מגישים: יהונתן שאקי, 204920367. אור שחר, 209493709. יש לשים לב שחלקים מ-"חלק 1" רלוונטים לתשובה לכל החלקים (מודגש).

חלק 1:

יצרנו תג שמסמן תחילת משפט ותג שמסמן סוף משפט. הוספנו שני תגי התחלה בתחילת כל משפט ושני תגי סיום בסוף כל משפט.

הוספנו תג למילים שלא קיימות ב-train, וכדי שהרשת תכיר את התג הזה ותוכל לאמן אותו (ליצור לו embedding מוצלח וכנראה שונה משל שאר המילים) ועליו (להתמודד איתו כשהוא מופיע) החלפנו בהסתברות נמוכה מילים ב-train לתג הזה (מעין mask).

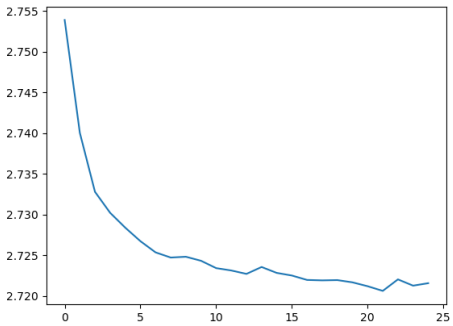
כעקרון, היינו רוצים, בכל מעבר על ה-train data, לעשות mask שונה; הבעיה שנראה שלעשות את זה באופן דינמי יכול להאט משמעותית את האימון. לכן, החלטנו לשכפל את כל ה-train data כמות קבועה של פעמים ובכל שכפול להחליף מילים אחרות. החלטנו לשכפל 3 פעמים. חשוב לשים לב שהגרפים בהתאם ולכן האימון נראה מהיר יותר מאשר שהיה קורה ללא השכפול (כל נקודה היא אחרי מעבר על כל ה-augmented data). האמור לעיל בוצע בכל הסעיפים בתרגיל.

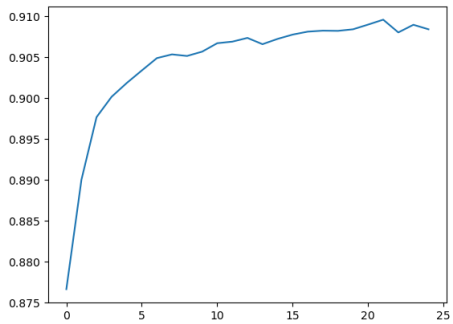
ארכיטקטורה: מתחילים מ-embedding, ואז dropout של 0.3, ואז שכבת tanh בגודל 500, ואז softmax בגודל הנדרש (36 ל-pos ו-5 ל-ner). ארכיטקטורות עמוקות יותר (עם leakyRELU בנוסף) לא תרמו לביצועים.

הרצנו 25 אימונים על כל ה-augmented data עבור pos ו-50 עבור ner (בגלל של-ner יש פחות data אימון)

גרפים:

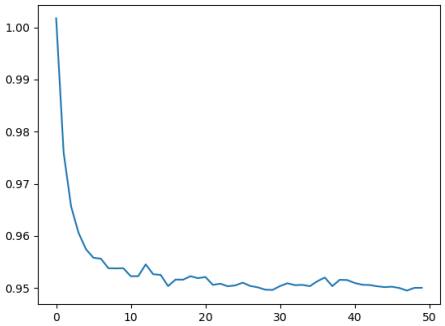
על pos:

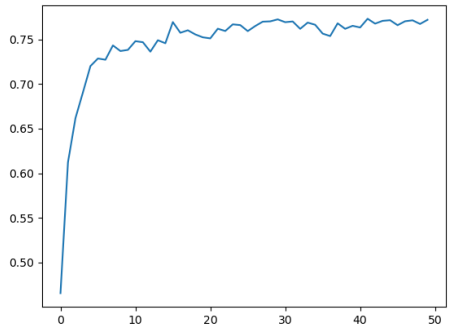




(הדיוק בסופו של דבר היה 0.9)

על ner:





(בסופו של דבר הדיוק היה 0.76)

חלק 2:

```
Most similar words to dog:
muhammad 0.7034705853461563
labour 0.6998496216113942
intersection 0.6765602612696248
throw 0.6382018205781824
sultan 0.6311740208391059

Most similar words to england:
previously 0.9009473338245269
rule 0.8500773404200397
australia 0.8063409338273496
property 0.7983477530776709
instead 0.7752753846783795

Most similar words to john:
george 0.9220484429316901
event 0.8988161740885584
section 0.894692282777918
william 0.8894472752280504
james 0.8782858999808393

Most similar words to explode:
lifeguard 0.600053586434226
recreating 0.5932232016072673
meadow 0.5543629962768829
currituck 0.5335439262211472
ruan 0.5227968700698917

Most similar words to office:
board 0.6681674892325012
court 0.6197735812307626
lot 0.6174371328485805
larger 0.6166757055712344
specific 0.6033201253924071
```

חלק 3:

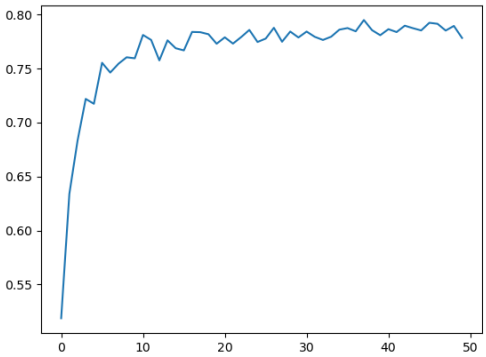
כמו בחלק אחד, גם כאן היה תג של Unknown. המרנו את כל המילים ל-lowercase לפני שבדקנו אותם מול ה-embedding.

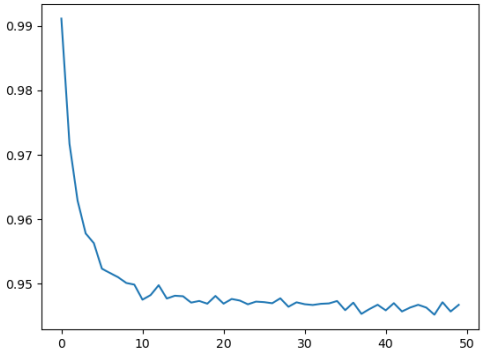
הארכיטקטורה הייתה זהה לחלק 1.

מילים שהופיעו ב-train אבל לא ב-embedding קיבלנו וקטור חדש, בדיוק כמו בסעיף 1. למעשה, אפילו הפכנו מילים שלא הופיעו ב-train אבל כן ב-embedding ל-unknown; זה בגלל שאנחנו ממשיכים לאמן את ה-embedding, ומילים שהיו קיימות רק במקור לא יתאמנו ו-"ישארו מאחור".

גרפים: היה שיפור! קטן אבל היה.

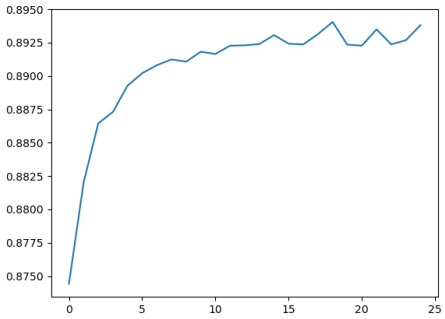
Ner:

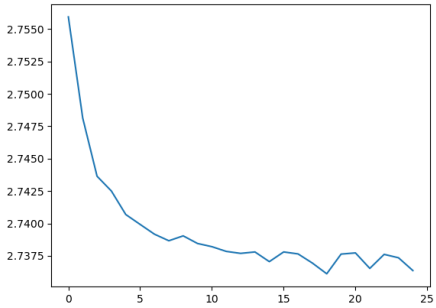




(בערך 0.785 בסוף, שיפור של קרוב ל-3 אחוז!)

:Pos





כאן דווקא לא היה שיפור, והתוצאה הייתה קרובה מאוד לזאת מחלק 1, 0.9~.

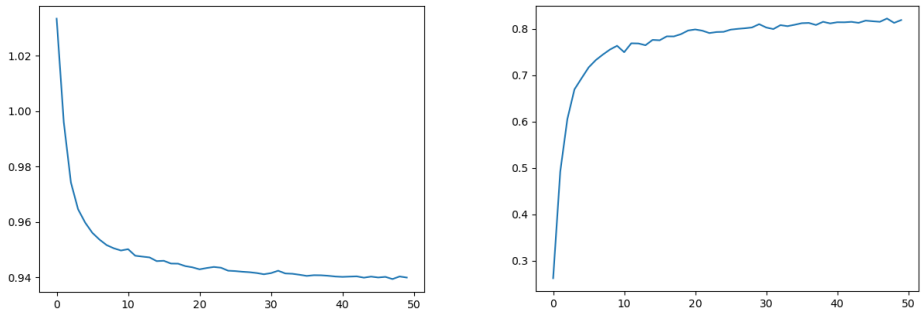
חלק 4:

ההמרה ל-subwords נעשתה באופן דומה ל-embedding הרגיל – גם כאן, אם הייתה תחילית\סיומת שלא ראינו כמוה, החלפנו אותה ב-unknown. האימון על unknown נעשה באופן הבא: עבור מילים באימון שלא נעשה להם mask, גם התחלית והסיומת היו ללא mask. רק אם מילה נבחרה להיות masked, הייתה הסתברות (0.2) שהתחילית\הסיומת שלה יהיו גם masked.
(כי במבחן זה גם יהיה ככה – אם מילה מוכרת היא הייתה באימון, ולכן בהכרח גם התחלית והסיומת שלה היו באימון וממליא לא יהיו מוסתרות).

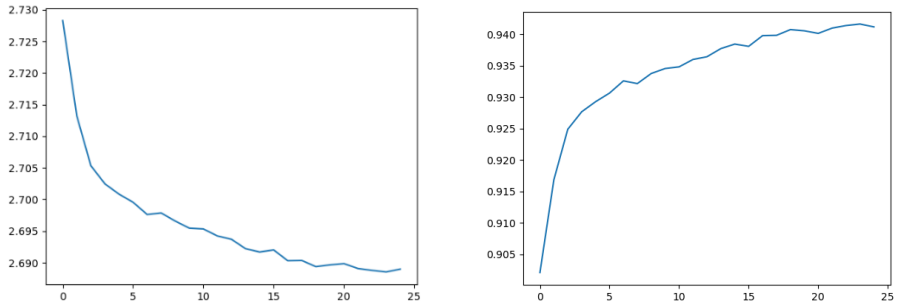
הארכיטקטורה הייתה זהה לחלק אחד, אלא שכאן היה יותר embedding, כפי שנדרש בתרגיל.

גרפים:

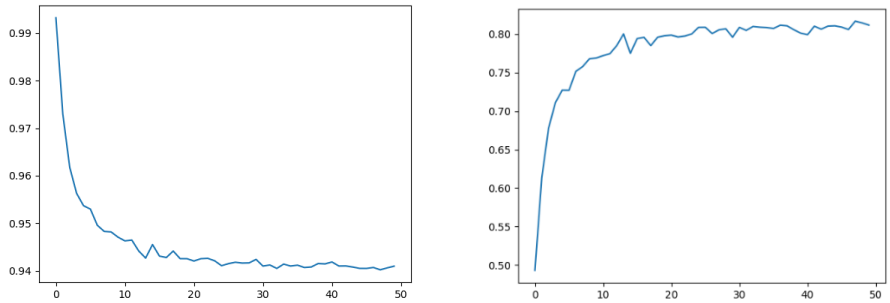
בלי pretrained embedding, ner (בסופו של דבר דיוק של 0.816):



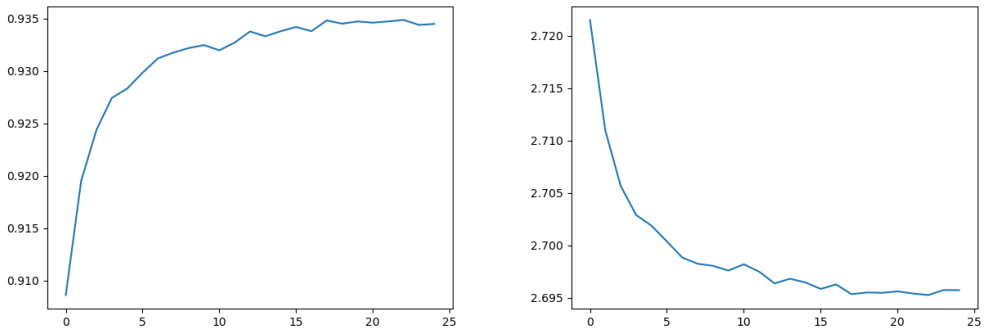
בלי pretrained embedding, pos (בסופו של דבר דיוק 0.94!):



עם pretrained embedding, ner (בסופו של דבר דיוק של 0.814, דומה מאוד ללא pretrained):



עם pretrained embedding, pos (דיוק של 0.936, קרוב לבלי embedding ואפילו קצת פחות טוב



:

ניתוח:

נראה שהטכניקה כאן מועילה עבור שתי המשימות באופן ניכר, בשיפור של כמה אחוזים ברמת הדיוק. השיפור משמעותי יותר מאשר השיפור של השימוש ב-pretrained embedding בחלק שלוש; בנוסף, לא נרשם שיפור נוסף כשהוספנו כאן את ה-pretrained embedding. אנחנו מסיקים שהתרומה של המשימות אינה משלימה, אלא שהוספת prefixes, suffixes טובה יותר מאשר ה-pretrained, ואפילו לא נדרשת לה (אולי כי ה-pretrained התאמן בסיטואציה בה לא היו prefixes/suffixes ולכן הוא תופס הרבה מידע שנכלל בהם ממליא).

חלק 5:

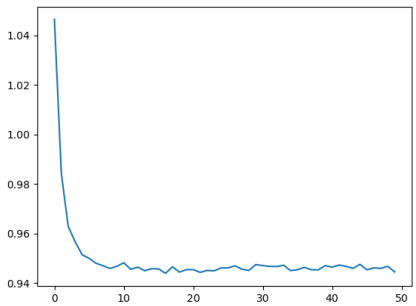
ראשית, ביצענו את ההמרה לתווים באופן הבא. בשביל שיקולים של גדלים קבועים (חשוב בשביל batching ויעילות), הנחנו שכל מילה היא עד אורך קבוע כלשהו (נבחר ל-15) וחתכנו את הסיומת עבור מילים ארוכות יותר. המרנו תווים "ריקים" (שהוספו לסוף ולהתחלה, באופן שווה ככל האפשר, על מנת להשלים ל-15) לתג מיוחד (נפרד עבור תחילת מילה וסוף מילה). לאחר מכן, ביצענו את ה-embedding לכל תו והרצנו כפי שנדרש ותואר במאמר.

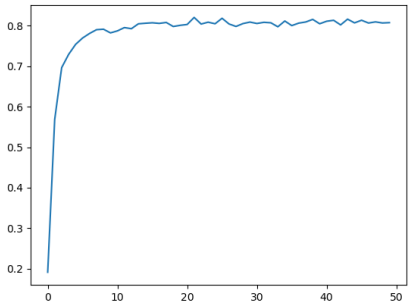
האם היה שיפור: עבור ner הגענו לדיוק על dev של כ-0.81, ועבור pos כ-0.92. כפי שניתן לראות מהשוואה לסעיפים הקודמים, אכן מדובר בשיפור לעומת 1, 3, אבל לא טוב יותר מ-4.

הארכיטקטורה הייתה זהה לחלק אחד, אלא שהוספנו לקלט לרשת העיקרית תוצאה של חלק נוסף של הרשת עם ה-CNN; בתת רשת היה char embedding ראשית, בגודל 30, ל-15 תווים (כפול חמש מילים), כפי שתואר לעיל. לאחר מכן מתבצעת קונבולוציה עם 30 פילטר בגודל שלוש על שלושים, כך שלכל מילה ופילטר מתקבלים 13 תוצאות, עליהם מתבצע poolmax שמביא תוצאה אחת לכל פילטר. בסופו של דבר מתקבלים 150 מספרים שמצטרפים ל-embedding הרגיל של המילים בדרכם לרשת העיקרית. לפני הקונבולוציה יש dropout של 0.5 כדי לצמצם overfitting.

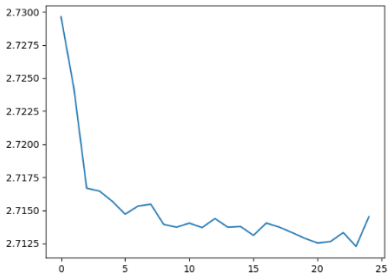
משהו קצת מוזר: במקרים מסוימים (עבור ner) הרשת פשוט חזתה כל הזמן O מהר מאוד. כדי להתגבר על זה, החלטנו באימון הראשון לתת משקל נמוך ל-O, ואז לחזור למשקל אחיד. (עבד כמו קסם)

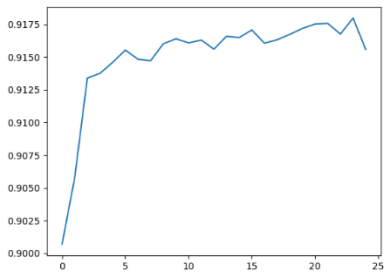
גרפים של ner (בסופו של דבר, דיוק של 0.82):





גרפים של pos (בסופו של דבר דיוק של 0.915):





כאשר משנים פרמטרים:

מוסיפים פילטרים (60 במקום 30): הדיוק דורדר מעט. עבור ner הגענו ל-0.8, ועבור pos ל-0.913, קרוב מאוד לדיוק עם הכמות המקורית.

מורידים פילטרים (15 במקום 30): שוב 0.8 עבור ner. עבור pos 0.888, שזה דרדור משמעותי יחסית.

משנים window size (אני מניח שמתכוונים ל-window size על ה-characters, דהיינו כמות התווים שפילטר רואה): עבור 5 במקום 3: הדיוק היה 0.81 עבור ner (נמוך באחוז מהקודם) ועבור pos היה 0.925 (גבוה בקצת מהקודם). נראה שהאם זה משפר תלוי במשימה, או לחלופין סתם קצת רנדומיזציה והשינוי לא מהותי.

עבור חלון של 7: כאן דווקא הדיוק על ner היה 0.82, בדיוק כמו עם גודל 3. עבור pos, הדיוק היה 0.914, בערך כמו הגודל 3. לסיכום, נראה שהגדלת הפילטר לא משפיע על הביצועים יותר מידי.

לעומת זאת, הקטנה של הפילטר (ל-2, אחרת זה כבר לא ממש פילטר) נותנת 0.81 עבור ner ו-0.913 עבור pos, שזו כן פגיעה מסוימת בביצועים.

הבנת הפילטרים: מצורפים, מתוך מעבר על כל שלשות האימון (ממשימה 4 :) + הוספת ריפוד, ומאחר וכל פילטר הוא בכל מקרה בגודל שלוש) לכל פילטר 5 השלשות עליהם הוא מביא את הציון הכי גבוה, ו-5 עבורם הוא מביא את הציון הכי נמוך. התמונה הראשונה עבור ner והשניה עבור pos.

```
0: ['1/1', '1/2', '3/1', '8/3', '8-1'] ['jaz', 'az', 'azz', 'ziz', 'z']
1: ['zza', 'ujj', 'ozz', 'azz', 'czk'] ['+7', '+15', '+5', '+45', '+35']
2: ['-29', '-22', '-27', '-28', '4-2'] ['aii', 'zbi', 'zai', 'zhi', 'zzi']
3: ['22/', '28/', '29/', '12/', '02/'] ['zia', 'zza', 'zim', 'zam', 'ziz']
4: ['28/', '22/', '263', '269', '23/'] ['xin', 'x', 'xmi', 'xia', 'xon']
5: ['822', '813', '223', '623', '833'] ['azz', 'arz', 'itz', 'icz', 'asz']
6: ['y.q', 'ujj', 'mcq', 'yaq', 'gjo'] ['+', '+15', '+17', '+7', '++3']
7: ['2/3', '1/8', '1/3', '7/8', '3/8'] ['kia', 'afa', 'uia', 'aua', 'aia']
8: ['zej', 'ziz', 'j.j', 'zka', 'zyr'] ['775', '745', '7-5', '735', '755']
9: ['j.j', 'jez', 'zej', 'jka', 'zka'] ['+65', '575', '+35', '55', '555']
10: ['in-', 'iff', 'ift', 'inq', 'ifa'] ['yvy', 'lyb', 'b', "'ve", 'bbb']
11: ['833', '831', '873', '813', '872'] ['hiu', 'aju', 'uzu', 'nzu', 'riu']
12: ['rp.', '0.2', 'pfd', '0.0', 'rps'] ['464', '444', '454', 'zza', 'zzi']
13: ['28:', '26:', '12:', '228', '27:'] ['juh', 'jp', 'jka', 'j.j', 'j.']
14: ['zza', 'ozz', 'azz', 'zzi', 'rza'] ['++3', '+35', '+33', '+27', '+7']
15: ['81-', '818', '71-', '815', '61-'] ['kaz', 'j.j', 'kij', 'kuz', 'jaz']
16: ['53-', '52-', '535', '43-', '33-'] ['ziz', 'j.j', 'j', 'jaz', 'jez']
17: ['022', '2/3', '092', '8/3', '021'] ['zwi', 'zif', 'zai', 'zhi', 'zzi']
18: ['s23', '/32', '/30', 's-1', '/16'] ['nzo', 'hao', 'ico', 'nho', 'oco']
19: ['zka', 'czk', 'szk', 'jko', 'jka'] ['+65', '+5', '+25', '+45', '+35']
20: ['403', '833', '803', '43-', '40-'] ['unj', 'jaz', 'unz', 'ujj', 'ziz']
21: ['822', '+22', '821', '+62', '861'] ['aii', 'az', 'azz', 'jaz', 'ziz']
22: ['13/', '23/', '132', '133', '135'] ['zzi', 'zif', 'zaf', 'z', 'ziz']
23: ['0-5', '0-4', '0-0', '0-3', '075'] ['h.k', 'ijk', '.66', '.cn', ',']
24: ['0.8', '022', '008', '024', '00-'] ['iji', 'ika', 'imi', 'zzi', 'zza']
25: ['02/', '22-', '22/', '03/', '23-'] ['zia', 'zma', 'zpa', 'zzi', 'zza']
26: ['08/', '09/', '70m', '30m', '60m'] ['ii', 'jui', 'ziz', 'uzi', 'zzi']
27: ['8-2', '982', '9-2', '962', '892'] ['nij', 'j', 'arj', 'muj', 'iej']
28: ['534', '434', '413', '114', '524'] ['ija', 'ka', 'ika', 'jka', 'kka']
29: ['523', '423', '828', '223', '123'] ['jaz', 'z', 'ziz', 'unz', 'ujj']
```

נראה שיש פילטרים שמביאים ציונים גבוהים בעיקר לשלשות של מספרים (11, 28, 29) כמה פילטרים מביאים ציונים גבוהים גם לדברים מבוססי מספרים, אבל לא רק; למשל, 7 מביא ציונים גבוהים לשברים, 2 מביא ציונים גבוהים למספרים שליליים (או סיום ציון טווח), 16 נראה מתעניין בתחילת טווח, 23 מתעניין בתחילת וסיום טווח ביחד, 13 מביא ציונים גבוהים למספרים שמסתיימים בנקודותיים. 26 נותן ציונים לדקות.

(קצת מוזר שכל כך הרבה דברים קשורים למספרים, האמת)

```
0: ['2%-', '8%-', '%-3', '92%', '45%'] ['-86', '-80', '-87', '-89', '-82']
1: ['%-3', '2%-', '8%-', '92%', '45%'] ['884', '88,', '883', '888', '887']
2: ['&', 'pg&', 'eg&', 'at&', 'nt&'] ['.f.', '.g.', '.c.', '.v.', '.d.']
3: ['wed', 'wad', 'wid', 'h&r', 'weg'] ['18', '138', '183', '173', '188']
4: ["'40", "'90", "'71", "'70", "n's"] ['deu', 'daz', 'dau', 'dez', 'gdu']
5: ['%', 'p&g', 'g&g', 'o&y', 'a&w'] ['663', '664', '620', '-50', '-04']
6: ['&', 'at&', 'nt&', 'pg&', 'eg&'] ['--', 'p53', 'g-2', 'p-5', '...']
7: ['%-3', 'op-', 'p-3', 'p-e', 'p-5'] ['63.', '93.', '53.', '83.', '78.']
8: ['3-d', '92-', '98-', '97-', 'wid'] ['y.j', 'h.j', 'ahy', 'hey', '~e~']
9: ['s&p', 'j&b', 'j.p', '92%', '45%'] ['381', '343', '310', '327', '322']
10: ['wis', 'kes', 'bos', 'mrs', 'hys'] ['pt.', '.d.', '20%', '28%', '%']
11: ['r2-', '3-1', '3-a', '3-t', '3-d'] ['d.s', 's.s', 'sus', 'u.s', '...']
12: ["b'g", '16%', "'n'", 'p&g', 'g&g'] ['.43', '.13', '.73', '.53', '%-3']
13: ['53.', 'm8.', '56.', '3.8', '3.0'] ['say', 'sa', 'ss', 's', '~s~']
14: ['.49', '649', '.99', '699', '949'] ['sos', 'sus', 'uss', '-us', 'sms']
15: ['s&l', 's&p', '&', 'pg&', 'eg&'] ['.f.', '.g.', '.c.', '.v.', '.d.']
16: ['%', '28%', '50%', '60%', '20%'] ['46.', 'h.h', 'h.', '.h.', '...']
17: ['j&b', 'j&l', 'jr.', 'p&g', "j'a"] ['343', '310', '327', '3-7', '322']
18: ['-0.', 'q.', '88.', 'ft.', '11.'] ['35,', '73,', '75,', '55,', '33,']
19: ['eks', 'sks', '%', 'cks', 'rks'] ['50.', '57.', '53.', '29.', '20.']
20: ['n.y', 'm.w', 'e.w', 'c.w', 'i.w'] ['52%', '72%', '50%', '%', '53%']
21: ['j&l', 'j&b', 'g&e', 'g&g', 'd&b'] ['5,7', '307', '6-7', '327', '3-7']
22: ['ngs', '%', 'ogs', 'gsx', 'gst'] ['.23', '.63', '.13', '.53', '.33']
23: ['sri', 'smi', 'aqi', '%', '%-3'] ['.73', '.53', '.31', '.80', '.78']
24: ['92%', '89b', '29/', '19/', '4ti'] ['~s~', 'eps', 'ess', 'e-s', 'ees']
25: ['.r.', '.j.', '.y.', 'sr.', '.e.'] ['046', '796', '586', '0-6', '896']
26: ['98.', '92.', '94.', '58.', '68.'] ['--', '-', '-a-', 'de-', 'ed-']
27: ['ing', 'y&r', 'g&g', 'p&g', 'ngu'] ['h.j', 'h.', '.h.', 'hd.', '...']
28: ['j&b', 'd&b', 'h&r', 'p&g', '&sa'] ['.28', '548', '.68', '.78', '008']
29: ["a's", "'s", "o's", "n's", "r's"] ['.c.', '.v.', '.d.', 'el.', 'ec.']
```

אכן נראה שעל pos נלמדו דברים קצת שונים! למשל, יש פילטר שמזהה סימן שיכות (29), פילטר שמזהה ing (נותן לזה ציון הכי גבוה, 27) אחד שמסמן מספרים ואז נקודה (26) אחד לאחוזים (16) ואחד שמשום מה מזהה w_d כאשר במקום ה-_ יכול לבוא כל תו (כנראה) (3). עוד אחד מזהה זוג נקודות ובניהם תו (25).

יתכן שהסיבה לשוני היא בעקבות המשימות עצמם, או בגלל שה-train data שונה. כדי לבדוק את זה, נצטרך train data עם משפטים זהים ותגיות נפרדות לכל אחת מהמשימות.