עיבוד שפה טבעית - תרגיל בית רטוב 1

אופיר צפריר 305219768 תומר שווקי 203764618

אימון:

מאפיינים:

המאפיינים בהם עשינו שימוש מפורטים כאן כאשר <word> היא מילה מסויימת באוצר המילים שנבנה עבור סט האימון ו- <tag> הוא תיוג מסוים שקיים בסט האימון. עבור כל מאפיין שאנו מתארים כאן אנחנו מחזירים וקטור בינארי באורך כל הקומבינציות האפשריות שראינו בסט האימון ואנחנו מדליקים רק את הביטים עבורם התנאי מחזיר ערך חיובי.

1. Word-Tag Feature: if curr_word=<word> and curr_tag=<tag>, Size=#word*#tags
2. Prefix-Tag Feature: if curr_word_prefix=<prefix> and curr_tag=<tag>, Size=#prefix*#tag
   כאשר <prefix> היא תחילית מילה שנראיתה בסט האימון, אנחנו מייצרים תחיליות עד אורך 4.
3. Suffix-Tag Feature: if curr_word_suffix=<suffix> and curr_tag=<tag>, Size=#suffix*#tag
   כאשר <suffix> היא סיומת מילה שנראיתה בסט האימון, אנחנו מייצרים סיומות עד אורך 4.
4. TagTrigram-Tag Feature: if prev_prev_tag, prev_trag = <t_{i-2}, t_{i-1}> and curr_tag=<tag>, Size=#tag^3
   כאשר <t_{curr-2}, t_{curr-1}> הם זוגות טאגים שהופיעו בסט האימון.
5. TagBigram-Tag Feature: if prev_tag=<t_{i-1}> and curr_tag=<tag>, Size=#tag^2
6. TagUnigram-Tag Feature: if curr_tag=<t_i>, Size=#tag
7. PreviousWord-Tag Feature: if prev_word=<word> and curr_tag=<tag>, Size=#word*#tags
8. 2PreviousWord-Tag Feature: if prev_prev_word=<word> and curr_tag=<tag>, Size=#word*#tags
9. NextWord-Tag Feature: if next_word=<word> and curr_tag=<tag>, Size=#word*#tags
10. 2NextWord-Tag Feature: if next_next_word=<word> and curr_tag=<tag>, Size=#word*#tags
11. StartCapital-Tag Feature: if first letter in curr_word is capital and curr_tag=<tag>, Size=#tags
12. ContainCapital-Tag Feature: if curr_word contains capital letter and curr_tag=<tag>, Size=#tags
13. AllCapital-Tag Feature: if curr_word letters are all capital and curr_tag=<tag>, Size=#tags
14. Number-Tag Feature: if curr_word is a number and curr_tag=<tag> Including comma separated numbers, Size=#tags
15. CapitalSentence-Tag Feature: if curr_word begins with a capital letter and is at the beginning of a sentence and curr_tag=<tag>, Size=#tags
16. ContainNumber-Tag Feature: if curr_word contains a number and curr_tag=<tag>, Size=#tags
17. ContainHyphen-Tag Feature: if curr_word contains hyphen and curr_tag=<tag>, Size=#tags
18. ContainDot-Tag Feature: if curr_word contains dot and len(curr_word)>1 and curr_tag=<tag>, Size=#tags
19. ContainApostrophe-Tag Feature: if curr_word contains apostrophe and curr_tag=<tag>, Size=#tags

סך הכל במודל 1 וקטור המאפיינים הוא ממימד 2,094,748 כאשר אנו ממסכים מילים שמופיעות לכל היותר פעם אחת.
סך הכל במודל 2 וקטור המאפיינים הוא ממימד 424,548 כאשר אנו לא ממסכים מילים כלל. במודל 2 אנו משתמשים באותם המאפיינים בהם אנו עושים שימוש במודל 1. לאחר מספר ניסויים בהם ניסינו להוריד מאפיינים שונים ראינו כי אנו מקבלים את התוצאות הטובות ביותר עבור כל המאפיינים.

האצת תהליך האימון:

לאחר כתיבת הקוד הראשוני ראינו שתהליך האימון ארוך מדי (סדר גודל של יום) לכן עשינו את השיפורים הבאים כדי להאיץ את תהליך האימון:

1. מקביליות: מיקבול תהליך הוצאת המאפיינים של סט האימון.
2. חישוב פונקצית מחיר ונגזרתה: חישוב הפונקציות נעשה בפעולות מטריציות בלבד ללא חישוב איטרטיבי כלל (לולאות).

3. ייצוג יעיל של מטריצות: על מנת לשמור מטריצות של סט האימון בזיכרון אנו משתמשים בייצוג sparse של מטריצות. עבור בניית מטריצה אנו משתמשים במטריצה מסוג scipy.sparse.coo_matrix ועבור ביצוע חישובים אנו עוברים למטריצה מסוג scipy.sparse.csr_matrix.

<u>היפר-פרמטרים:</u>

1. רגולריזציה: מצאנו שיש tradeoff בין זמן האימון להתכנסות למינימום כתלות בערך הרגולריזציה (lambda). לאחר שבדקנו מספר ערכים מצאנו כי עבור מודל 1 $lambda=10^{-3}$ ועבור מודל 2 $lambda=10^{-4}$ אנו מקבלים איזון טוב בין שני הגורמים הללו.
2. גודל חיפוש אלומה: עשינו מספר ניסויים עם גדלים שונים של חיפושי אלומה ומצאנו שיש tradeoff בין דיוק המודל לזמן ההסקה, ראינו כי עבור חיפוש אלומה בגודל 1 יש איבוד זניח בדיוק לעומת רווח גדול מאוד בזמן הריצה.
3. מיסוך מילים נדירות: בדקנו את האפשרות להסיר מאוצר המילים מילים שמופיעות מעט פעמים, מצאנו שעבור מודל 1 אנו מקבלים את התוצאות הטובות ביותר כאשר אנו ממסכים מילים אשר מופיעות פעם אחת לכל היותר. עבור מודל 2 אנו מקבלים את התוצאות הטובות ביותר כאשר אנו לא ממסכים כלל מילים.

פירוט חומרה וביצועים מופיעים בסוף הדו"ח

<u>הסקה:</u>

אנו משתמשים באלגוריתם ויטרבי כפי שהוצג בכיתה עם השיפורים הבאים:

1. חיפוש אלומה: בכל שלב של ההסקה אנו מעבירים לשלב הבא רק את ההיפותזות עם ההסתברות הגבוהה ביותר.
2. מקביליות: כדי להאיץ את התיוג על סט הבוחן ועל קבצי התחתרות אנו מבצעים תיוג על מספר משפטים במקביל.
3. log של π: כדי למנוע בעיות נומריות בזמן ההסקה וגם כדי להאיץ ביצועים אנו מחשבים את log של הכניסות בטבלה π. אין פגיעה בנכונות של המודל מכיוון ו- log היא מונוטונית עולה. הרווח בביצועים נובע ממעבר מפעולות כפל לפעולות חיבור.

<u>מבחן:</u>

עבור מודל 1 מטריצת הבלבול מצורפת בקובץ ההגשה בשם confusion_matrix.txt. לאחר בחינת מטריצת הבלבול אנו רואים שעיקר השגיאות של המודל הן בין תארים (JJ) לשמות עצם (NN). אנו מציעים להוסיף מידע נוסף למודל על תארים ושמות עצם ידועים בשפה האנגלית, ניתן לבנות מידע זה על סמך סט האימון, בנוסף ניתן להוסיף חוקים ידועים בשפה האנגלית למודל כדי לדעת מה ההסתברות לרצף מסויים בשפה האנגלית.

בנוסף ביצענו אנליזה נוספת על המילים בהן אנו טועים בתיוג ומהן הסקנו איזה פיצ'רים עלינו להוסיף על מנת לשפר את הדיוק, לדוגמא מקף באמצע מילה, פסיק המופיע כחלק ממספר, התייחסות יותר רחבה לאותיות גדולות.

כדי לבחון את מודל 2 השתמשנו בשיטה הנקראת cross validation בה חילקנו את סט האימון ל-7 חלקים (100 משפטים בכל חלק) כאשר בכל פעם חלק אחר שימש כסט הבוחן. לבסוף אחוז הדיוק אליו התייחסנו הוא הממוצע של אחוזי הדיוק עבור כל איטרציה. השתמשנו בשיטה זו מכיוון והקובץ מכיל מעט משפטים ולכן לא רצינו לחלקו באופן נוקשה לסט אימון וסט מבחן. לבסוף אנו משתמשים בכל הדוגמאות כדי לאמן את המודל אשר בעזרתו אנו מתייגים את קובץ התחרות.

צילום מסך של תהליך האימון ובחינה של מודל 1:

```
Loading data
Generate words and tags lists
Remove rare words from vocabulary
Extract features from training data
Loading time:  59.894131898880005
Running training
Optimize
      fun: 131.55989494852182
 hess_inv: <2094748x2094748 LbfgsInvHessProduct with dtype=float64>
      jac: matrix([[ 0.00081933,  0.00053161,  0.00054363, ..., -0.00011194,
         -0.00036383, -0.00093712]])
  message: b'CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH'
     nfev: 346
      nit: 201
   status: 0
  success: True
        x: array([ 0.8193297 ,  0.53160778,  0.54362543, ..., -0.11193821,
       -0.36383182, -0.93711785])
Training time:  1274.2074868679047
Running Viterbi
Viterbi time:  143.37708735466003
tagger accuracy: 0.9530286390132635
```

תחרות:

כאשר אימנו את המודלים עבור התחרות בחרנו את ההיפר פרמטרים אשר נתנו לנו את התוצאות הטובות ביותר על סט הבוחן. למעשה המודלים שלנו למדו באופן עקיף את סט הבוחן ומותאמים אליו. לכן אנו צופים להבדלים בתוצאות בין סט הבוחן לקבצי התחרות. לעומת זאת אנו משערים כי הדאטה בקובץ התחרות נלקח מאותה ההתפלגות של סט הבוחן ולכן מודל שמספק ביצועים טובים על קובץ המבחן יספק ביצועים טובים על קובץ התחרות. לכן אנו משערים כי נקבל ירידה באחוז הדיוק של ~1% לעומת סט הבוחן כלומר נקבל דיוק של 94%.

בניסיון לשפר את תוצאות התיוג על קבצי התחרות ניסינו לבצע model ensembling, בשיטה זו אנו מאמנים מספר מודלים על סט האימון ומבצעים תיוג על קובץ התחרות בעזרת כל אחד מהמודלים, למעשה כל מודל נותן הצבעה לתיוג שהוא חושב שהכי מתאים. לאחר מכן אנו קובעים את התיוג של המילה על פי התיוג שקיבל את מספר ההצבעות הגבוה ביותר. אך מכיוון ולא קיבלנו שיפור משמעותי כאשר בחנו את השיטה על קובץ המבחן החלטנו לוותר עליה בסוף ולא להגיש אותה.

בנוסף כדי לבצע הסקה על קבצי התחרות באופן מהיר מימשנו ממשק שמירה וטעינה של מודלים מאומנים.

חלוקת עבודה:

את תכנון המערכת ביצענו יחד, כלומר איך לעבור על הדאטה, איך להוציא מאפיינים עבור מילים, איזה מאפיינים להוציא, כיצד לבצע את האימון, כיצד לבצע את ההסקה נעשה בחשיבה משותפת.

לאחר מכן פתחנו git repository ועבדנו על הקוד במקביל. מכיוון וכל אחד מאיתנו נגע בכל מודול של הפרויקט והיה שותף לתכנון, לכתיבת קוד ודיבוג קשה לנו לעשות הפרדה מוחלטת בין העבודה של שנינו.

חומרה:

אנו מאמנים את המודלים שלנו על 2 מערכות:

1. מערכת 1: Intel core i7-4770 בעל 4 ליבות, 16GB ראם, מערכת הפעלה Ubuntu16.
2. מערכת 2: 2x Intel Xeon E5-2687 סך הכל 22 ליבות, 128GB ראם, מערכת הפעלה Ubuntu16.

ביצועים:

את התהליך אנו מחלקים ל-3 חלקים, זמן הוצאת המאפיינים, זמן האימון (הפעלת אלגוריתם L-BFGS), ביצוע הסקה וחישוב דיוק:

מערכת 1	מאפיינים (ד')	אימון (ד')	הסקה (ד')	דיוק
מודל 1	3.7	26.7	5.5	95.3%
מודל 2	0.5	2.1		95.7%

מערכת 2	מאפיינים (ד')	אימון (ד')	הסקה (ד')	דיוק
מודל 1	1.0	21.24	2.4	95.3%
מודל 2	0.16	1.17		95.7%

1. עבור דיוק של מודל 2 אנו מדווחים את הדיוק שקיבלנו ב- 7fold-cross validation.
2. עבור מודל 2 איננו מבצעים הסקה מכיוון ואין סט ולידציה.