical productions*

* 1. Repeat till convergence or max iteration limit is reached:
     1. For each unary rule A 🡪 B where (j-1,j,B) ∈ cyk-matrix:[[5]](#footnote-5)
        1. If cyk-matrix[j-1,j,A] > P(A 🡪 B) + cyk-matrix[j-1,j,B]:
           1. cyk-matrix[j-1, j, A] ← P(A 🡪 B) + cyk-matrix[j-1,j,B]
           2. cyk-backtrace[j-1, j, A] ← (-2, A null)[[6]](#footnote-6)
  2. for i ← j – 2 down to 0 do:

*Part V: iterate over binary rules, search for each possible split k between i and j*

* + 1. for k ← i + 1 to j-1 do:
       1. for each binary rule R = A 🡪 B C where cyk-matrix[i,k,B] < ∞ AND cyk-matrix[k,j,C] < ∞ :4
          1. if cyk-matrix[i,j,A] > P(R) + cyk-matrix[i,k,B] + cyk-matrix[k,j,C]: // (i.e. R is more probable for LHS = A)

cyk-matrix[i,k,A] ← P (R) + cyk-matrix[i,k,B] + cyk-matrix[k,j,C]

cyk-backtrace[i,j,A] ← (k,B,C)

*Part VI: iterate over all unary rules in cell [i,j]*

* + 1. Repeat till convergence or max iteration limit is reached:
       1. For all unary rules R=A🡪B where {B | (j-1,j,B) ∈ cyk-matrix}:4
          1. If cyk-matrix[j-1,j,A] > P(R) + cyk-matrix[j-1,j,B]:

cyk-matrix[j-1, j, A] ← P(R) + cyk-matrix[j-1,j,B]

cyk-backtrace[j-1, j, A] ← (-2, A null)5

1. return BUILD-TREE(cyk-matrix, cyk-backtrace, start-symbols)

ניתוח סיבוכיות זמן ריצה כפונקציה של אורך המשפט N, ו-G גודל הדקדוק: O(N3∙G)

* הלולאה החיצונית j רצה N איטרציות. הלולאות הפנימיות i,k רצות מספר משתנה של פעמים, כאשר מספר האיטרציות בכל לולאה חסום מלמעלה בערך משתני הלולאות החיצוניות- i,j . המספרים הללו חסומים ב- O(n3). באלגוריתם CYK המקורי עוברים מספר קבוע של פעמים על כל החוקים בדקדוק, כלומר, סיבוכיות האלגוריתם היא- O(n3∙G) . זהו גם זמן הריצה אצלנו, במקרה הגרוע. בפועל, יצרנו אינדקס כך שבפועל לא היה מעבר על כל החוקים בכל איטרציה, מה ששיפר משמעותית את זמני הריצה.

ניתוח סיבוכיות המקום כפונקציה של אורך המשפט N, S מספר הסמלים, ו-G גודל הדקדוק: O(N2∙S + N + G)

* במימוש שלנו, משפט הקלט וכל החוקים מוחזקים בזיכרון – O(N+G) .
* המטריצה מכילה טבלאות גיבוב שבכל טבלה יש לכל היותר S כניסות – O(N2∙S) .
* טבלת ה-backtrace היא טבלת גיבוב המכילה מסלולי parse שונים, ובפרט גם את מסלול ה-parse בעל ההסתברות הגבוהה ביותר. הטבלה בנויה ממפתחות וערכים של שלשות (triplets). המפתח הוא שלשה של תא בטבלת ה-CYK וסמל (LHS של חוק), והערך הוא שלשה של מספר ושני סמלים. המספר יכול להיות: (א) -1 בשביל לייצג חוק לקסיקלי, (ב) -2 בשביל לייצג חוק אונארי, (ג) מספר שלם לא-שלילי בשביל לייצג "נקודת פיצול" של חוק בינארי מסמל אחד השולט (dominates) על הטווח [i,j], לשני "סמלים בנים" השולטים על הטווחים- [i,k] ו- [k,j]. לפיכך, טבלת ה-backtrace תכיל N2∙S כניסות לכל היותר (שני המשתנים הראשונים של שלשת המפתח הם- i ו-j אשר ערכם נע בין 0 ל-N, והמשתנה השלישי בטבלה הוא סמל מתוך S) – O(N2∙S)

**שאלה 4**

שאלה 4.1: התוצאות עבור h=-1,0,1,2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **h** | **Tagging accuracy** | **Bracketing FMeasure** | **Bracketing Precision** | **Bracketing Recall** |
| **0** | 78.42 | 55.57 | 59.71 | 51.96 |
| **1** | 82.32 | 58.60 | 60.05 | 57.21 |
| **2** | 84.13 | 59.88 | 61.09 | 58.71 |
| **1-** | 82.39 | 57.60 | 57.93 | 57.27 |

שאלה 4.2: סיווג טעויות:

* מצאנו הרבה שיבושים לקראת סוף של יצירת העץ (כלומר, לקראת שורש העץ). למשל, קיימת נטייה לצמצום מספר הבנים שיש לבן של TOP, ביחס ל-gold.
* שגיאות רבות בעת תיוג NP, בעיקר החלפה של NP ב-VP ו-ADJP. שגיאה זו גוררת תיוג לא נכון של NN (הבן של NP).
* בחוקים שיש בהם זוג סמלים זהים אחד ליד השני, אנו מוצאים שיבושים בניתוח העץ. דוגמה: S 🡪 NP VP **PP PP** SBAR yyDOT, מנותח כ- S 🡪 S yyCM CC S yyDOT

שאלה 4.3: שיפורים אפשריים:

* אסטרטגיה מושכלת יותר עבור מילים לא ידועות (במקום לתייג NN כברירת מחדל).
* לגבי השגיאות בתיוג של NP, בחינת השגיאות מראה שהוספת ההקשר של האב (מרקוביזציה אנכית) יכולה לסייע בהפחתת שגיאות התיוג.

**שאלה 5**

שאלה 5.1: על מנת לשפר את תוצאות ה-parser, ביצענו:

1. מרקוביציה אנכית (vertical markovization): לכל צומת קודדנו מי הוא אביו. הפעולה הזו מוסיפה לכל צומת עוד מידע / קונטקסט, אשר משמש את ה-parser בבחירה יותר מדויקת של עץ. החיסרון שבשינוי כזה הוא בהגדלת ההסתברות למעברים בלתי מוכרים. כלומר, כל שכללי הדקדוק יותר ספציפיים ועם יותר מידע, נדרש הרבה יותר training data **בשביל "לכסות" את כל הניואנסים.**
2. **אסטרטגיה חדשה לטיפול במלים לא ידועות: בדומה לתיוג באמצעות** ngrams**, בשלב עיבוד מקדים המרנו מילים לא ידועות בקבוצת האימון ל-**UNK **ולמדנו את התחביר.**

**שאלה 5.2:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **h** | **Tagging accuracy** | **Bracketing FMeasure** | **Bracketing Precision** | **Bracketing Recall** |
| **0** | 88.71 | 62.33 | 58.87 | 66.23 |
| **1** | 89.42 | 65.63 | 61.86 | 69.89 |
| **2** | 89.47 | 66.32 | 62.43 | 70.72 |
| **1-** | 89.47 | 66.34 | 62.19 | 71.09 |

**שאלה 5.3: סיווג בעיות:**

* **ניכר שיפור בבעיות משאלה 4 (שיבוש תיוגים לקראת שורש העץ ושיבושים סביב** NP**).**
* **עדיין יש נטייה לעץ "להתפצל" לשני ענפים במקום לשמור שורה ארוכה של בנים.**

**שאלה 5.4: שיפורים עתידיים:**

* **שקלול החשיבות של הקונטקסט כתלות במרחק: אובייקט "קרוב" לאלמנט אותו מתייגים יקבל משקל גבוה יותר מאובייקט "רחוק". לצורך כך יהיה צורך להגדיר מדד לקירבה (שאיננו בהכרח מספר המילים בין האלמנט המתוייג לאובייקט הקונטסט).**
* **בנוסף לחוקים הרגילים, חוק של 3 רמות לדוגמה: S->NP->NN , (ושקלול המעברים).**
* **לגבי התיוגים הלא נכונים של NP שגורים תיוג NN – יש לגבש שיטה שתבצע פיזור יותר מדויק של ההתפלגות.**

1. כל הסתברויות ה-P באלגוריתם הן בעצם- מינוס לוג של ההסתברות שחושבה בפועל. [↑](#footnote-ref-1)
2. ראה Figure 13.4 בפרק 13 בספר הקורס (עמ' 8). האינדקסים באלגוריתם מתייחסים גם הם לאינדקסים של התאים בטבלה. [↑](#footnote-ref-2)
3. זה לא קטע קוד נפרד (שאר הקטעים מקוננים בלולאה הזו), אבל מטעמי בהירות אנו מתייחסים ללוגיקה / משמעות של הלולאה הזו בנפרד. [↑](#footnote-ref-3)
4. המספר -1 מסמן את ה-triplet כחוק כ-triplet טרמינלי [↑](#footnote-ref-4)
5. חיפוש משופר בזכות אינדוקס החוקים לפי הארגומנט השמאלי של ה-RHS שלהם (חוקים אונריים ובינאריים בנפרד). [↑](#footnote-ref-5)
6. המספר- -2 מסמן את ה-triplet כתוצר של חוק אונרי, ולפיכך הערכים בו מתפרשים באופן קצת שונה. [↑](#footnote-ref-6)