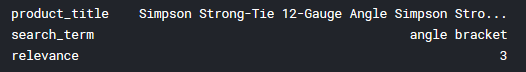
**Character level LSTM**

במטלה זו נתבקשנו למצוא את מידת הרלוונטיות בין שאילתת חיפוש לבין שם מוצר מסוים המופיעים באתר של חברת Home Depot. מידת הרלוונטיות נמדדת בין 1 (לא רוולנטי בכלל, לדוגמא חיפשתי "שלט לחדר" והמוצר הוא "בקבוק מים ליטר וחצי") ל3 (מאוד רלוונטי, לדוגמא חיפשתי "שלט לחדר" וקיבלתי מוצר בשם "שלט לחדר בגודל 20x20).

דוגמא:



**סקירת הנתונים**

איזה מידע עומד לרשותנו בקבצים?

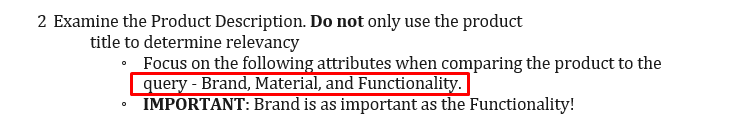
עבור הtrain יש לנו לכל רשומה: כותרת של המוצר, מילת חיפוש, מידת רלוונטיות. (ומס' סידורי של המוצר, שימוש נסביר בהמשך).

עבור הtest יש לנו כמובן את אותה רשומה רק בלי עמודת הרלוונטיות.

יש את הקובץ של מאפייני המוצרים. בקובץ זה מופיע עבור כל מוצר (לפי המספר הסידורי שלו) מספר פרטים טכניים על המוצר.

יש את הקובץ שכולל תיאור קצר של מספר שורות על כל מוצר (גם כאן, לפי מספר סידורי).

יש את הקובץ שבו ניתנו ההוראות למדרגי הרלוונטיות כיצד לפעול, וניתן להפיק מידע חשוב מהקובץ הזה. לדעתי החלק הכי חשוב שאפשר ללמוד ממנו זאת השורה הבאה:



ההוראה הניתנת למדרגים היא לתת דגש על שם המותג והחומר ממנו הוא עשוי וכמו כן על הפונקציונליות שלו. אפשר להניח מזה שצירוף מידע זה עבור כל מוצר כדי לאמן את הרשת וכמובן לבצע פרדיקציה בהמשך יוכל לשפר לנו את התוצאות – אנחנו למעשה שואפים לבצע דירוג בדיוק כמו שהמדרגים המקוריים היו עושים.

אז איך מתבצע הדירוג?

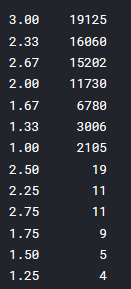
ישנם 3 שופטים שבסופו של דבר כל אחד נותן דירוג של 1/2/3. לשלושת המספרים האלה עושים ממוצע וזה הדירוג הסופי.

כמות הנתונים?

74067 רשומות בקובץ הtrain.

166693 רשומות בקובץ הtest.

איזון הנתונים?



טוב אז כמו שניתן לראות, הנתנים מאוד לא מאוזנים. באופן כללי רוב המוחלט של הדוגמאות הוא בין 2-3 ויש מעט מאוד דוגמאות שהן בין 1-2. הדבר עלול לגרום לרשת שלנו לנבא רק מספרים גבוהים ולא "להתאמץ" לנבא מספרים שהם בין 1-2.

**Pre Processing**

אחלק את החלק הזה ל-2, כיוון שבאיזה שהוא שלב בביצוע המטלה ניסיתי לשנות כיוון ולהתייחס לבעיה דווקא כבעיית Multi class classification (וזאת כיוון שלא מדובר בניבוי של מספר רציף בין 1-3 אלא טווח הערכים הוא קבוע – סה"כ 13 אופציות).

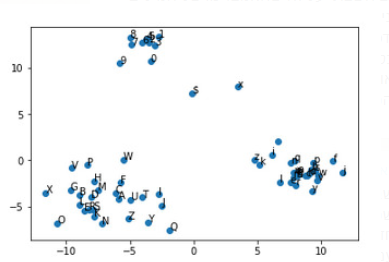
אמנם, תחילה התחייסנו לבעיה זו כבעיית חיזוי ולכן ביצעו את הצעדים הבאים:

* קריאת המידע של קובץ הtrain.
* הפיכת כל כותרת וכל שאילתא לוקטור המופרד לתווים.
* כעת, שיניתי כל תו לייצוג ע"פ OneHot וזאת על מנת שכולם יכנסו מנורמלים לרשת באותה מידה ולא יהיה מצב שנגיד התו A שמיוצג ע"י המספר 30, יקבל חשיבות גדולה יותר מהתו b שמיוצג ע"י המספר 2.
* על הרשת לקבל קלט בגודל קבוע, לכן היה צורך לבצע OneHot לכל ביטוי ולאחר מכן לבצע padding לגודל קבוע. הגודל שנקבע לpadding הוא לפי האורך הממוצע של כותרת ושאילתא. הייתי שמח לבצע עם האורך המקסימלי אך כמות הזיכרון לא אפשרה את זה, וההרצה הייתה קורסת כל הזמן. הסיבה לכך שלא בחרתי בlabelencoder כיוון שאם ניקח לדוגמא את מה שהוא עושה לאות a הוא למשל יהפוך אותה ל0. לאחר מכן את האות z הוא יהפוך ל26. אם הרשת שלנו תקבל את זה כinput היא תחשוב שz "חשוב" יותר או "חזק" יותר כיוון שהמספר שלו גבוה יותר. זה הוא לא מצב שאנחנו רוצים להגיע אליו, כיוון שכל אות צריכה לקבל התייחסות שווה.

המשך כאשר שינינו לבעית קלאסיפיקציה:

* כל רשומה שינינו את הy שלה מערך (מספר בין 1-3) לאחד מבין ה13 class הקיימים.
* אמנם בהמשך זיהינו כי ישנם בערך 50 רשומות שהclass שלהם הוא מספר שהשארית שלו היא כפולה של רבע. שזה מוזר, כי זה היה ממוצע בין 3 שופטים ומספר זה לא הגיוני כל כך (יתכן והשמיטו החלטה של שופט). לכן, החלטנו להוריד את הclass האלה ולהשאר רק עם 7 class השונים: 1, 1.33 ,1.667, 2, 2.33, 2.667, 3.

עוד מידע חשוב! כן ניסיתי לבצע אמבדינג ואכן ניתן לראות כי התוצאות מאוד מעניינות. את האמבדינג לתווים ביצעתי ע"פ Word2Vec. להלן התוצאות:



ניתן לראות כי התוצאות מעניינות מאוד כיוון שהוא אכן הפריד בין מספרים לאותיות קטנות ואותיות גדולות ומוצא שאכן הם קשורים ביניהם. אמנם, הדבר לא היה שימושי ולא הביא לתוצאות טובות יותר. המילון של האמבדינג הכיל רק: אותיות קטנות, אותיות גדולות ומספרים.

**Validate Strategy**

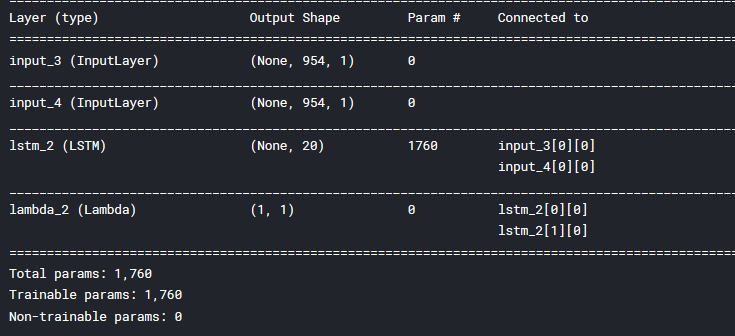
אז כן, מאוד מאוד הייתי רוצה להשתמש ב stratified k foldשמוודא בדיוק כך שבכל fold יהיה מספר שווה של class, אמנם הדבר באמת גמר לי את הזיכרון כל פעם, ולא הייתי מצליח לבצע את זה על יותר מfold 1. ולכן ביצעתי train-test split עם shuffle ו0.85/0.15 חלוקה. חבל, אני בטוח שהדבר היה עוזר לתוצאות.

**Building Our Siamese network**

עברתי בתהליך הזה דרך קשה מאוד.מאוד. (הדפסנו את מה שביקשת רק בשלב האחרון כלומר רק למודל הסופי). בכל השאר ביצענו השוואה רק על התוצאות על הtest.

**שלב א':**

התחלתי עם רשת פשוטה: קלט שמאל וקלט ימין -> שכבת LSTM -> דמיון ע"י מנהטן דיסטנס (עם אקספוננט) ונרמול התוצאות כך שיהיו בין 1-3.

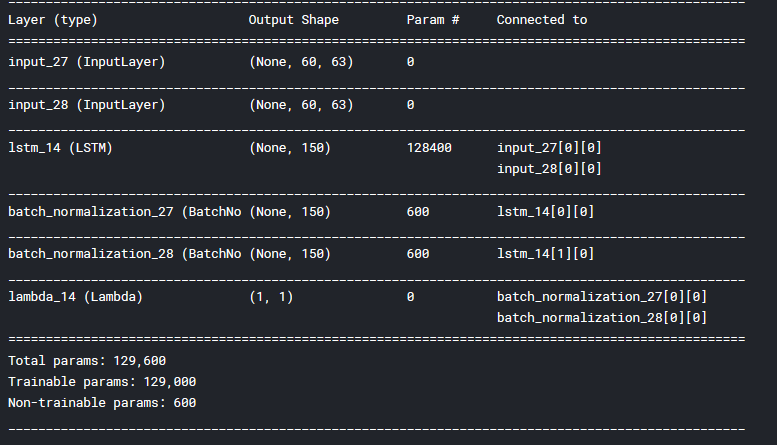


נתחיל לסכם בטבלה כדי לראות את ההתקדמות המודל:

|  |  |  |
| --- | --- | --- |
| Model type | RMSE | MAE |
| LSTM + Manhattan distance (prediction) | 0.532 | 0.423 |

**שלב ב':**

לאחר מכן, ניסיתי לשנות את חישוב הדימיון לCosine Similarity.



כנראה בגלל החלוקה הלא מאוזנת באיזה שהוא שלב הרשת לומדת כי הכי נכון מבחינתה זה תמיד יהיה לנבא 2.33 ובכך למזער את הmse. ואכן היא צדקה, בעיקרון קיבלנו rmse יחסית נמוך (נראה זאת בהשוואה לדברים בהמשך).

|  |  |  |
| --- | --- | --- |
| Model type | RMSE | MAE |
| LSTM + Manhattan distance  (prediction) | 0.532 | 0.423 |
| LSTM + Cossim + batchNorm  (prediction) | 0.532 | 0.423 |

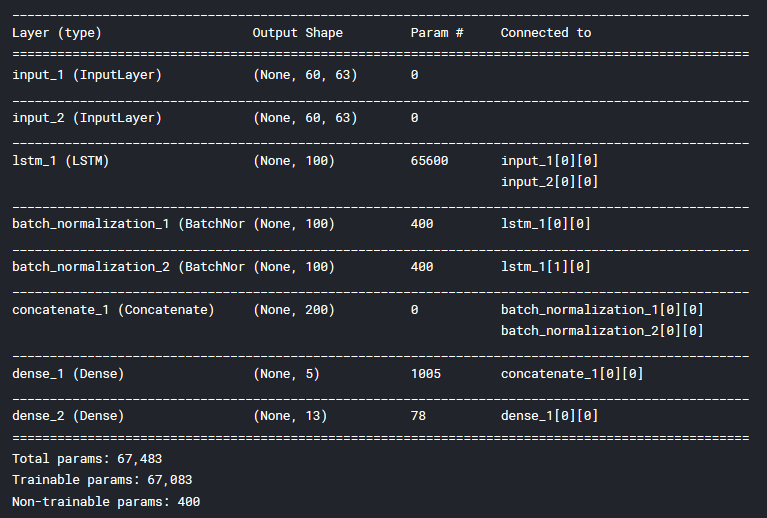
אותם תוצאות.

רגע, למה? אם נסתכל על הפרדקיציה, נראה כי המודל תמיד מנבא 2.33 ולכן התוצאות זהות לגמרי.

**שלב ג':**

כאן החלטתי לנסות כיוון חדש. יתכן ופרדיקציה היא לא השיטה הנכונה, ולכן אנסה לעבור לבעיית Multi class classification. אז איך השינוי מתבטא?

כאשר מקבלים את הפלט של שכבת הLSTM, במידה ונבצע את אותה פעולה כמו מקודם (עם הדמיון) איך ממש משמעות אמיתית לבחור לאיזה שהוא קלאס, כי התוצאה כבר בידנו, לכן שיניתי את הבעיה לרשת בצורה הבאה: לאחר ביצוע הlstm נבצע conccate ל2 החלקים, נחבר לשכבת FC, ואת זה לבסוף נוציא בעזרת softmax עם 13 קטגוריות (האופציות של הניבוי).



|  |  |  |
| --- | --- | --- |
| Model type | RMSE | MAE |
| LSTM + Manhattan distance  (prediction) | 0.532 | 0.423 |
| LSTM + Cossim + batchNorm  (prediction) | 0.532 | 0.423 |
| LSTM + concat + FC  (classification) | 0.263 | 0.14 |

**שלב ד':** הוספת שם המוצר והחומר ממנו עשוי המוצר כפי שתיארתי בשלב הכנת הנתונים שאלו היו הוראות השופטים לכן מניח שהדבר יכול לעזור (הרשת אותה רשת)

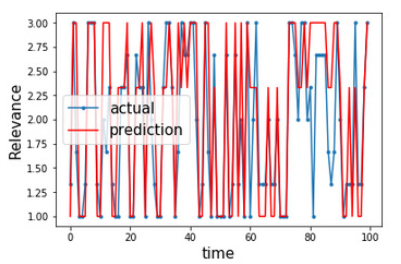
|  |  |  |
| --- | --- | --- |
| Model type | RMSE | MAE |
| LSTM + Manhattan distance  (prediction) | 0.532 | 0.423 |
| LSTM + Cossim + batchNorm  (prediction) | 0.532 | 0.423 |
| LSTM + concat + FC  (classification) | 0.263 | 0.14 |
| LSTM + concat + FC + Brand and Mat  (classification) | 0.228 | 0.056 |

בחקירת הפונקציה, ניתן לראות כי המודל כמעט לא מגיע למספרים הנמוכים. בעיה. ננסה לפתור זאת בשלב ה'

**שלב ה':** ניסיתי לבצע data augmentation כדי לנסות להבטיח ניבויים בכל הclass ולא רק בגבוהים. איך עשיתי את זה? יצרתי באקראית רצפים של אותיות (כל תו אקראי ולא תלוי בקודם) כך עבור כותרת וגם עבור שאילתא ולהם נתתי דירוג אקראי של 1/1.33 כיוון שאני מניח שיצירה של 2 ביטויים אקראים לא יהיו דומים בכלל. בצורה הזאת אני מאזן את הdataset שלי ואביא אותו לנבא גם תוצאות נמוכות. הרשת תהיה בדיוק אותה רשת כמו מקודם.

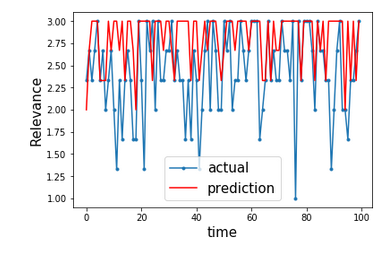
|  |  |  |
| --- | --- | --- |
| Model type | RMSE | MAE |
| LSTM + Manhattan distance  (prediction) | 0.532 | 0.423 |
| LSTM + Cossim + batchNorm  (prediction) | 0.532 | 0.423 |
| LSTM + concat + FC  (classification) | 0.263 | 0.14 |
| LSTM + concat + FC + Brand and Mat  (classification) | 0.228 | 0.056 |
| LSTM + concat + FC + Brand and Mat + Augmentation  (classification) | 0.542 | 0.364 |

אז התוצאה של הדאגטה אוגמנטיישן פחות טובה מהקודם, אך לדעתי עדיף את המצב הזה כי הניבויים שהוא נותן גם מגיעים למספרים הנמוכים, כלומר אנחנו פותרים בעיה יותר כללית, מאשר ספציפית לדאטה סט הזה.



הבעיה במקרה שלי שהורידה את הrmse היא שהוא כבר בעיקר הולך לנבא 3 ולא נעצר כמעט ב2.667 או 2.33 ולכן לדעתי זה הוריד את הrmse, מצד שני הרווחנו את כל הניבויים הנמוכים.

ככה זה היה נראה לפני (התוצאות טובות יותר, אך הרוב מתנזק ל2.33/2.66/3:



**סיכום:**

|  |  |  |
| --- | --- | --- |
| Model type | RMSE | MAE |
| LSTM + Manhattan distance  (prediction) | 0.532 | 0.423 |
| LSTM + Cossim + batchNorm  (prediction) | 0.532 | 0.423 |
| LSTM + concat + FC  (classification) | 0.263 | 0.14 |
| LSTM + concat + FC + Brand and Mat  (classification) | 0.228 | 0.056 |
| LSTM + concat + FC + Brand and Mat + Augmentation  (classification) | 0.542 | 0.364 |

נבחר את ההגרסא הכחולה לפחות עבור הChar Level.

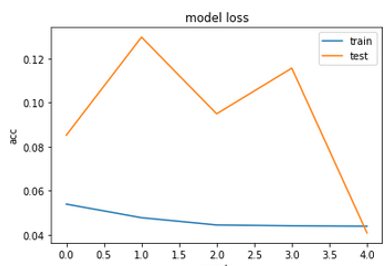
**Naïve model / benchmark**

מהניסיון הרב שצברתי כבר על הdata, לדעתי אם ננבא את כולם 3 או אפילו ננסה גם לנבא באקראית או 3/2.667/2.33 נקבל תוצאות שהם ממש יהיו קרובות למודלים הקודמים, כיוון שזה מה שהמודלים בתכלס עושים.

|  |  |  |
| --- | --- | --- |
| MAE | RMSE | Model type |
| 0.617 | 0.815 | Only 3 |
| 0.458 | 0.6 | Only 2.667 |
| 0.433 | 0.535 | Only 2.33 |
| 0.5 | 0.66 | Random between the 3 |

מסקנה: התוצאות של הרשת שלנו (כמעט לאורך כל הדבר, חוץ מהשינוי לקלאסיקפיקיישן) הביאו לאותם תוצאות כמו המודל הנאיבי (כיוון ששניהם ניבאו כמעט תמיד את אותה תוצאה). לכן, אני חושב שניתן לקבוע כי מדובר בבאנצ' מארק חזק.

ההדפסות הסופיות ושורת הסיכום למודל שלנו:

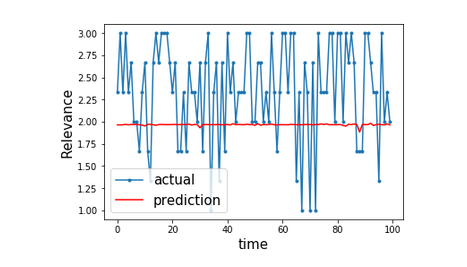


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model type | runtime | Train RMSE | Val-RMSE | Test-RMSE | Train MAE | Val-MAE | Test-MAE |
| LSTM + conc + BrandMaterial + Multi Class | 15 min | 0. 231 | 0.229 | 0.228 | 0.058 | 0.057 | 0.056 |

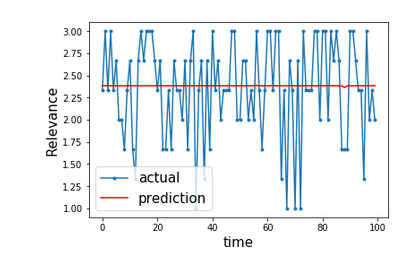
**feature extractor for other ML model**

בחלק זה, ביצענו את הרשת המקורית עד אחרי שלב הקונקט, ובמקום לתת כעת לשכבה של FC, כאשר בסיומה sofmax (כלומר שלב הסיווג), נשתמש כעת בשני מודלי ML שונים. בחרתי להשתמש ב2 מודלים שעדיין לא יצא לי להשתמש בהם בעבר, כדי ללמוד עליהם גם:

Xgboost: זה הוא Ensamble של עצי החלטה. האלגוריתם פועל בצורה איטרטיבית ככה שבכל פעם הוא מנסה לשפר את העץ הקודם שהוא בנה, כלומר אם השגיאה בעץ הקודם הייתה גדולה בגלל פיצ'רים מסוימיים אז הוא מנסה לתת להם משקלים שונים כדי לתקן את זה. הוא פועל בצורה רוחבית. להלן הניבוים שלו:



Lightgbm: דומה ברעיון לXgboost רק שהוא בונה לעומק, ולכן פועל בצורה יותר יעילה. הוא כל פעם הולך לעומק ומחפש את הדרך הכי טובה ללכת בה. נראה שנמצא היום יותר בשימוש. להלן הניבויים:



ולתוצאות:

שניהם בסופו של דבר, ניבאו את אותו ערך כל הזמן, ולכן קשה לקבוע איך טיב התוצאות שלהם. נסכם בטבלה:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model type | runtime | Train RMSE | Val-RMSE | Test-RMSE | Train MAE | Val-MAE | Test-MAE |
| LSTM + conc + BrandMaterial + Multi Class | 15 min | 0. 231 | 0.229 | 0.228 | 0.058 | 0.057 | 0.056 |
| LGB |  |  |  | 0.537 |  |  | 0.44 |
| XG |  |  |  | 0.675 |  |  | 0.573 |

**סיכום חלק א'!**

* הבעיה המרכזית אני חושב הייתה פה, שהיה חסר מידע על זוגות של לא רלוונטים, והדבר גרם לרשת "לא להתאמץ" ולנסות להגיע ל1ים. ניסיתי לפתור את זה בצורה של Data augmentation שבו אנו ממציא אותיות אקראיות ומניח שהקשר בין רצף תווים אקראי אחד לאחר הוא ברלוונטיות 1/1.33. הדבר אכן עזר לי כדי להגיע לניבויים הנמוכים אך מצד שני במבחן התוצאה אל מול הtest הוריד לי הRMSE לכן "נאלצתי לחזור למודל הקודם". **מחשבה לעתיד** להמשך יתכן ואפשר בחלק ב' לבצע תהליך דומה, רק לקחת את הרשומות הלא רלוונטיות ולהחליף מילים במילים הנרדפות שלהן וככה ליצור לנו עוד Data.
* כמו כן, קריאת הDataset וההוראות הוביל להבנה כי הוספת המותג ושם החומר של המוצר יכול להוסיף לנו עוד נקודות למודל, ולכן למדתי כי חשוב לקרוא כל דבר קטן שאפשר להבין על הDataset שלנו. בסוף תפקיד המודל הוא להיות הכי קרוב שאפשר לשופטים שדירגו את הדירוגים המקוריים, ולכן כל פיסת מידע שכזו יכולה רק לתרום.
* אני חושב שהשינוי המשמעותי ביותר היה ההבנה שניתן לעבור לבעיית Multi class classification במקרה שלנו.
* אמנם לא השתמשתי בזה בסוף, אך בחלק של הEmbedding גם יצא לי להבין בזה יותר, ואכן למדתי משהו חדש.
* כמובן וכמובן שלמדתי על רשת סיאמית. הרשת הזו מתאימה בדיוק למקרים כאלה.
* **מחשבה לעתיד** משהו שבאמת חשבתי שיוכל להוסיף לנו, אבל אין זמן לעשות הכול הוא להשתמש בtriple loss.

מה זה? Triple loss מטרתו לא רק כמו שתמיד אנחנו מנסים לקרב בדמיון את מי שהוא זהה לנו, אלא גם הוא מרחיק את מי שהוא לא אנחנו. ככל ש"נקשה" עליו ונביא דוגמאות יותר דומות אחד לשני (אך בכל זאת שונות), ככה נוכל להפריד את הclass בצורה טובה יותר.

למה זה טוב? זה היה מבדיל לנו בניבויים שבין 2.33 לבין 2.66 ו3. מה עושה בעצם הtriple loss? הוא דואג לנו ל2 דברים. מצד אחד הוא מאוד מקרב בדמיון את אלה שדומים, אך באמצעות הmargin שלו הוא דואג גם להרחיק לנו את אלה שהם יחסית דומים אלינו. למה אני חושב שזה היה כל כך טוב? כי אני חושב שאצלנו הרשת לא כל כך הצליחה להבדיל בין 2.33 למשל ל2.667 והרבה טעויות נוצרו בגלל זה. אם היינו מצליחים באמת להפריד כל class בצורה אבסולוטית, זה היה ממש משפר את המודל (להערכתי).

* אז שוב, למדתי המון המון המון על החלק הזה:

רשתות סיאמיות, עבודה עם אמבדינג, ביצוע קונקט ולתת לרשת "להחליט בעצמה מה טוב לה", 2 מודלי הML של סעיף ד', עבודה מעמיקה ביותר עם הdataset, ניסיון לחשוב מחוץ לקופסא בהקשר של סוג הבעיה ואיך ניתן לפתור אותה, K fold satisfied שחבל לי שלא יצא לי להשתמש בו, אבל מה לעשות שאין בידי כוח חישוב גדול.