דו"ח עבודה 3- סדנה בלמידה עמוקה

בעבודה זו קיבלנו את מאגר הנתונים Home depot product search relevance אשר מכיל נתונים על משפטי חיפוש ועל מוצרים אשר חזרו בעבור אותו חיפוש. עבור כל מוצר קיבלנו טקסט המתאר אותו.
מטרתנו במשימה זו הינה לחזות את מידת הרלוונטיות בין שורת חיפוש ובין המוצר שהתקבל בעבורו.
הקושי שנתקלנו בו במשימה זו הינו העובדה ששורות החיפוש נכתבות ע"י משתמשים כך שנכתבות בשפה פשוטה ולא מקצועית בהקשר של המוצר ולעומת זאת תיאור המוצר כתוב ע"י המתמחים כך שיתכן מצב שהמילים במשפטי החיפוש לא יתאמו למילים בתיאור המוצר.
לפתרון המשימה התבקשנו ליצור מודלים המתבססים על רמת התו ועל רמת המילה של הנתונים וניסינו לנתח מה מבין שני המימושים יניב תוצאות טובות יותר ומדוע.

בחרנו להשתמש ב- siamese network, ההתאמה של רשת זו למשימה שלנו היא בזכות זה שמקבלת 2 קלטים שונים ולומדת את הקשר ביניהם. נרצה ללמוד את המיפוי בין מושגים שנכתבו ע"י משתמשים אל מול מושגים שנכתבו ע"י המומחים ובזאת להשתמש ללמידת הרלוונטיות בין שורת החיפוש ותיאור המוצר.

חלק א' –Character level LSTM

סעיף a- בסעיף זה התבקשנו לבצע עיבוד מקדים על נתוני האימון ונתוני המבחן לרצפים של תווים. כדי לבצע את הניתוח השתמשנו בעמודות המתארות את מילת החיפוש ואת תיאור המוצר שהתקבל. איחדנו את הטבלאות על מנת לקבל סט אימון וסט מבחן הכולל את העמודות הנ"ל ובנוסף את עמודת הרלוונטיות של התיאור למילת החיפוש אותה בעצם אנו רוצים לחזות.

- שלב 1- איחוד נתוני האימון והמבחן לטבלאות:

תחילה ביצענו JOIN בין טבלת ה- train המקורית לבין טבלת Product descriptions על בסיס מזהה של המוצר. בנוסף על מנת ליצור את טבלת המבחן לקחנו את טבלת התוצאות והסרנו ממנה את הסיווגים בעלי ערך 1- כיוון שסיווגים אלו מרעישים את הנותנים ולא תורמים ללימוד של המודל. לאחר מכן ביצענו JOIN בין טבלת ה-test לטבלת התוצאות וקיבלנו את סט המבחן שלנו.

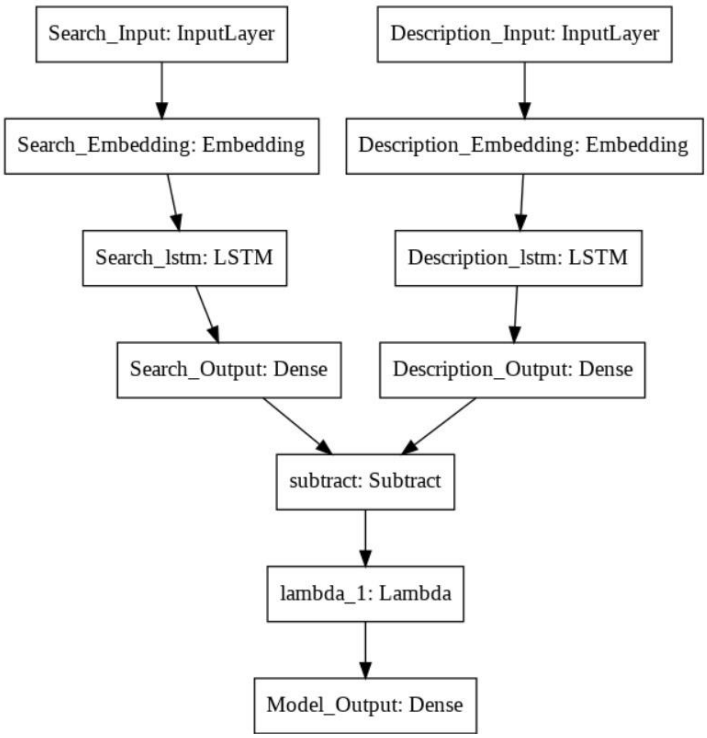
- שלב 2- עיבוד הטקסט לתווים

עבור טבלת האימון וטבלת המבחן הפכנו את העמודות של מילת החיפוש ושל תיאור המוצר לאותיות קטנות כדי להקטין את המילון שייווצר לנו בהמשך. לאחר מכן יצרנו סט של כל ה-Tokens שנמצאים בעמודות אלו שבעזרתם יצרנו שני מילונים: מילון ראשון- מתו לאינדקס ומילון שני הופכי מאינדקס לתו. בהמשך, עבור עמודות אלו עשינו טרנספורמציה של כל אות למספר בעזרת שימוש ב- enumerate כאשר הגבלנו את כמות האותיות ל-1000 מתוך הנחה ש-1000 האותיות הראשונות יתארו לנו באופן יותר מדויק את המוצר שאנו מחפשים.

סעיף b- בסעיף זה בנינו Siamese network אשר מקבלת כקלט שני קלטים שונים אחד עבור מילת החיפוש והשני עבור תיאור המוצר ונותנת כפלט את מידת הרלוונטיות של תיאור המוצר למילת החיפוש. כדי להכניס את הקלטים אל הרשת היינו צריכים להחליט מראש על אורך הקלט שיהיה אחיד לשני הקלטים, כמו שצויין בסעיף a החלטנו על אורך קלט בגודל 1000, כאשר האורך המקסימלי של מילת חיפוש היא 60 תווים בסט האימון והאורך המקסימלי של תיאור מוצר הוא 5516.

לאחר שהחלטנו שאורך הקלט יהיה 1000 עבור כל עמודה היינו צריכים לבצע Padding לתיאורי מוצר אשר היו באורך פחות מ-1000, עבור תיאורי מוצר הגדולים מ-1000 חתכנו את התיאור ולקחנו את 1000 התווים הראשונים. עבור מילות החיפוש ביצענו Padding על מנת להשלים את אורך הקלט ל-1000 תווים.

המודל:



כפי שניתן לראות המודל שני קלטים אחד עבור מילת החיפוש ואחד עבור תיאור המוצר.

שני הקלטים עוברים דרך שכבות הבאות בהתאם לארכיטקטורה של Siamese network:

- Embedding
- LSTM
- Dense.

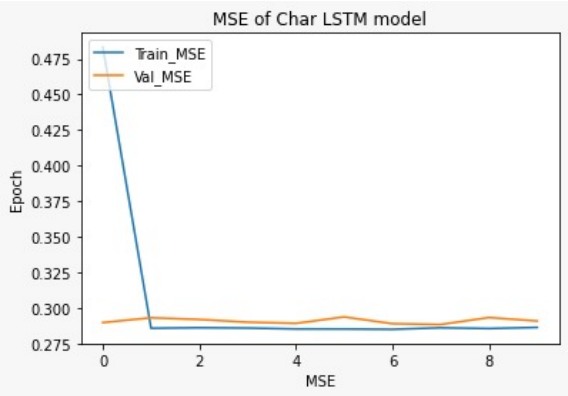
עבור הפלטים שיוצאים משכבת ה-Dense של כל סוג קלט אנו מבצעים חיסור בשכבת ה-Subtract בין הפלטים ולאחר מכן מבצעים ערך מוחלט לתוצאה המתקבלת. אנו עושים זאת בכדי לחשב את המרחק האוקלידי בין שני הקלטים ולקבל פלט אחד שיעבור Dense נוספת עם נוירון אחד והפלט הסופי בעצם יציין את מידת הרלוונטיות של תיאור המוצר ביחס למילת החיפוש.

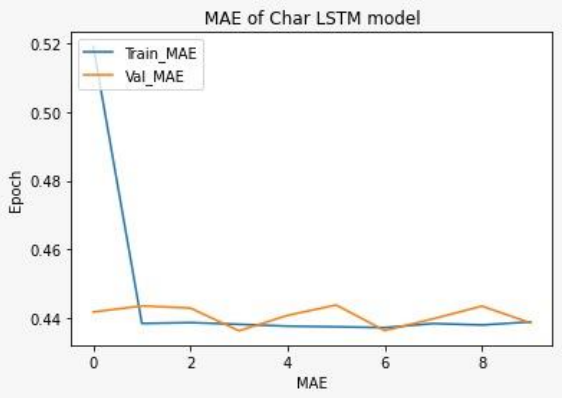
Layer (type)	Output Shape	Param #	Connected to
Search_Input (InputLayer)	[(None, 1000)]	0	
Description_Input (InputLayer)	[(None, 1000)]	0	
Search_Embedding (Embedding)	(None, 1000, 64)	4416	Search_Input[0][0]
Description_Embedding (Embeddin	(None, 1000, 64)	4416	Description_Input[0][0]
Search_lstm (LSTM)	(None, 128)	98816	Search_Embedding[0][0]
Description_lstm (LSTM)	(None, 128)	98816	Description_Embedding[0][0]
Search_Output (Dense)	(None, 64)	8256	Search_lstm[0][0]
Description_Output (Dense)	(None, 64)	8256	Description_lstm[0][0]
subtract (Subtract)	(None, 64)	0	Search_Output[0][0] Description_Output[0][0]
lambda_1 (Lambda)	(None, 64)	0	subtract[0][0]
Model_Output (Dense)	(None, 1)	65	lambda_1[0][0]

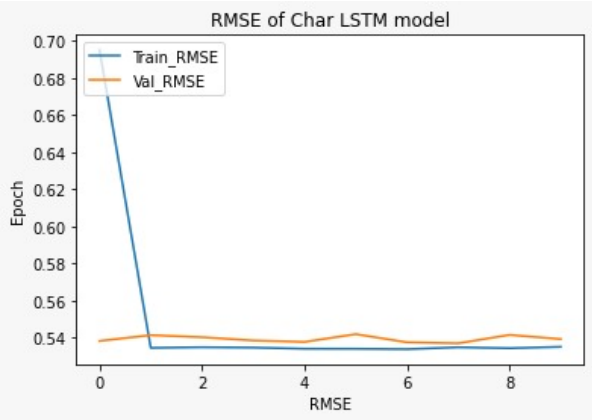
Total params: 223,041
Trainable params: 223,041
Non-trainable params: 0

לאחר שבנינו את המודל חילקנו את נתוני ה-train לסט אימון וסט ולידציה שאותם חילקנו לקלטים לפי מילת חיפש ותיאור מוצר כדי להכניס בהתאמה לרשת, בנוסף יצרנו בהתאמה את מידת הרלוונטיות של האימון והולידציה. כמו כן גם סט המבחן חולק לשני קלטים לפי מילת חיפוש ותיאור מוצר ויצרנו גם את הרלוונטיות עבור סט המבחן.

לרשת הכנסו את שני הקלטים של האימון (מילת חיפוש ותיאור מוצר) ואימנו את הרשת לאורך 10 epochs בשימוש מטריקות של MSE,MAE ו-RMSE במטרה לחזות את מידת הרלוונטיות על סט הולידציה והמבחן.





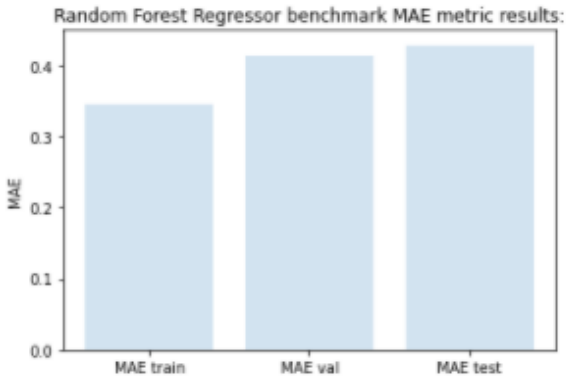


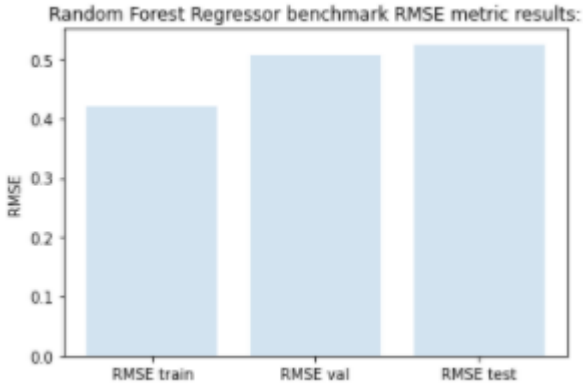
סעיף c- בסעיף זה התבקשנו ליצור מודל נאיבי בסיסי עבור רמת התו. בחרנו להשתמש במודל Random Forest Regressor על מנת לחזות את מידת הרלוונטיות של תיאור מוצר למילת חיפוש.

השתמשנו ב-Count Vectorizer כדי לייצר סט אימון וסט מבחן כשבהמשך פיצלנו את נתוני האימון לסט אימון וסט ולידציה ביחס של 80/20 אחוז.

בשימוש ב- count vectorizer הגדרנו את המילון שלנו ככלל ה- tokens ב-dataset. ביצענו transform עבור מילת החיפוש ותיאור המוצר בסט האימון ובסט המבחן ואז ביצענו איחוד לכדי רשומה אחת וקלט זה ישמש כפ'יצר למודל. ייצאנו את ערכי הרלוונטיות מהאימון והמבחן על מנת שישמשו לצורך הערכה לתוצאות המודל. ביצענו split ליצירת סט אימון, ולידציה ומבחן, הפעלנו פונקציית fit של המודל והשתמשנו במטריקות MAE,MSE ו- RMSE להערכת תוצאות הפרדיקציה של המודל.

```
Random Forest Train RMSE: 0.41999617963133085
Random Forest Val RMSE: 0.5059801084625429
Random Forest Test RMSE: 0.5254932127878273
----------------------------------------
Random Forest Train MAE: 0.3448236694610177
Random Forest Val MAE: 0.41412934697649056
Random Forest Test MAE: 0.42952030993266765
```





סעיף d- בסעיף זה התבקשנו לבצע feature extraction לאחת מהשכבות במודל של ה- LSTM Character שמימשנו ולהשתמש בפיצ'רים אלו בשני מודלים שונים.

בחרנו לבצע פעולה זו עבור שכבת ה-subtract ולהריץ את הפלט של שכבה זו במודלים: Random Forest Regressor ו-XGBoost .

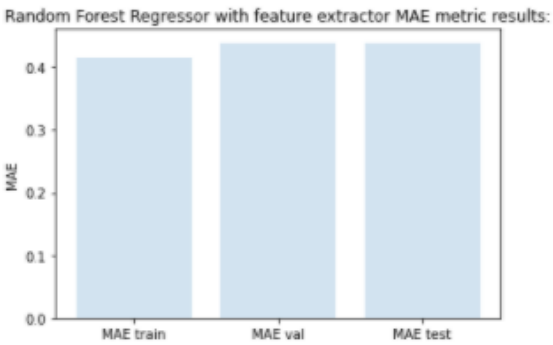
Random Forest Regressor

הגדרנו את ה-hyper parameters של ה-Random Forest Regressor באופן הבא:

```
rf_char = RandomForestRegressor(n_jobs=-1 ,n_estimators=25, max_depth=15)
```

ביצענו predict על סט האימון, הולידציה והמבחן ולאחר מכן ביצענו הערכה לפי המטריקות- MSE, MAE ו-RMSE.

```
Random Forest Train RMSE: 0.5116694189738207
Random Forest Val RMSE: 0.537192571490344
Random Forest Test RMSE: 0.5372712645464337
----------------------------------------
Random Forest Train MAE: 0.41764929226729136
Random Forest Val MAE: 0.4383367767135365
Random Forest Test MAE: 0.4392655200319233
```



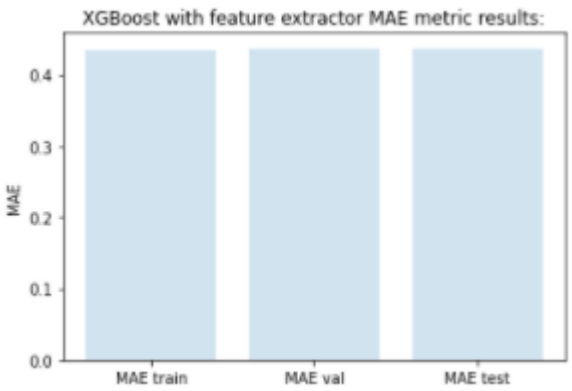
XGBoost

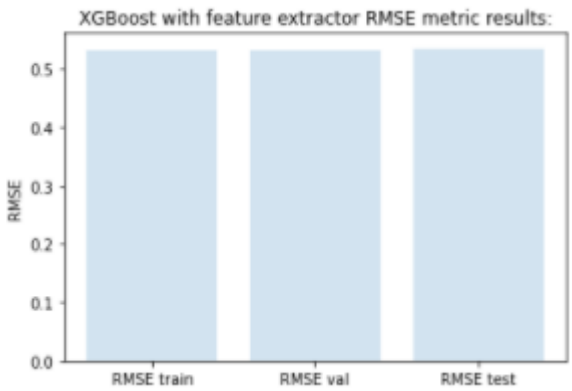
הגדרנו את ה-hyper parameters של ה-XGBoost באופן הבא:

```
xgb_char = XGBRegressor(n_jobs=-1)
```

ביצענו predict על סט האימון, הולידציה והמבחן ולאחר מכן ביצענו הערכה לפי המטריקות- MSE, MAE ו-RMSE.

```
XGBoost Train RMSE: 0.5322272890214622
XGBoost Val RMSE: 0.5323053043191498
XGBoost Test RMSE: 0.5349352005927916
----------------------------------------
XGBoost Train MAE: 0.4356048877640836
XGBoost Val MAE: 0.4367287282784715
XGBoost Test MAE: 0.43787083629924994
```





חלק ב

- **עיבוד מקדים של הנתונים:**

ביצענו עיבוד מקדים לטקסט שכלל :

1. הורדת סימני פיסוק
2. הורדת stop words
3. הפיכה לאותיות קטנות
4. Lemmatization- שבניגוד ל-stemming מחייב ששורש המילה קיים בשפה, זמן ההכנה של הנתונים הוא יחסית ארוך אבל מכיוון שאנחנו עושים זאת פעם אחת וכמות הנתונים לא 'ענקית' אז העדפנו את זה ע"פ stemming.
5. ניקוי מספרים ממילים – הרבה פעמים יש מספר שמחובר לטקסט כמו מידות או סוג מוצר ספציפי אז בחרנו ליצור תבנית אחידה למספרים כאשר פשוט פיצלנו את המילים שמכילות מספר לתווים ומספרים לפי הרצף המקורי.

דוגמה:

טקסט מקורי –

Toro Personal Pace Recycler 22 in. Variable Speed Self-Propelled Gas Lawn Mower with Briggs & Stratton Engine

טקסט לאחר עיבוד –

toro personal pace recycler 22 variable speed self propelled gas lawn mower briggs amp stratton engine

- **Word2vec**

בחרנו להשתמש במודל של genism work2vec, הפרמטרים של המודל עבורם ביצענו את החיפוש על מנת להביא את רשת הנוירונים לאופטימיזציה הם גודל שכבת ה-embedding, מספר ה-epochs וגודל החלון של ה-word2vec.

ביצענו את אימון המודל כשלב מקדים לפני המודל המורכב ואז השתמשנו במודל המורכב בשכבות embedding שטענו אותם עם המשקולות של ה-word2vec ועשינו כי שכבות אלה לא יתאמנו כלל מכיוון שכבר אימנו אותם מראש. בנוסף השתמשנו ב-Mask zero מכיוון שהמשפטים הם באורכים משתנים ונרצה שהמודל יוכל להתעלם מ-padding.

- **Siamese model**

בחלק זה בחרנו להשתמש ב-3 קלטים עבור המודל

1. כותרת
2. תיאור הפריט
3. מילות חיפוש

פרט לשינוי המתואר ב-word2vec המודל נשאר דומה למודל של התווים.

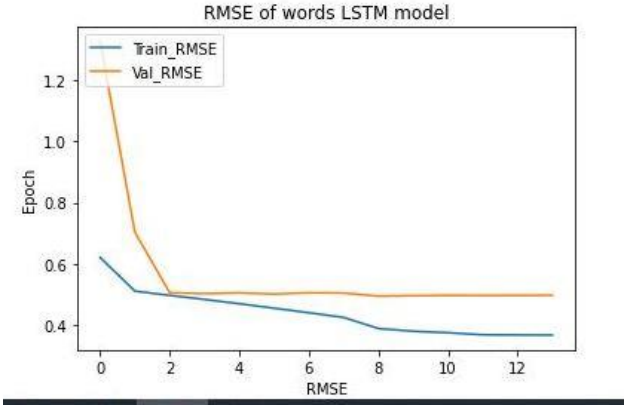
שי ארץ קדושה – 203276258 גל בוזגלו – 203886528 גל בורביע - 203535802

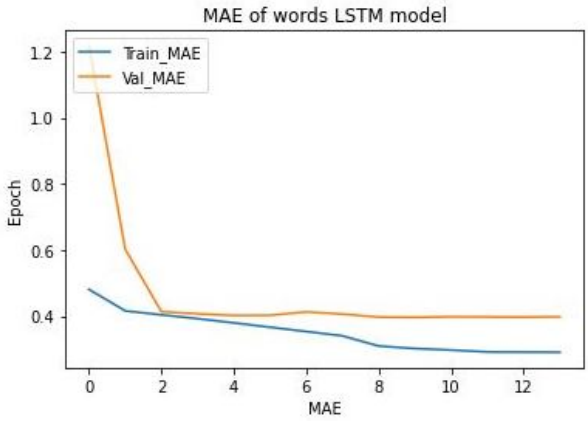
ארכיטקטורה –

פרמטרים טובים ביותר –

```
# params
batch_size=128
epochs = 20
search_size = 8
title_size = 20
desc_size =96
emb_size =32
```

- **תוצאות המודל**
  - **מודל 1-**






```
Siamise model train score MSE - 0.138372850992898
Siamise model train score RMSE - 0.3719850144735645
Siamise model train score MAE - 0.29645042043655084
Siamise model validation score MSE - 0.24778723196318125
Siamise model validation score RMSE - 0.4977823138312381
Siamise model validation score MAE - 0.39799782848117815
Siamise model test score MSE - 0.2943617024144202
Siamise model test score RMSE - 0.5425511058088631
Siamise model test score MAE - 0.43533954302190486
```

ניתן לראות שע"פ התוצאות שקיבלנו כי המודל שלנו נוטה טיפה להתאמת יתר אולם הבעיה היותר מרכזית בו היא שניתן לראות כי מהאפוק השני המודל שלנו לא מצליח לשפר משמעותית את ה-validation data.
על מנת להבין יותר טוב את הבעיה של המודל שלנו אנחנו נבחן את התוצאות של המודל על הנתונים השונים.

statistic	Train labels	Train pred	Val labels	Val pred	Test labels	Test pred
Mean	2.3827	2.3715	2.379	2.3735	2.3805	2.3818
Median	2.33	2.408	2.33	2.4	2.33	2.4098
Std	0.534	0.3439	0.534	0.3195	0.5353	0.3138

מכיוון שאנחנו רואים כי ההבדל המשמעותי בין תוצאות המודל לבין התוצאות האמיתיות הוא בסטיית התקן של התוצאות אנחנו נבדוק כמה טוב המודל שלנו מזהה תצפיות שהערך שלהן גדול מ-2.8 או שהערך שלהן קטן מ-1.4. כלומר נראה כמה המודל שלנו מצליח לחזות ערכים נמוכים (אין קשר בין החיפוש למוצר) או ערכים גבוהים (יש קשר חזק בין מילות החיפוש למוצר)

	Train labels	Train pred	Val labels	Val pred	Test labels	Test pred
>2.8	13452	4604	5673	1613	28604	8267
<1.4	3571	408	1544	95	7973	431
סה"כ תצפיות	51846	51846	22221	22221	112067	112067

ניתן לראות כי ישנם הבדלים משמעותיים בין בתוצאות החיזוי לתוצאות האמת עבור ערכים שנמצאים בקצוות בכל אחד מסוגי הנתונים שבחנו.
מניתוח תוצאות אלה אנחנו מבינים כי המודל שלנו נמצא ב-underfitting הוא איננו מצליח לזהות באופן טוב תצפיות בקצוות והוא למעשה "לומד" את הממוצע של החיזוי ורוב החיזויים נמצאים בטווח של הממוצע ללא קשר לערך האמיתי של התצפית. בנוסף ניתן לראות כי יש מגמה של overfitting ע"פ התוצאות של סוגי הנתונים השונים אולם זו איננה הבעיה העיקרית של המודל.
על מנת לטפל בבעיות שהצגנו בחרנו לבצע מספר שינויים:

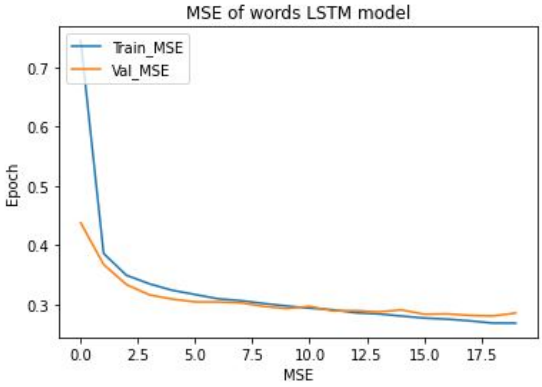
1. הוספת שכבות dropout למודל על מנת להוריד את ההתאמת היתר של המודל, הסיבה שבחרנו ב-dropout היא מכיוון שכבר במודל הראשוני השתמשנו ב-batch normalization ובנוסף לא רצינו להשתמש ברגולריצזיה מסוג l2 מכיוון שאנחנו חשבנו שאם המשקולות של המודל יהיו יותר 'אחידות' אז זה יכול להחמיר את הבעיה של underfitting ויקשה יותר לדייק בתוצאות שנמצאות בקצוות.
2. עבור הבעיה העיקרית שלנו שהיא שהמודל לומד את הממוצע ומתקשה לדייק בתוצאות שנמצאות בקצוות, ניסינו מספר פתרונות כדי שהמודל ילמד יותר טוב.

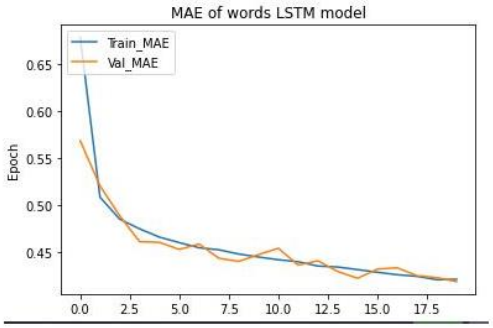
א. הגדלת נתוני האימון – בחרנו להגדיל את נתוני האימון בשתי צורות שונות, בפעם הראשונה הכפלנו את כל התצפיות שנמצאות בקצוות (מעל 2.8 או מתחת 1.4) חשבנו שבצורה כזו נצליח לשפר את הדיוק בתצפיות הקשות וכך לשפר את הדיוק הכללי של המודל. האופן השני היה להכפיל את נתוני האימון כך שבמקום לקחת רצף של מילים נקח set כדי להוסיף יותר חשיבות למשמעות של המילים מאשר לסדר שלהם במשפט.

ב. הורדת גודל ה-batch על מנת שהמודל לא יתכנס לממוצע.

הפעולות שביצענו עזרו לשפר את תוצאות המודל בצורה קטנה, מה שעזר בנוסף היה צמצום גודל ה-embedding.

תוצאות המודל






```
Siamise model train score MSE - 0.20624872520866142
Siamise model train score RMSE - 0.4541461496133832
Siamise model train score MAE - 0.363908887417947
Siamise model validation score MSE - 0.28078250750080436
Siamise model validation score RMSE - 0.5298891464266884
Siamise model validation score MAE - 0.4223782611281916
Siamise model test score MSE - 0.27225462200303363
Siamise model test score RMSE - 0.5217802430171476
Siamise model test score MAE - 0.41521773760304725
```

```
Runtime for xgboost  is 48.78004789352417 sec
Xgboost train score MSE - 0.211125498976538
Xgboost train score RMSE - 0.4594839485515658
Xgboost train score MAE - 0.37386603059331475
Xgboost validation score MSE - 0.28049204288595425
Xgboost validation score RMSE - 0.5296149949595028
Xgboost validation score MAE - 0.4279336317557449
Xgboost test score MSE - 0.2654496764395739
Xgboost test score RMSE - 0.5152180862892664
Xgboost test score MAE - 0.4121852942967907
```

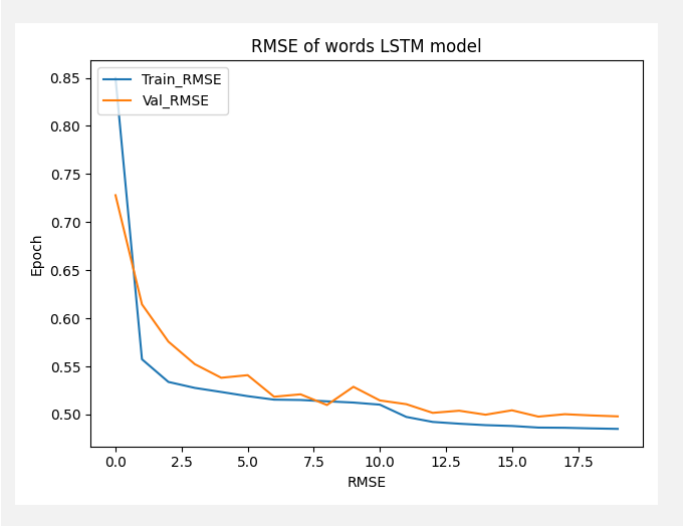
ניתן לראות כי הצלחנו להקטין את ההתאמת היתר, השגיאה של נתוני האימון והווליד‌ציה גדולים מהמודל הקודם (מכיוון שהוספנו נתונים בקצוות שהשגיאה בהם היא יותר גבוה) אבל בנתוני המבחן הצלחנו לרדת כאשר את התוצאה הטובה ביותר נתן xgboost.

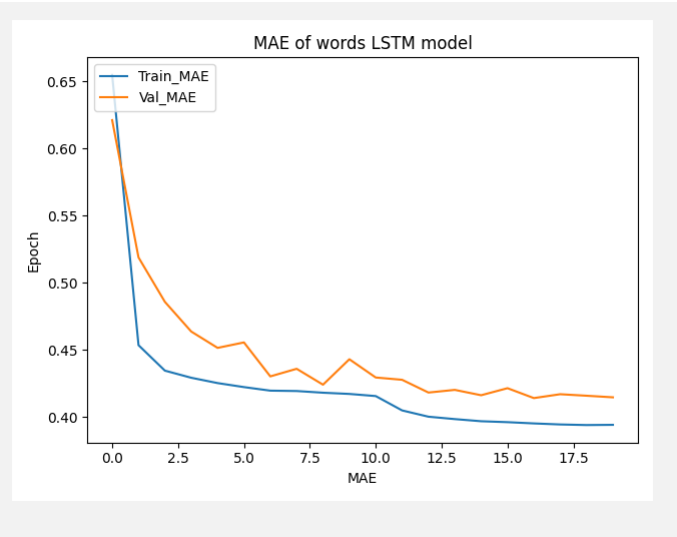
- פעולות נוספות שביצענו על מנת לנסות לשפר אתת המודל –

1. הרחבה של משפטים – עבור משפטים קצרים ביצענו הרחבה למילים על ידי המילים הקרובות ביותר אליהן ב-word2vec שעוברות סף מסוים של דמיון.
הסף שבחרנו הוא 0.6.
2. מכיוון שקראנו כי מודל word2vec מצריך המון נתונים להתאמן עליהם ניסינו להשתמש במודל שאומן מראש (בחרנו מודל של ויקיפדיה מכיוון שהמודל של גוגל לא נטען בזיכרון ומודלים על טוויטר נראו לנו פחות מתאימים).
התוצאה של המודל הזה הייתה משמעותית פחות טובה מכל המודלים.
3. שינוי מספר המילים שנלקחות בחשבון בחיפוש – מכיוון שהממוצע של מילות החיפוש הוא יחסית נמוך, 3, אז ניסינו למצוא את האיזון בין להשתמש בפחות מדי מילים ואז לאבד מידע בחיפושים ארוכים לבין לבצע יותר מדי padding כאשר בוחרים מספר גבוה יותר.
4. שינויים בפרמטרים של המודל כמו מספר הקודקודים בשכבת dense או ב-lstm, גודל שכבת ה-embedding ועוד.

לאחר שהפעולות הללו לא הצליחו להביא לשיפור משמעותי בתוצאות המודל הבנו שיש צורך לבצע שינוי בארכיטקטורה של המודל או לבצע שינוי בקלט של המודל. מכיוון שראינו כי שינויים במודל מביאים במקרה הטוב לשיפורים קלים בתוצאות החלטנו לבדוק את ההשפעה של מודל ה-word2vec על התוצאות של המודל הכללי, לשם כך אימנו מודל word2vec עם גודל של 32 מספר רב של איטרציות כ-1000 במספר.

- **תוצאות מודל 3**





את התוצאה הטובה ביותר מבין כל המודלים השונים במקרה הזה נתן ה-Catboost model.

```
Runtime for catboost is 471.56444478034973 sec
Catboost train score MSE - 0.16324352032572717
Catboost train score RMSE - 0.40403405837345835
Catboost train score MAE - 0.32540209543968235
Catboost validation score MSE - 0.2264210074758876
Catboost validation score RMSE - 0.4758371648745898
Catboost validation score MAE - 0.38378533939427895
Catboost test score MSE - 0.25792554397895573
Catboost test score RMSE - 0.507863706105246
Catboost test score MAE - 0.41067603686654297
```

- טבלת השוואות בין המודלים השונים

Model type	runtime	Train rmse	val-RMSE	Test-RMSE	Train MAE	Val MAE	Test MAE
Naïve benchmark	20 sec	0.4188	0.505	0.5252	0.34440	0.4126	0.4293
Character level LSTM	588 sec	0.5342	0.5337	0.5359	0.4356	0.4364	0.4293
Feature extractor - Random forest	13.8 sec	0.5114	0.5335	0.5371	0.4166	0.4373	0.439
Feature extractor - xgboost	8 sec	0.5322	0.5323	0.5349	0.4166	0.4367	0.4378
Word level input	244 sec	0.4780	0.4980	0.5150	0.3972	0.4146	0.4275
Feature extractor - Random forest	184 sec	0.4636	0.4955	0.5102	0.3787	0.4065	0.4178
Feature extractor – Catboost	471 sec	**0.4040**	**0.4758**	**0.5079**	**0.3254**	**0.3838**	**0.4107**

פעולות שניתן לבצע על מנת לשפר את המודל –

- פעולה שאנחנו חושבים שיכולה לעזור היא טיפול טוב יותר מתאים לדאטה, בקובץ ההוראות שמצורף לנתונים יש הסבר על אופן בו נתנו לכל תצפית את ערך הרלוונטיות. אנחנו חושבים שאפשר לנסות לגזור מהחוקים האלה מאפיינים נוספים לבעיה שיכולים להוביל לשונות גבוה יותר בין התצפיות וכך לשפר את התוצאות של התצפיות שנמצאות בקצוות ומובילות לשגיאה גבוה. לדוגמה אם נעביר כקלט נוסף למודל את אחוז המילים ממילות החיפוש שמוכלות במילות התיאור\הכותרת של אותו מוצר אז אנחנו מאמינים שפעולה זו יכולה לשפר את התוצאות ביחד עם חוקים נוספים שניתן לגזור מקובץ ההוראות.
- פעולה נוספת שלדעתנו יכולה לעזור היא להשתמש במודל של עיבוד שפה טבעית על מנת לנסות להבין מה הם החלקים במשפט שמתארים הכי טוב את המוצר, מכיוון שלרוב הטקסט שמאפיין את המוצר הוא ארוך משמעותית ממילות החיפוש אז לדעתנו זה פוגע במודל הסיאמי ולכן קיצור הטקסט רק למילים רלוונטיות יכול לעזור לשפר את המודל הסיאמי.