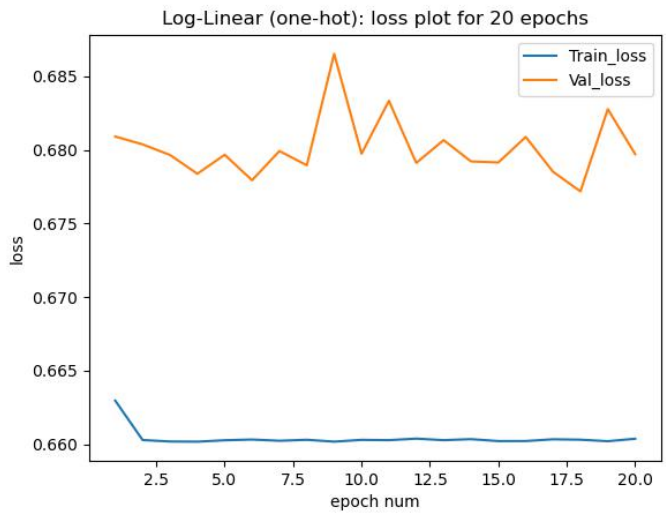
עיבוד שפה טבעית - תרגיל 3

דניאל אזולאי ת"ז 311119895

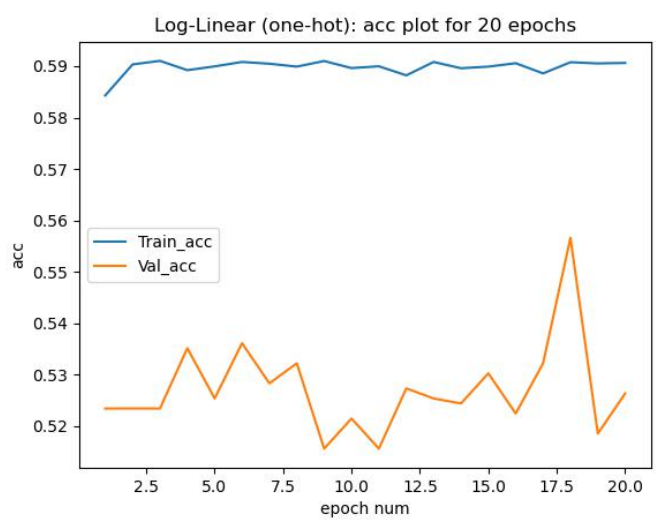
אסף שול ת"ז 207042714

12 בדצמבר 2022

(6) (a) גרף ה־loss עבור train,validation כתלות במספר ה־epoch:



(b) גרף ה־accuracy עבור train,validation כתלות במספר ה־epoch:



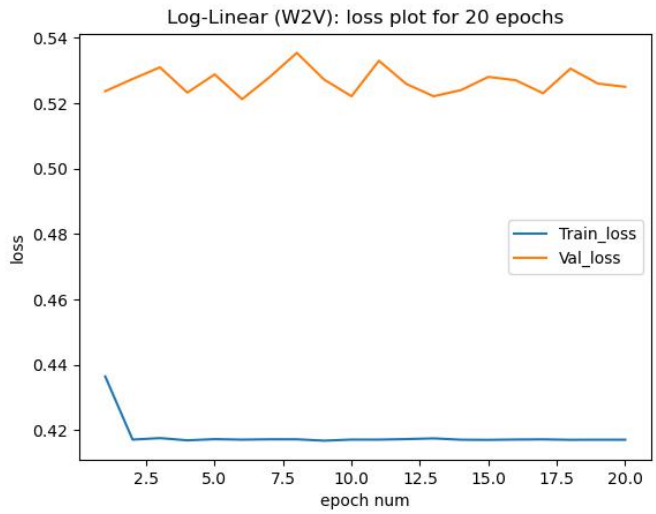
ערכי ה־loss וה־accuracy שהתקבלו על הטסט, וערכי ה־accuracy על הקבוצות המיוחדות הנתונות:

```
==================================================
Log-Linear (one-hot) Test results:
     accuracy (test) = 0.5576, loss (test) = 0.673
     accuracy (neg ) = 0.4839
     accuracy (rare) = 0.28
==================================================
```

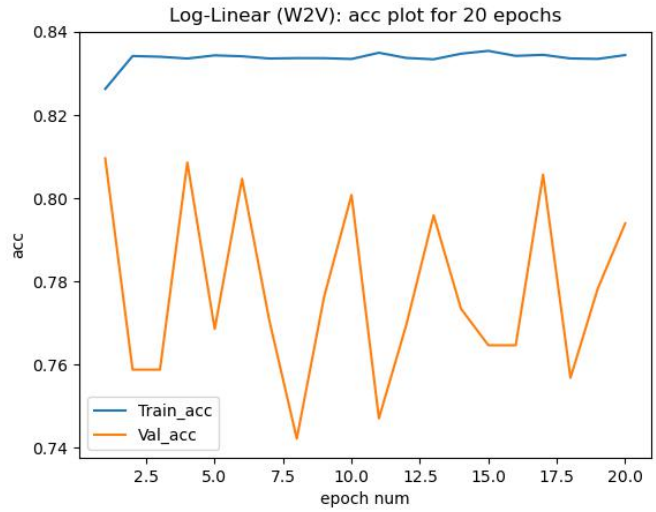
<u>התוצאות עבור הקבוצות המיוחדות</u>

עבור מילים נדירות:

(7) גרף ה־loss עבור train,validation כתלות במספר ה־epoch:



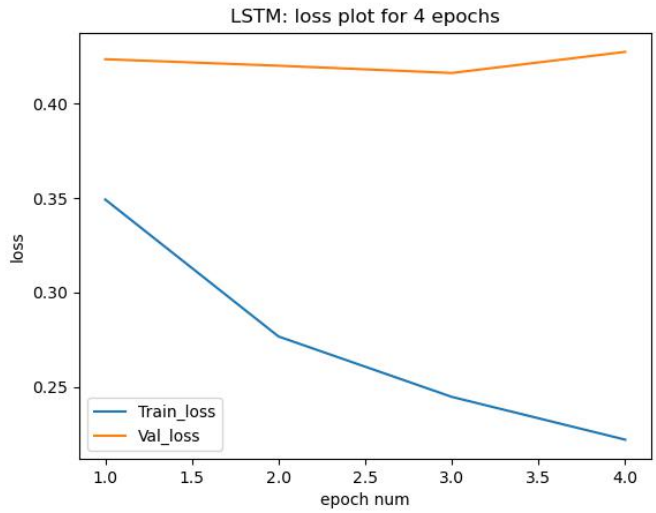
גרף ה־accuracy עבור train,validation כתלות במספר ה־epoch:



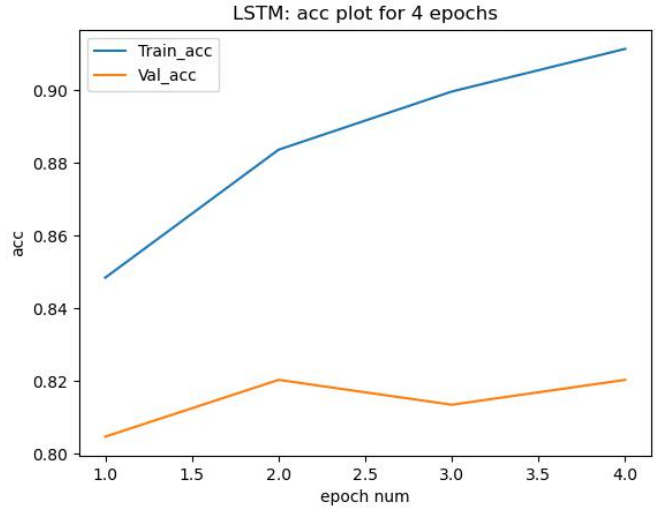
ערכי ה־loss וה־accuracy שהתקבלו על הטסט, וערכי ה־accuracy על הקבוצות המיוחדות הנתונות:

```
==================================================
Log-Linear (W2V) Test results:
     accuracy (test) = 0.8223, loss (test) = 0.4877
     accuracy (neg ) = 0.5645
     accuracy (rare) = 0.7
==================================================
```

(8) גרף ה־loss עבור train,validation כתלות במספר ה־epoch:



גרף ה־accuracy עבור train,validation כתלות במספר ה־epoch:



ערכי ה־loss וה־accuracy שהתקבלו על הטסט, וערכי ה־accuracy על הקבוצות המיוחדות הנתונות:

```
==================================================
LSTM Test results:
     accuracy (test) = 0.8711, loss (test) = 0.3278
     accuracy (neg ) = 0.629
     accuracy (rare) = 0.84
==================================================
```

(9)

(1) השוואת one-hot-vector מול word2vec .

- מי מספק תוצאות טובות יותר?

מודל word2vec מספק תוצאות טובות יותר (loss משמעותית נמוך יותר ו־accuracy משמעותית גבוה יותר),
בכל שיערוך (על כל אחד מהסטים - train, validation, test).

- הסבר אפשרי לכך:

זה יכול לקרות כתוצאה מסיבות רבות, הסבר אפשרי אחד הוא שהמיפוי למימד נמוך, שבו יש "אובדן" מסויים של מידע, תופס טוב יותר את הסנטימנט של משפט ומפחית "רעש" אפשרי בדגימות, רעש שעלול להתקבל ממילים שלא עוזרות למודל בניתוח הסנטימנט. וכך לאחר המיפוי מקבלים שמעט הפיצ'רים שנותרו הם אינדיקטיבים יותר לסנטימנט.

(2) השוואת המודלים הלוג־לינארים מול LSTM:

- מי מספק תוצאות טובות יותר?

מודל LSTM מספק תוצאות טובות יותר מ־word2vec (ובהתאם גם מ־one-hot-vector שהשיג תוצאות עוד פחות טובות).
בכל שיערוך קיבלנו loss נמוך יותר ו־accuracy גבוה יותר (על כל אחד מהסטים - train, validation, test),

- הסבר אפשרי לכך:

התוצאות של LSTM טובות יותר ככל הנראה בגלל ש־LSTM הוא מודל יותר אקספרסיבי ממודלים לוג־לינארים, בגלל המבנה המיוחד שלו: הוא בנוי משכבות של תאי LSTM שמהווים hidden layer. שכבה זו היא דו כיוונית, מה שמאפשר מידול קשרים שמשפיעים על הסנטימנט בשני הכיוונים של המשפט.
ובכלל במודל זה יש משמעות לסדר הופעת המילים במשפט, לעומת המודלים האחרים שמקבלים כקלט ממוצע של קידודי מילים, כי לקיחת הממוצע והכנסתה כאינפוט גורמת לאיבוד של את סדר המילים במשפט.
מסיבות אלה (כנראה), המודל מצליח ללמוד טוב יותר את הקשרים הרלוונטיים לניתוח הסנטימנט.

(3) יחס הסדר החד לעיל נשמר גם עבור הקבוצות המיוחדות. נרחיב בהתאם להוראות עבור כל קבוצה.
עבור negated polarity , מודל one-hot-vector Log-Linear הוא עם התוצאה הכי פחות טובה עם $\text{accuracy} = 0.4839$,
ומודל LSTM סיפק את התוצאה הטובה ביותר עם $\text{accuracy} = 0.629$.
הסבר אפשרי עבור קבוצה זו: גם כאן ניתן להסביר את ההצלחה של LSTM בעזרת האקספרסיביות שלו,
כי נדרש מודל אקספרסיבי על מנת למדל קשרים של שלילה של משפט, ואת קשרים אלו מודל לוג־לינארי פשוט מתקשה לבטא.
הסבר נוסף ולא פחות חשוב, הוא שהמודלים הלוג־לינארים מתעלמים לחלוטין מהסדר של המילים במשפט,
אך ייתכן מאוד שהמיקום של מילים אלו (עם סנטימנט הפוך לסנטימנט של המשפט) הוא משמעותי.
למשל, מודל לוג־לינארי שמקבל וקטור שהוא ממוצע הייצוגים של המילים עבור המשפט bad movie not great,
לא ידע להבדיל בין המשפט הזה למשפט great movie not bad , בעוד מודל LSTM מאומן עשוי לדעת "להפעיל נכונה"
את ההשפעה של not על הפלט שמתקבל לאחר הצירוף not great, כי הייצוג של not נכנס כאינפוט לתא LSTM ,שהפלט שלו נכנס לתא שמקבל כאינפוט גם את great .

לכן בעוד שמודל לוג־לינארי בהכרח יטעה עבור אחד המשפטים הללו, מודל LSTM יכול להצליח בשניהם.

<u>עבור מילים נדירות,</u> מודל one-hot-vector Log-Linear הוא עם התוצאה הכי פחות טובה <u>באופן משמעותי</u>, עם $\text{accuracy} = 0.28$,
בעוד שמודל LSTM סיפק את התוצאה הטובה ביותר עם $\text{accuracy} = 0.84$.

<u>הסבר אפשרי עבור קבוצה זו:</u> ראשית מודל one-hot-vector Log-Linear עשוי להיכשל כישלון כה חרוץ, כיוון שהייצוג הוא כה ספארסי, ואז נקבל כי המודל שהוגדר ללא רגולריזציה, עלול לספק ערכי w יחסית גבוהים עבור פיצ'רים שמתאימים לחלק מהמילים הנדירות, וערכי w נמוכים למילים נדירות אחרות.

ערכים אלו לא השפיעו משמעותית על אימון המודל ועל התוצאות על הטסט הרגיל,

כי המילים הן נדירות, וגם אם הן הופיעו בחלק מהמשפטים אז חלק גדול מהמילים במשפט לרוב היו מילים שאינן נדירות,

ואז הן הטו את הכף לכיוון הנכון.

אך כאשר הופיעו רק מילים נדירות, כמו בדוגמאות הללו, אז המילים שיצאו עם ערכי w גבוהים יחסית הן אלו שהטו את הכף.

הכללה זו אכן מאוד לוקה בחסר, כיוון שהיא מבוססת על מעט מאוד דוגמאות שבהן המילים הופיעו,

וייתכן שהן הופיעו בהקשרים שונים מאוד.

כבר במודל הלוג־לינארי עם הפחתת מימד ניכר השיפור על המילים הנדירות, עם $\text{accuracy} = 0.7$.

הסבר אפשרי לכך, הוא שכיוון שהמילים הללו שמופיעות מעט מאוד ובהקשרים שונים, הן מתפקדות

כמעין רעש, ובהתאם הפחתת המימד מצמצמת את השפעת המילים הרועשות מביניהן,

וכך עשויה לתפוס גם את הקשרים ביניהן שיותר מעידים על סנטימנט המשפט.

מודל LSTM נחל הצלחה מרשימה על מילים נדירות עם $\text{accuracy} = 0.84$.

ראשית נקודת ההתחלה של המודל היא טובה יותר, כי הוא מקבל את הוקטורים במימד מופחת,

וראינו הרגע שהפחתת מימד משפרת כנראה באופן משמעותי את ייצוג הקשר בין הופעות של מילים באותו הקשר לבין הסנטימנט של המשפט, כי במודלים הלוג־לינארים זה גרם להבדל גדול מאוד.

יתר־על־כן, כאמור, האקספרסיביות של LSTM , שנובעת גם מכך שהמבנה שלו מתייחס גם לסדר הופעת המילים,

מאפשרת לשפר את ההבנה של המודל גם עבור מילים שמופיעות מעט, כי הוא מבין טוב יותר את ההקשר שהן מופיעות בו.

למשל ייתכן שמילה נדירה שבפני עצמה יש לה סנטימנט חיובי, אם היא מופיעה ראשונה במשפט,

היא מעידה מאוד על סנטימנט המשפט לכיוון חיובי, אך אם היא מופיעה באמצע המשפט, אותה מילה משפיעה פחות.

אם אכן קיימים קשרים בסגנון זה, אז LSTM הוא היחיד מבין המודלים שיש לו את הפוטנציאל לזהות זאת.

לסיום אנקדוטה - נצרף צילום מסך של תשעת ה- epoch הראשונים של אחד המודלים הלוג-לינארים, שבהם בוצעו האימון + הפרדיקציה על הולידציה. השתמשנו בחבילה בשם tqdm שמאוד עזרה לנו לעקוב אחרי ההתקדמות של המודלים, עם הצגת הנתונים ו-progress bars . בפרט עבור ה-LSTM ראינו מהר מאוד בהתחלה שהאימון עומד לקחת שעתיים (כי ראינו מיד כמה זמן בערך יקח ה-batch הראשון), וזה עזר לנו לבדוק את עצמנו מיד, ולמצוא את הטעות שגרמה לכך.

```
 ======================================== epoch [1] ========================================
Epoch [1]: 100%|██████████| 1150/1150 [00:01<00:00, 796.50batch/s, accuracy=82.6, loss=0.436]
Val-eval-Epoch [1] : 100%|██████████| 16/16 [00:00<00:00, 664.84batch/s, accuracy=81, loss=0.524]

 ======================================== epoch [2] ========================================
Epoch [2]: 100%|██████████| 1150/1150 [00:01<00:00, 810.37batch/s, accuracy=83.4, loss=0.417]
Val-eval-Epoch [2] : 100%|██████████| 16/16 [00:00<00:00, 675.17batch/s, accuracy=75.9, loss=0.527]

 ======================================== epoch [3] ========================================
Epoch [3]: 100%|██████████| 1150/1150 [00:01<00:00, 728.75batch/s, accuracy=83.4, loss=0.418]
Val-eval-Epoch [3] : 100%|██████████| 16/16 [00:00<00:00, 502.53batch/s, accuracy=75.9, loss=0.531]

 ======================================== epoch [4] ========================================
Epoch [4]: 100%|██████████| 1150/1150 [00:01<00:00, 759.06batch/s, accuracy=83.4, loss=0.417]
Val-eval-Epoch [4] : 100%|██████████| 16/16 [00:00<00:00, 560.79batch/s, accuracy=80.9, loss=0.523]

 ======================================== epoch [5] ========================================
Epoch [5]: 100%|██████████| 1150/1150 [00:01<00:00, 722.47batch/s, accuracy=83.4, loss=0.417]
Val-eval-Epoch [5] : 100%|██████████| 16/16 [00:00<00:00, 504.98batch/s, accuracy=76.9, loss=0.529]

 ======================================== epoch [6] ========================================
Epoch [6]: 100%|██████████| 1150/1150 [00:02<00:00, 557.74batch/s, accuracy=83.4, loss=0.417]
Val-eval-Epoch [6] : 100%|██████████| 16/16 [00:00<00:00, 406.98batch/s, accuracy=80.5, loss=0.521]

 ======================================== epoch [7] ========================================
Epoch [7]: 100%|██████████| 1150/1150 [00:01<00:00, 709.11batch/s, accuracy=83.4, loss=0.417]
Val-eval-Epoch [7] : 100%|██████████| 16/16 [00:00<00:00, 543.55batch/s, accuracy=77.1, loss=0.528]

 ======================================== epoch [8] ========================================
Epoch [8]: 100%|██████████| 1150/1150 [00:01<00:00, 775.00batch/s, accuracy=83.4, loss=0.417]
Val-eval-Epoch [8] : 100%|██████████| 16/16 [00:00<00:00, 655.47batch/s, accuracy=74.2, loss=0.535]

 ======================================== epoch [9] ========================================
Epoch [9]: 100%|██████████| 1150/1150 [00:01<00:00, 842.93batch/s, accuracy=83.4, loss=0.417]
Val-eval-Epoch [9] : 100%|██████████| 16/16 [00:00<00:00, 746.42batch/s, accuracy=77.6, loss=0.527]
```