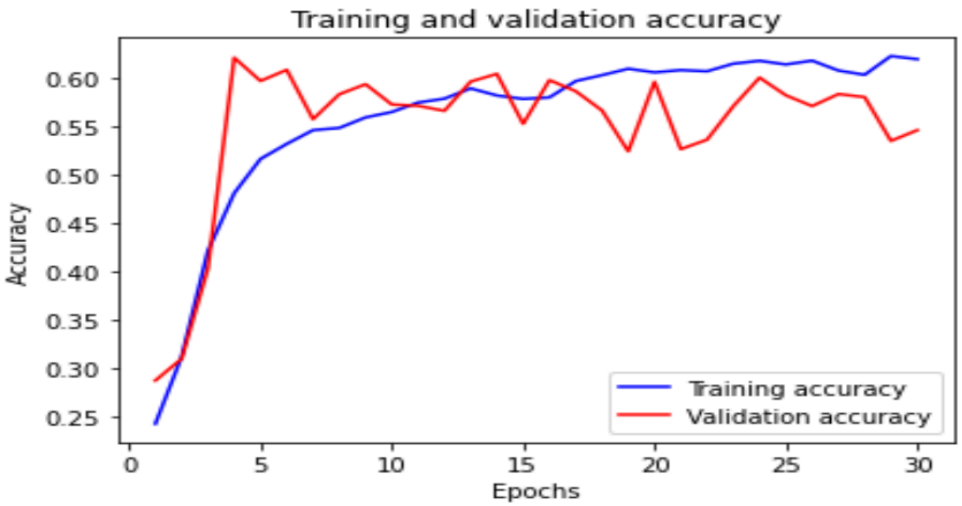
דוח מטלה NLP 2

<u>אימון מודל 1</u>

המודל שביססנו עליו את משימה זו היה מודל support vector machine (SVM).
ראינו שהדאטה של ה- dev וה- train מכילים ציוצים, אז החלטנו לפחות בתור התחלה להשתמש במודלים מאומנים מראש של ה- embedding שאומנו על twitter. כיוון שקיבלנו דיוק מספק, לא התעכבנו על שינוי של המודלים הללו עבור המשימה הראשונה על אף שברור שעשויות להיות למודל השפעות משמעותיות על הביצועים. מבין המודלים של twitter נראה היה שהביצועים הטובים ביותר (לא בפער גדול) היו עבור המודל glove-twitter-200.
הורדנו סמלים ומספרים מהמילים כדי שהן יופיעו במודל בסיכוי גבוה יותר.
אז לכל מילה הותאם וקטור ממודל זה (שגודלו 200), מילים שלא היו במודל קיבלו וקטור בגודל 200 של המספר 6 (קרוב לערך המקסימלי של המספרים במודל), ותחילת / סוף משפט סומנו על ידי וקטור בגודל 200 של המספר 7.
כדי לזרז את האימון השתמשנו בפרמטר - C=0.1.
קיבלנו דיוק של f1_score=0.53 עבור הרצה זו על ה- DEV.

<u>אימון מודל 2</u>

הפעם אימנו מודל FF.
השתמשנו באותו קלט למודל הזה כמו במודל הראשון (embedding על ידי אותו מודל של מילה עם הקונטקסט של ה2 מילים מכל כיוון). לאחר קצת אופטימיזציה, החלטנו להשתמש ברשת בעלת 2 שכבות. השכבה הראשונה בעלת 128 נוירונים והשנייה בעלת 32 נוירונים. בחרנו בפונקציית האקטיבציה ELU (למרות שגם ReLU ו- Leaky_ReLU הפיקו תוצאות טובות).
השתמשנו ב- cross_entropy בתור ה- loss function וכדי להתגבר על חוסר האיזון בין התיוגים (קיימים הרבה יותר תיוגים של 0 מ- 1) העברנו לה גם פרמטר של משקול התיוגים השונים (על פי הדאטה באימון). גודל batch של בין 4-32 היו הטובים ביותר, החלטנו להמשיך עם גודל של 8.
על אף שהדיוק השתפר, הוא היה מאוד וריאבילי בין epoch, אז כדי לייצב את המודל ולמנוע overfit הוספנו שכבת dropout בתדירות נמוכה (0.3). זה עשה עבודה טובה בייצוב ושיפר עוד את הביצועים:



ניתן לראות שהחל מ- epoch 4 הדיוק נע בין 0.5 ל- 0.62.

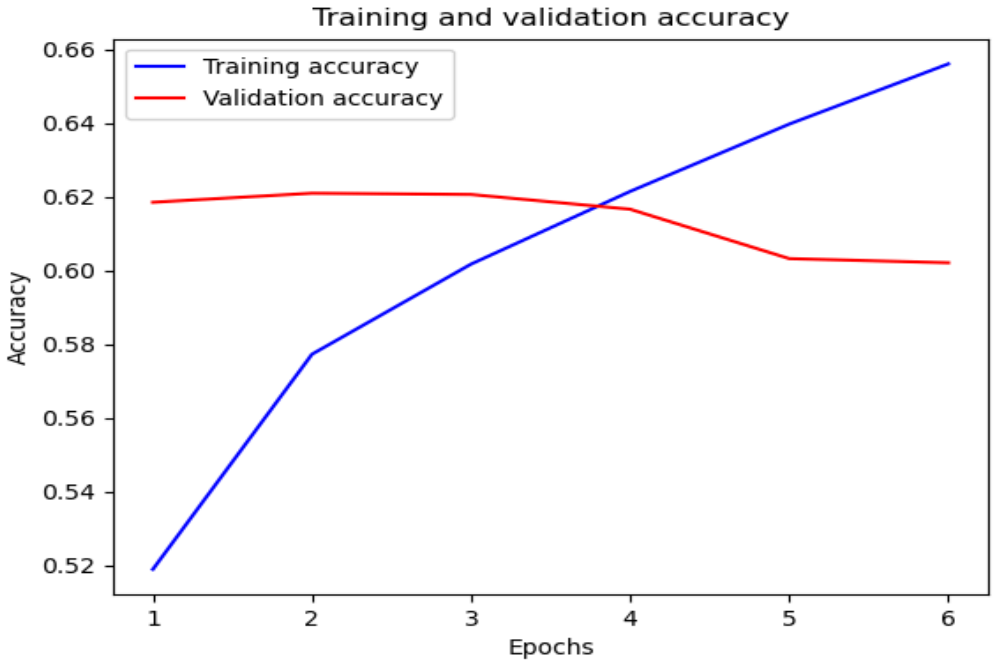
מודל 3:

בחרנו להשתמש ב- LSTM כבסיס למודל זה. השתמשנו במודל חד כיווני פשוט. בתור קלט הכנסנו משפטים, כאשר כל מילה במשפט יוצגה על ידי וקטור מאותו מודל מאומן מראש שהשתמשנו בהם במודל 1 ו- 2. גם כאן, מילים שלא הופיעו במודל סומנו בתור וקטור של המספר 6. במודל זה לא השתמשנו ב- batches כיוון שהגדלים של המשפטים היו שונים זה מזה ודרישה בסיסית של batching זה שהגודל יהיה זהה. לא רצינו להצטרך לעשות padding כי חששנו שזה עשוי לשבש את ביצועי המודל, ובכל מקרה גודל משפט לא מוגבל הוא תכונה טובה. כדי למנוע overfit הוספנו למודל שכבת dropout.

סך הכל, המודל היה בנוי כך:

LSTM עם dropout=0.4, שכבת ביניים בגודל 64, 2 שכבות לינאריות עם ELU ביניהן כפונקציית אקטיבציה. פונקציית ה- Loss שהשתמשנו בה היא cross entropy, וגם כאן העברנו את משקלי התיוגים בתור קלט.

קיבלנו דיוקים שנעים בין 0.61 ועד 0.63, והאימון מתבצע תוך פחות מ- epoch 5.



המודל התחרותי:

התחלנו במודל ה- LSTM מהמודל השלישי וכיוון שרצינו לשפר את הביצועים עקב התחרות, התחלנו בניסוי ותהייה אז דבר ראשון שעלה לנו לראש בעצם זה לממש את bi directional LSTM שדיברנו עליו גם בהרצאה וגם בתרגול ולבחון איך זה ישפיע לנו על הביצועים, ואכן זה העלה בכמעט 5% את הדיוק על קובץ ה-dev.

לאחר מכן בדקנו עבור hidden_dim שונים(64,128,256) אך לא ראינו שיפור ואף היה ירידה ביציבות ואחוזי הדיוק ולכן כדי להמנע מ-overfit ולקבל הכללה טובה יותר כי בכל זאת כמות התיוגים

של 1 היא נמוכה מאד ביחס לכמות התיוגים של 0 בקובץ ה- train החלטנו להשאר עם hidden_dim = 64.

הדבר הבא שעשינו זה הוספת מודלים שיבצעו את האימון על קובץ ה- train. הוספת מודלים מאומנים לייצוג היא חשובה כדי שניתן יהיה להכליל את הממצאים שלנו גם כאשר חלק מהמילים לא מוכרות לחלק מהמודלים. כך אנחנו מבטיחים שבסיכוי גבוה יהיה מודל אחד לפחות שיכיל את מילות הקלט שנקבל, ולכן יפיק ייצוג בעל הגיון. המודלים שהוספנו הם:

model_1_name = glove-wiki-gigaword-300
model_2_name = word2vec-google-news-300
model_3_name = glove-twitter-200
model_4_name = fasttext-wiki-news-subwords-300

ומעבר לכך שזה שיפר לנו את אחוזי הדיוק גם באיזור ה- 3%-5% דיוק זה גם ייצב את אחוזי הדיוק על קובץ ה-dev. בנוסף, הוספנו להטמעה גם שדות שנותנים מידע על המילה עצמה (אותיות גדולות, מספרים וכד')

כמו כן, כיוון שהשתמשנו במודלים רבים בתור קלט, הווקטור המייצג כל מילה היה די גדול והיה צריך לוודא שהמודל לא מתבסס רק על חלקים קטנים ממנו ועושה overfit. כדי להתמודד עם זה הוספנו שכבות dropout. הוספנו dropout ב- LSTM, אחרי ה- LSTM ואחרי שכבת האקטיבציה (ReLU). למרות שזה לא שכיח, הוספנו שכבת dropout על הקלט עצמו, כך הכרחנו את המודל להשתמש בכל הווקטור.

תוספת אחרונה שביצענו, בגלל חוסר האיזון בתיוג רצינו להפחית את הלמידה ממשפטים רווי 0. עם זאת, רצינו לאפשר למודל ללמוד על כל המשפטים. אז בהסתברות של 0.3 השמטנו משפטים שיש בהם רק 0, כך שלאורך ה- epochs כל המשפטים נלמדו אך עם פחות משקל.