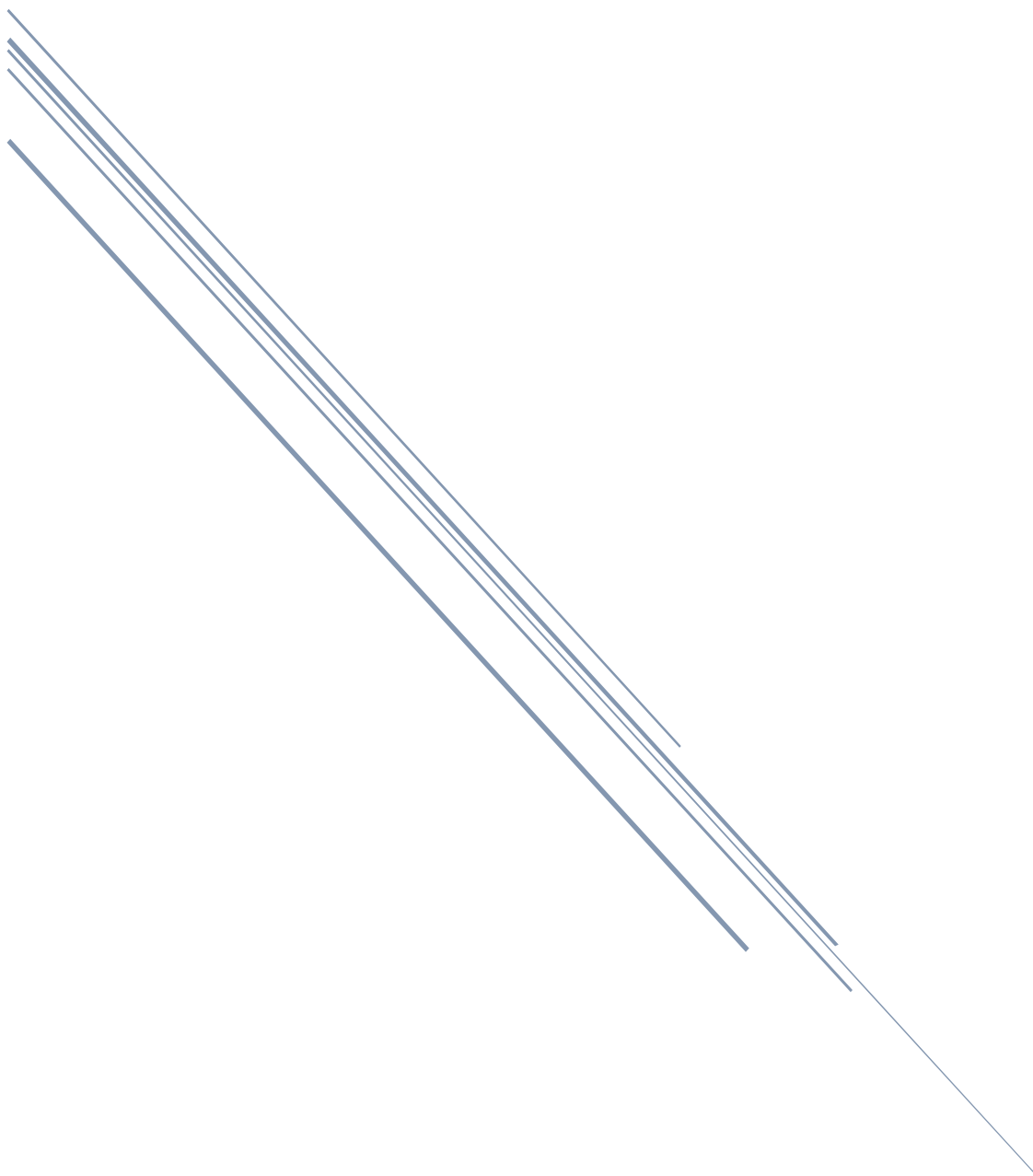


OPINION MINING

דו"ח מסכם



2	רקע
3	פיתוח הרעיון הכללי
3	תכנון מקורי
4	פתרון
4	RNN-RECURRENT NEURAL NETWORK
10	מאגר מידע (DATASET)
11	עיבוד מקדים (PRE-PROCESS)
11	הכנה של כל ביקורת (הורדת כמות FEATURES וסינון רעשים)
11	טכנולוגיות בשימוש
12	הפיכה של ה-DATASET לרצף (SEQUENCE)
12	טכנולוגיות בשימוש
12	שלב אחרון חלוקת המידע
13	טיפה סטטיסטיקה
13	דוגמא לתהליך
15	בניית מודל
15	BASLINE
16	הבנה של ה-BASLINE
18	הרצה של ה-BASLINE ותוצאות
19	שינוי המודל
20	שינוי גודל המילים אשר נלקחות בחשבון
22	שינוי גודל של כל רצף (maxlen)
23	שינוי גודל ייצוג של מילה (Emmbedings)
24	החלפה מ-BLSTM ל-bi-BLSTM
25	הוספת שכבת conv1d ו-maxpooling ולמעשה הפיכה המודל ל-BiLstm-cnn
26	תיקון טעות משלב העיבוד המקדים
28	מחשבות על העתיד

רקע

Opinion mining או בשמו היותר מוכר Sentiment analysis הוא חלק מתחום הסיווג (classification) ומטרתו היא לסווג טקסט לרגשות.

לנו כבני אדם לרוב קל לדעת מלקרוא טקסט או לשמוע שיחה אם מדובר באמרה חיובית או שלילית, אבל למחשב דבר זה לא בא בקלות.

לדוגמא:

בהינתן שתי המשפטים הבאים:

1. The new Superman movie is a terrible one
2. The new Spiderman movie is not so bad as it seems

לבן אדם יהיה קל להבין כי הביקורת הראשונה איננה חיובית אך השנייה כן. למחשב? לא ממש.

כאן נכנס התחום של Opinion mining, תחום שמטרתו היא לגרום למחשב ללמוד להבדיל בין רגשות.

בפרויקט זה בוצע ניסיון לעשות זאת על ביקורות של סרטים.

פיתוח הרעיון הכללי

תכנון מקורי

התכנון המקורי היה לבצע את הפרויקט על ידי מימוש של מודל בסיסי אשר איננו קשור לרשות נורונים, הרעיון היה לקרוא ציורים על ידי שימוש ב-Tweepy המהווה wrapper של ה-Twitter streaming API עבור פיתון.

לאחר מכן, לסנן את הציורים אשר קשורים לסרטים ולסוגם בצורה הבאה:

1. עבור כל מילה ביצוע חישוב של הקוטביות שלה (polarity). כאשר בשלב זה ההתייחסות היא לשלושה רמות של קוטביות: שלילית (Negative), ניטרלית (Neutral) ו-חיובית (Positive).

למשל:

עבור המשפט הבא: "the new supermen movie is the worst movie by far" היינו אמורים לקבל:

The	Neutral
New	Neutral
Supermen	Neutral
Movie	Neutral
Is	Neutral
The	Neutral
Worst	Negative
Movie	Neutral
By	Neutral
Far	Neutral

2. עבור כל חלוקה כזאת חישוב של כמות המילים החיוביות וכמות המילים השליליות בציוץ, תוך כדי התעלמות מהניטרליות ולפי החישוב הזה קביעה אם מדובר בציוץ חיובי או שלילי (אם יש יותר מילים שליליות מאשר חיוביות אז הציוץ שלילי, אם ההפך אז חיובי).

במקרה של הדוגמא: הביקורת הייתה מסווגת כשלילית.

הערה: אין התייחסות למחיקת מילות עצירה (stop words) מראש מהסיבה שהן היו נמחקות בשלב הנוכחי כי הן היו מסווגות כניטרליות.

לצורך חישוב הקוטביות של המילים ניתן היה להשתמש במילון אשר מהווה sentiment lexicon ז"א אומרת מחזיק בתוכו מילים והקוטביות שלהם.

הגישה התבררה כבעייתית מכמה סיבות:

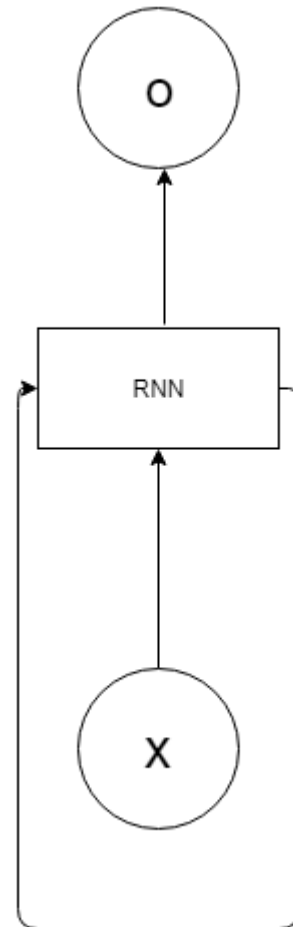
1. אין תלות בין המילים, בשיטה זו ההתייחסות היא למילה בלבד ולא ליחס בינה לבין אחרות.
2. כפל משמעות בשפה האנגלית:
 - בהינתן ציוץ "the new supermen movie is barely satisfactory" לפי החישוב מעלה, אנו נקבל שהביקורת חיובית למרות שהיא אינה כזאת, הסיבה לכך היא ש-'barely' היא מילה ניטרלית מבחינת הלקסיקון.
 - בהינתן שלילה כפולה כמו: "this new venom movie not a bad movie" המודל יראה לנו שהציוץ שלילי, למרות היותו חיובי.

גישה נוספת אשר נשקלה היא שימוש ב-Naïve Bayes לאימון המודל אך גם גישה זאת היא בעייתית שכן, כזכור מודל זה משתמש בהנחת יסוד נורא בעייתית (במקרה שלנו): חוסר תלות בין המילים במשפט.

פתרון

RNN-Recurrent neural network

עבודה עם מודל של Supervised DL . מסוג RNN או אם נדייק RNN עם תאי LSTM.



ישנם מספר סיבות לבחירה זו:

1. מודלי RNN אינם מניחים הנחות על המידע אשר הם מקבלים לסווג.
2. הקשר בין המילים נלמד בעזרת word embeddings ובעזרת יכולת הזיכרון של המודל.

את רשתות ה-RNN ניתן לדמות לתהליך למידה אשר עובר האדם עצמו, הרי כשאנחנו מגיעים להרצאה במהלך הסמסטר, אנחנו לא מתחילים את הלמידה שלנו כל פעם מחדש אלא אנו מתבססים על ידע שלמדנו בשיעורי העבר או בקורסי העבר.

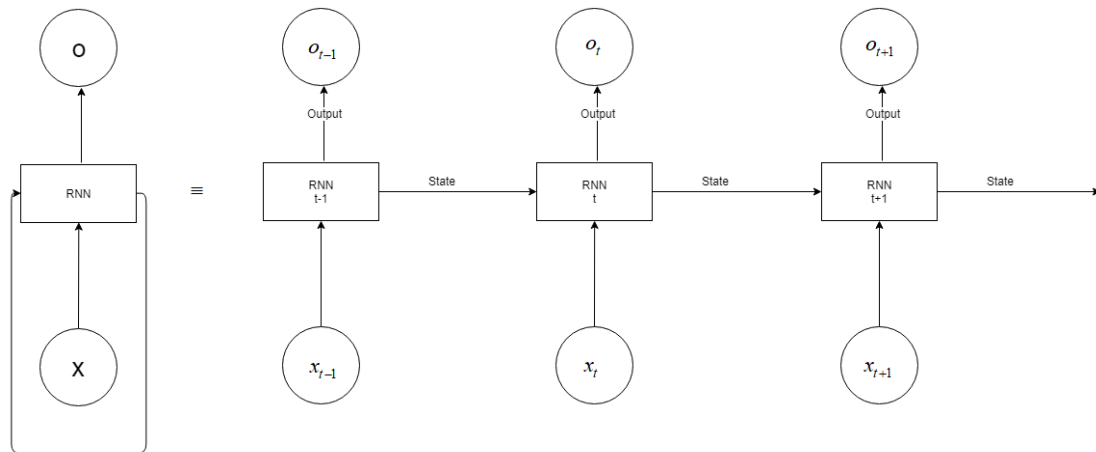
אותו דבר כאשר אנחנו באים לקרוא ספר, אנחנו לא נשענים על המילה שאנחנו קוראים באותו רגע, אלא על שורות, פסקאות ואף פרקים שלמים שקראנו מקודם ובעזרתם בשילוב עם מה שקורה כעת, אנחנו יכולים להבין את העלילה.

זה בדיוק הרעיון אשר עומד מאחורי RNN

על רשתות RNN אפשר לחשוב כעל רשתות בעלות לולאה פנימיות אשר מאפשרות זרימה של מידע משלב אחד של הרשת לשלב העוקב שלו.

למעשה, אנו יכולים לדמות את הרשת הזאת לרשת בעלת העתקים מרובים של עצמה, כאשר כל העתק מעביר את המידע אשר בתוכו לבן שלו.

אם נפתח את הלולאה נקבל:



דבר שמאוד מזכיר רצף כלשהו כדוגמת מערך או רשימה מקושרת.

כך נוצר מצב שהבן יודע מה קרה צעד אחד לפניו ויכול להסיק מסקנות גם מכך. זאת היא בדיוק הסיבה שמודל זה אמור להיות מודל טוב לניתוח סמנטי.

אם זאת, ארכיטקטורת RNN קלאסית היא בעייתית מהסיבה הפשוטה שהיא מתחשבת אך ורק בעבר המידי (2-3 צעדים אחורה) ולא בכל העבר.

משמעות הדבר, תלויות ארוכות מדי (מעל ל- 2-3-gram) לא נקלטות על ידי ארכיטקטורה זאת.

דוגמאות לבעייתיות הארכיטקטורה:

אם נחזור לאנלוגיה של למידה ונניח שהזיכרון שלנו עובד בתצורה של RNN קלאסי:

1. נניח ששיעור הוא צעד אחד אחורה בעבר.
2. נניח שבשיעור ה-3 במהלך הסמסטר למדנו על פונקציות אנונימיות.
3. נניח שבשיעור ה-7 אנחנו לומדים על Pipelining ובכדי לבנות pipeline אנו צריכים את הידע מהשיעור ה-3.

במצב זה, אנחנו לא נזכור את השיעור ה-3.

דוגמא נוספת יכולה להיות הצורך בחיזוי מילה:

נניח ויש לנו את המשפט הבא:

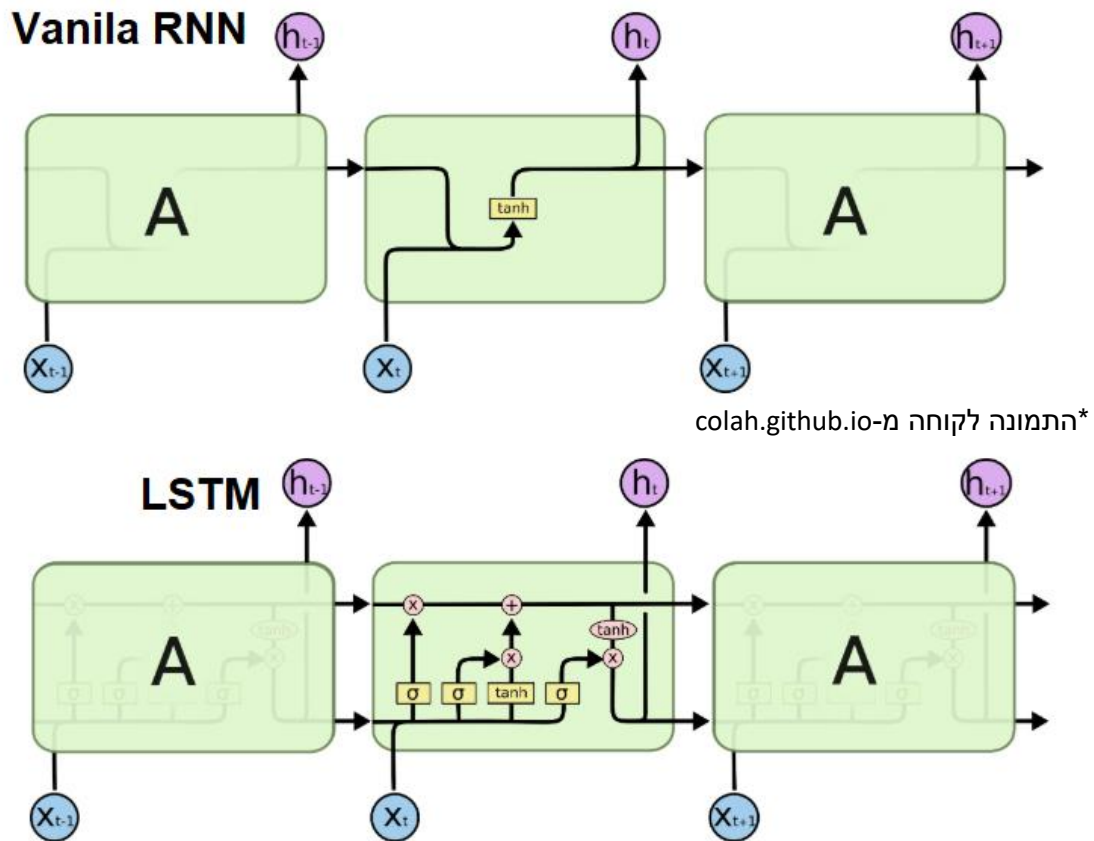
"מיכאל הוא גבר בן 30 מניו-יורק, מיכאל הכיר את אנה בקפה הפינתי, אנה לומדת אדריכלות במכללה ניו-יורק. אנה סיפרה למיכאל שהיא אוהבת ____."

היינו מעוניינים שהרשת תחזה "אדריכלות" מההקשר של "לומדת אדריכלות". אך במקרה של רשת RNN קלאסית דבר זה איננו אפשרי שכן, איננה בנויה לעבודה עם תלויות ארוכות טווח.

הפתרון:

מעבר לתאי LSTM או Long Short Term Memory (תאי זיכרון לטווח קצר)

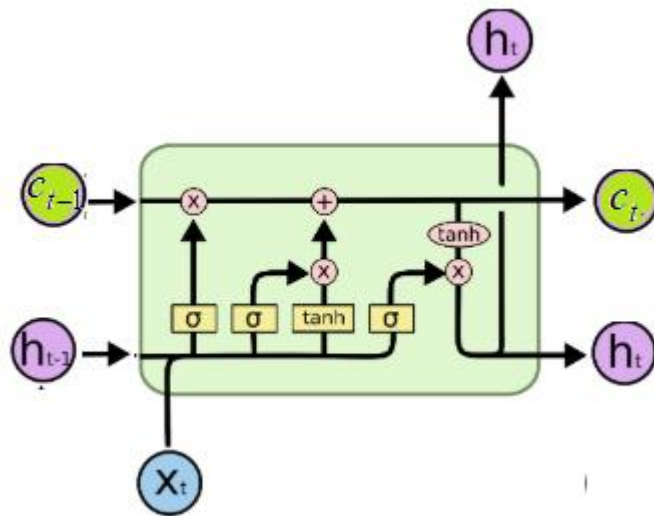
כפי שראינו ארכיטקטורת RNN קלאסית מתקשה מאוד עם תלויות ארוכות. LSTM באה לפתור את הבעיה על ידי הוספת שערים אשר משפיעים על הזרימה של המידע (כמות) מתוך התא ובתוך התא, על הזיכרון ועל השכחה של התא.



אם נתבונן בתמונה מעלה אנו נראה את ההבדל הפיזי ביניהם:

1. מעבר מידע בין התאים:
בעוד שבארכיטקטורה הקלאסית ישנו קו מידע אחד אשר מחבר בין כל התאים ואחראי על העברת פלט בין התאים (הפלט של התא מועבר לתא הבא אחריו), ב-LSTM יש קו מידע נוסף אשר אחראי על העברת הזיכרון אשר התא הקודם זכר.
2. פעולות בין וקטורים:
ניתן לראות כי בארכיטקטורה הקלאסית אין פעולות אשר מתבצעות בעוד שבחדשה התווספו מספר (כמו חיבור למשל).
3. שכבות/שערים: ניתן לראות כי בארכיטקטורה הקלאסית ישנו שער אחד בלבד בעוד שבארכיטקטורה החדשה יש 4.

הסבר מעמיק:



1. נתחיל מהבסיס והוא הבנת הסמלים:
• x_t - רצף/וקטור הקלט עבור יחידה t .

• h_t - הפלט של תא t .

• c_t - הזיכרון/מצב המעודכן של תא t .

• σ - שכבת sigmoid.

• \tanh - שכבת tanh.

• \times - כפל של מידע.

• $+$ - חיבור של מידע.

2. הסבר על השכבות:

2.1. מדוע נבחרה שכבת ה-tanh :

אחת מהבעיות של RNN מסורתי ובכלל באימון של רשתות נוירונים הוא בעיית ה"גרדיאנט הנעלם" או בעיית ה-vanishing gradient problem, לפני שנדבר על בעיה זו, נזכר בכמה מילים איך הרשת באמת לומדת:

בזמן אימון הרשת, בכל פעם שמסתיים לו forward propagation ז"א המידע זורם לו מתחילת הרשת לסופה וכאשר הרשת מחזירה את תוצאות החיזוי שלה, מחושב ה-Loss או למעשה היחס בין הפלט שציפינו שיהיה לבין החיזוי של המודל.

לאחר חישוב ה-Loss מתחיל תהליך הפוך הנקרא backward propagation. בתהליך זה, מחושבים gradient-ים עבור פונקציית ה-Loss ביחס למשקלים ובעקבות החישוב שלהם מתבצעים עדכונים של המשקלים לכל אורך הרשת (בגלל זה גם התהליך נקרא back כי הרשת חוזרת מהסוף להתחלה מחשבת gradient ו מעדכנת את המשקלים).

לצורך הפשטות נוכל להסביר את תהליך ה-backward בצורה הבאה:

1. המודל מתחיל מלהסתכל על השכבה האחרונה שלו ולבדוק את הפלט של כל אחד מהנירונים.
2. הגרדיאנט בודק איזה מהנירונים צדק בתחזית שלו ואיזה מהם טעו.
3. הגרדיאנט מגדיל את המשקל של אותו נירון אשר צדק ומוריד מהמשקל של הנירונים אשר טעו.
4. כתוצאה מסעיף 3 הגרדיאנט צריך לעדכן את כל המשקלים של החיבורים אשר מתחברים אליו מהשכבה הקודמת.
5. סעיף 4 ממשיך בלולאה אחורה עד אשר המודל חזר לשכבת הקלט.

מספר דגשים:

1. אנו מעוניינים לתת יותר משקל או במילים אחרות להתייחס יותר לנירונים אשר צדקו ולהתייחס פחות לאלו אשר לא.
2. בפועל, החישוב של השינוי הנצרך נעשה בעזרת נגזרות חלקיות (loss ביחס למשקלים) ובעזרת כלל השרשרת.
3. הנוסחה של השינוי היא $\text{new weight} = \text{old weight} - (\text{learning rate} * \text{gradient})$
4. המטרה הסופית היא הנמכת פונקציית ה-Loss.

כעת, אחרי שהבנו טיפה את התהליך, הגיע הזמן לדבר על הבעיה:

באופן כללי, מדובר בבעיה רצינית אשר נתקלים בה רבות כאשר באים לאמן רשת ניורונים. הבעיה כוללת את המשקלים או ליתר דיוק את השינוי של המשקלים.

לעיתים, חישוב הגרדיאנט בשכבות המוקדמות יותר של הרשת נהפך לקטן, דבר אשר בתורו גורם לשינוי לא משמעותי של המשקל, שינוי כל כך לא משמעותי שמבחינת הרשת, החיבור נשאר בעל אותה משמעות כמו שבעבר, דבר אשר לא באמת תורם למאמץ להקטין את ה-Loss.

הבעיה נוצרת מהסיבה הבאה: בסופו של דבר, אותו הגרדיאנט הוא מכפלה של תוצאות הנגזרות החלקיות (ביחס למשקל) של כל שלב ושלב אשר קדמו לאותו נירון בחישובים (או למעשה, אשר נמצאים בשכבות מאוחרות יותר משלו).

במילים אחרות: ככל שהנירון נמצא בשכבה מוקדמת יותר, כך הגרדיאנט שלו תלוי בהרבה יותר גורמים אשר קדמו לו ומוכפל בהרבה יותר גורמים אשר יכולים להקטין אותו.

נניח שכל תוצאות העבר שעליהן המכפלה מסתמכת קטנות, שבריים. אז המכפלה עצמה גם תהיה שבר אבל שבר שהוא עוד יותר קטן מהם.

כעת, יש לנו מספר קטן (שגם אותו אנחנו מכפילים במספר קטן שהוא "learning rate" או גודל הצעד שאנחנו מעוניינים לקחת) וכתוצאה נקבל שינוי קטן ולא משמעותי למשקל.

בעיה מקבילה אשר קיימת היא :

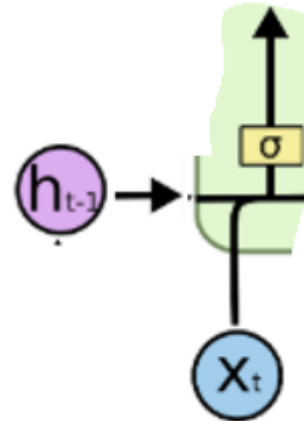
Exploding gradient שבה תוצאות העבר גדולות מ-1 והכפלתם גורמת לתוצאה לגדול עוד יותר, דבר אשר גורם בסופו של דבר לשינוי גדול מדי למשקל ואפשרות של דילוג מעל המצב האופטימלי ולפעמים אף הגדלת ה-Loss.

הסיבה לכך ש-Tanh היא פונקציה אשר פותרת את הבעיה היא שהנגזרת השנייה שלה היא כזאת שהיא לאורך זמן רחוקה מה-0.

2.2. הסיבה לבחירה ב-sigmoid היא הנרמול של המידע בין 0 ל-1, דבר אשר יכול לעזור לשכבה להחליט מה לזכור ומה לא.

3. שלבים עיקריים:

3.1.

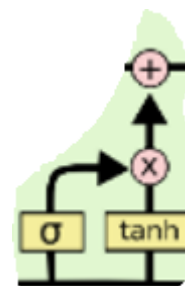


בשלב זה, השכבה מקבלת את הקלט הנוכחי ואת התוצאה של הנוירון הקודם. ועל בסיס המידע החדש מחליטה מה לעשות עם המידע שכבר שמור בתוכה. התהליך מתבצע בצורה הבאה:

ומעבירה כל מרכיב של הקלט דרך שער ה-sigmoid אשר מחליט מה "לשכוח" (להוציא 0-) ומה "לזכור" או כמה לזכור מהפלט הישן, כאשר 1 אומר לזכור הכל.

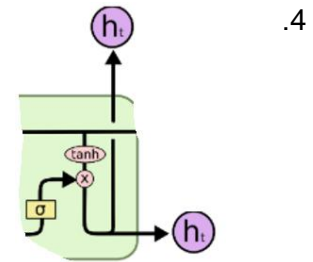
לאחר ההחלטה מה לזכור ומה לשכוח המידע מועבר לפס הזיכרון תוך כדי שהוא מוכפל בזיכרון אשר הגיע מהתא הקודם (הכפלת וקטורים).

3.2.



בשלב זה, השכבה מחליטה מה לזכור מהאינפורמציה החדשה אשר התקבלה בצורה הבאה:

- שער ה-sigmoid מחליט מה מהקלט רלוונטי וכדאי לשמור ומה פשוט כדאי לשכוח.
- שער ה-tanh מייצר וקטור המייצג את כל המועמדים לשמירה מהמידע החדש שהגיע.
- הכפלה בין התוצאה של שתי השערות יוצר את המידע אשר ישמר במודל.
- לבסוף, המידע החדש מתחבר לזיכרון הישן.



השלב האחרון, הוא השלב שבו מתבצעת ההחלטה של איזה מידע לפלוט החוצה ואיזה לא.

הפלטפורמה שנבחרה לבצע את העבודה היא keras עבור python. הסיבה לכך היא הנוחות שבשימוש, keras משמשת כ- high level wrapper עבור מספר פלטפורמות של nn כאשר בפרויקט זה השימוש הוא ב-tensorflow כ-backend.

מאגר מידע (Dataset)

בתור התחלה השתמשתי בפרויקט ב- Large Movie Review Dataset v1.0 של IDMB, מסד אשר מחזיק בתוכו כ-50000 סרטים (עבור supervised learning) כולל הקוטביות שלהן. ועוד כ-50000 ביקורות ללא תוויות עבור (unsupervised learning).

מחולק ל-2 חלקים שווים זה לזה, האחד לאימון והשני לבדיקה.

העבודה התבצעה מול החלק שקשור ל- supervised learning.

באוסף אין יותר מ-30 ביקורות עבור כל סרט, הסיבה לכך היא שישנה קורלציה ישירה בין הסרט לביקורות עליו. ביקורות ניטרליות לא הוכנסו למאגר.

החלוקה של המאגר בוצעה בצורה הבאה:

1. ביקורות שהרייטינג שלהם גדול מ-7 סווגו כחיוביות.
2. ביקורות שהרייטינג שלהם קטן מ-5 סווגו כשליליות.

תחילה התוכנית מבצעת הורדה של המאגר מהשרתים, במהלך ה-Preprocessing כל ה-50000 עוברות עיבוד מקדים ולאחר מכן מתבצעת חלוקה מחדש. ל-16500 ביקורות לבדיקה ו-33500 לאימון.

מחשבה לעתיד: הכנסה של שתי מאגרים נוספים:

1. מאגר מידע המכיל בתוכו ביקורות מתוך האתר rotten tomato's.
2. מאגר מידע המכיל בתוכו ציוצים מתוך Twitter.

התייחסות אליהם בשלב מאוחר ביותר בדו"ח.

עיבוד מקדים (Pre-process)

המשימה העיקרית בתהליך העיבוד המוקדם הוא לתרגם את הטקסט לתצוגה מספרית, תצוגה כזאת שהמודל ידע לעבוד איתה.

הכנה של כל ביקורת (הורדת כמות features וסינון רעשים)

1. כשלב ראשון, בוצעה הורדה ופתיחה של ה-dataset בתור קובץ tar ישירות משרתי Stanford אשר אחראיים על המסד.
2. השלב השני הוא קריאת הביקורות והתוויות כאשר: כל תווית חיובית מתורגמת ל-1 וכל תווית שלילית ל-0.
3. התוויות והביקורות נשמרות בשתי רשימות נפרדות.
4. לכל ביקורת בוצעו השלבים הבאים:
 - 4.1 הורדת כמות המילים (Features) בעלות אותה המשמעות או למעשה התייחסות אליהם כאל קבוצה של מילים אשר צריכות לעבור ניתוח כמילה אחת. ויצירת תוכן עקבי, בשביל לעשות זאת בוצעו הדברים הבאים:
 - 4.1.1 הפיכת כל המילים לאותיות קטנות (Lowercase).
 - 4.1.2 הורדת קיצורי שפה כמו למשל: we'll הפך ל- we will , הסיבה לכך שזהו דבר נדרש היא שאם מופיע לנו בביקורות גם we'll וגם we will למעשה, מופיעים לנו שתי וריאציות לאותה משמעות בדיוק.
 - 4.1.3 Lemmatizing של הביקורת:
 - מדובר בתהליך של קיבוץ מילים אשר נמצאים בצורות דקדוק שונות.
 - 4.1.4 הורדה של stop_wrods: מחיקה של מילים אשר חוזרות על עצמן תדיר בכל משפט (כמו: in, is וכו') שכן, הם לא עוזרים למודל שלנו לחזות יותר טוב.
 - 4.2 מחיקת רעש רקע:
 - 4.2.1 הביקורות בסופו של דבר נאספו מדף html בשל כך, חלק מהביקורות בעלות התגית הבאה
 , התגית הזאת היא רעש שלא תורם לנו לדבר.
 - 4.2.2 מחיקה של סימונים שאינם אותיות קטנות.

טכנולוגיות בשימוש

לשם מחיקת הרעש והורדת קיצורי שפה בוצע שימוש בביטויים רגולריים (Regex).
לשם תהליך ה-Lemmatizing בוצע שימוש ב-WordNetLemmatizer אשר נמצא בחבילת NLTK.
לשם מחיקת מילות עצירה נעשה שימוש במילון אשר מכיל מילות עצירה של השפה האנגלית גם הוא נמצא ב-NLTK.

הפיכה של ה-dataset לרצף (Sequence)

השלב הראשון הוא יצירת רצף מכל ה-Dataset, רצף אשר מתייחס אך ורק לכמות מסוימת של מילים שחוזרות על עצמן הכי הרבה בביקורות, כמות זאת תקבע בשלב מאוחר יותר.

- שלב ראשון הוא בניית המילון מה-dataset בשלב זה, כל המילים נשמרים ללא קשר לכמות שבחרנו.
 - ייצוג לפי frequency: כל מילה במילון מיוצגת בתור ID, ככל שהמילה יותר חשובה כך ה-ID נמוך יותר (המילה הכי חשובה תהיה בעלת ID 1).
 - הסיבה לבחירה במספר כלשהו ולא לקיחה של כל המילים היא הרצון לבחור רק את המילים החשובות ביותר, אם מילה חשובה אז כנראה שהיא נמצאת יותר פעמים ב-set (כמובן, שמדובר על מילים אחרי סינון של stop words אשר בוצע בשלב הקודם).
 - בכדי למצוא את המספר האידיאלי, נעשו מספר בדיקות מול אימון המודל.
 - התוצאה של השלב היא מערך דו מימדי שכל מערך בתוכו הוא רצף אשר נוצר מהביקורת, כאשר בתוך הרצף נמצאים רק המילים אשר שייכים לקבוצת המילים המדוברת מעלה.
- בשלב השני הפיכה של כל רצף לגודל אחיד שאותו נקבע בשלב מאוחר יותר.

- הסיבה לכך שאנחנו לא מרפדים לפי גודל מקסימלי היא כזאת: מצד אחד, אנו מעוניינים לתפוס רק את המילים המשמעותיות באמת לביקורת ומצד שני, אנחנו לא רוצים להכניס רעשי רקע שלא באמת רלוונטיים לביקורת, כמו סיפורים שאנשים נוטים לכתוב בביקורות או שמות של סרטים.
- הסיבה לכך שאנו בכלל מגדירים גודל אחיד היא שאנחנו צריכים להאכיל את המודל שלנו ברצפים בגדלים זהים.
- גם כאן המספר נבחר לאחר מספר ניסויים.
- שלב זה התבצע על ידי קיצוץ רצפים ארוכים מדי והוספה של אפסים לרצפים קצרים מדי לסוף הרצף.

טכנולוגיות בשימוש

כל התהליך בוצע בעזרת Tokenizer אשר מגיע כחלק מ-KERAS.

שלב אחרון חלוקת המידע

השלב האחרון בתהליך העיבוד המוקדם הוא חלוקה של המידע ל-train ו-test, החלוקה היא סטנדרטית 33 אחוז ל-train ו-66 ל-test.

תוצאת השלב הזה היא:

תוויות	ביקורות	
33,500 תוויות של 0 או 1	מערך דו מימדי של 33,500 רצפים, כאשר כל רצף בגודל 140 כל אחד.	אימון
16,500 תוויות של 0 או 1	מערך דו מימדי של 16,500 רצפים בגודל 140 כל אחד.	בדיקה

כל המידע שמור במערכי numpy.

טיפה סטטיסטיקה

תכונה	ערך
כמות ביקורות	50,000
כמות ביקורות חיוביות	25,000
כמות ביקורות שליליות	25,000
הביקורת הארוכה ביותר לפני עיבוד	13704
הביקורת הארוכה ביותר אחרי עיבוד	9149
הביקורת הקצרה ביותר לפני עיבוד	32
הביקורת הקצרה ביותר לאחר עיבוד	17
סך המילים ב-dataset לפני עיבוד	65471551
סך המילים ב-dataset לאחר העיבוד	40543254

כפי שניתן לראות, הצלחנו להוריד את כמות המילים הכוללת.

דוגמא לתהליך

אם ניקח את הביקורת הבאה:

Bromwell High is a cartoon comedy. It ran at the same time as some other programs about school life, such as "Teachers". My 35 years in the teaching profession lead me to believe that Bromwell High's satire is much closer to reality than is "Teachers". The scramble to survive financially, the insightful students who can see right through their pathetic teachers' pomp, the pettiness of the whole situation, all remind me of the schools I knew and their students. When I saw the episode in which a student repeatedly tried to burn down the school, I immediately recalled at High. A classic line: INSPECTOR: I'm here to sack one of your teachers. STUDENT: Welcome to Bromwell High. I expect that many adults of my age think that Bromwell High is far fetched. What a pity that it isn't!

ניתן לראות כי יש כאן רעש רב כמו, "",, וכו'. כמו כן, ניתן לראות הרבה stop words.

לאחר התהליך של מחיקת הרעש והורדת כמות המילים, נקבל:

bromwell high cartoon comedy ran time program school life teacher year teaching
profession lead believe bromwell high satire much closer reality teacher scramble survive
financially insightful student see right pathetic teacher pomp pettiness whole situation
remind school knew student saw episode student repeatedly tried burn school immediately
recalled high classic line inspector im sack one teacher student welcome bromwell high
expect many adult age think bromwell high far fetched pity

לאחר התהליך של תרגום לרצפים אנו נקבל מטריצה דו-מימדית, שכל שורה בה היא ביקורת, הביקורת שלנו (שורה) תראה כך:

[223, 656, 94, 1848, 7, 1496, 258, 31, 1191, 46, 3384, 4778, 209, 162, 223, 1763, 19, 2293, 453, 1191, 1665, 9104, 4889, 643, 14, 102, 976, 1191, 123, 397, 2836, 258, 524, 643, 115, 165, 643, 2980, 613, 1705, 258, 1038, 11549, 223, 218, 110, 3077, 63, 7407, 4, 1191, 643, 2158, 223, 402, 36, 604, 398, 30, 223, 131, 6773, 2036]

ניתן לראות כי יש מספרים אשר חוזרים על עצמם, משמעות הדבר הוא שמדובר באותה המילה.

ולאחר השלב האחרון של ריפוד או לחילופין קיצוץ הביקורת שלנו תראה כך:

```
[ 223 656 94 1848 7 1496 258 31 1191 46 3384 4778
209 162 223 1763 19 2293 453 1191 1665 9104 4889 643
14 102 976 1191 123 397 2836 258 524 643 115 165
643 2980 613 1705 258 1038 11549 223 218 110 3077 63
7407 4 1191 643 2158 223 402 36 604 398 30 223
131 6773 2036 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0]
```

בניית מודל

Baseline

בתור בסיס השוואתי לכל הניסויים שלנו, אנו נשתמש במודל המוצע על ידי keras תחת דוגמאות ב-

imdb_lstm.py

החלקים של המודל בקוד:

```
max_features = 20000
# cut texts after this number of words (among top max_features most common words)
maxlen = 80
batch_size = 32

print('Build model...')
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

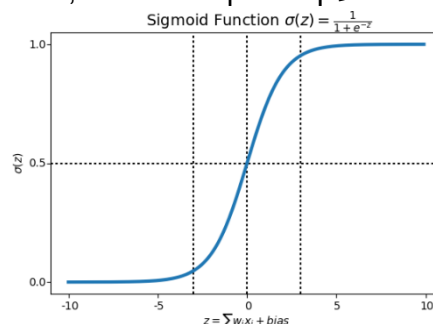
# try using different optimizers and different optimizer configs
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

print('Train...')
model.fit(x_train, y_train,
        batch_size=batch_size,
        epochs=15,
        validation_data=(x_test, y_test))
```

נכניס שינוי קל , במקום validation data נשתמש ב-validation split של 20 אחוז שמטרתו, לקחת 20 אחוז מהמידע אשר נכנס אל המודל ובמקום להתאמן עליו, לשמור אותו בצד ולבדוק עליו את ההתקדמות של המודל (דבר זה טוב בכדי לחשב את פונקציית ה-loss וה-acc על מידע שהמודל לא ראה בעבר ובכך לבדוק אם יש לנו מצב של Overfitting או Underfitting).

הבנה של ה-baseline

1. גודל המילון שבו הם ישתמשו הוא 20000 כלומר, ההתייחסות היא ל-20000 מילים התדירות ביותר.
2. גודל של כל ביקורת הוא 80.
3. גודל batch הוא 32
 - Batch_size הוא מספר הדוגמיות שמועברות לרשת בפעם אחת. למעשה, בכל forward propagation יועברו 32 רצפים לאימון ברשת.
 - למעשה, ניתן לחשוב על Batch כקבוצה של דוגמיות אשר מועברות יחדיו לאימון.
 - עקרונית ככל שה-Batch_size יותר גבוה כך ריצה אחת הלך וחזור (forward & backward propagation) אמורה לקחת פחות זמן, אך אין זה אומר שאיכות האימון תהיה טובה יותר. למעשה, מדובר ב-tradeoff בין איכות לבין מהירות וכמות זיכרון.
4. כמות epochs הוא 15, כלומר-הרשת תבצע 15 מעברים של כל המידע אחורה וקדימה (forward & backward).
 - למעשה, כאשר ה-batch_size גדול יותר אז ל-epoch אחד לוקח פחות זמן עד שהוא נגמר.
5. שכבות המודל:
 - 5.1 שכבת ה-embeddings:
 - שכבה זו היא השכבה אשר אחראית על יצירת word-embeddings למעשה, הפיכת כל מילה לוווקטור רב-ממדי, הרעיון מאחורי השכבה הזאת היא לתפוס את הסמנטיקה של המילים, כך שבסופו של דבר הוווקטורים של המילים אשר קשורות זו לזו יהיו קרובים יותר מבחינה גאומטרית.
 - גודל השכבה כאן הוא 128. כלומר, כל מילה תתורגם לוווקטור בעל 128 ממדים.
 - 5.2 שכבת LSTM:
 - גודל 128-משמעות הדבר היא שכמות הרכיבים בשכבה הוא 128
 - Dropout: טכניקה שבה המודל מוותר על חלק מהנירונים וזאת בכדי לנסות ולמנוע Overfitting של המודל.
 - Dropout זריקה של 20 אחוז מהחיבורים הנכנסים (x) או למעשה, ויתור על כ-20 אחוז מהקלטים אשר נכנסים לשכבה.
 - recurrent dropout של 20 אחוז, משמעות הדבר הוא ויתור על 20 אחוז מהחיבורים בין התאים בשכבה זו.
 - 5.3 שכבת dense (fully connected) בעלת פלט/נירון 1 בלבד.
 - משמעות השכבה היא חיבור של כולם לכולם, במקרה שלנו כל הנירונים של שכבת ה-LSTM מתחברים לנירון היחיד של השכבה הנ"ל.
 - 5.4 פונקציית ההפעלה:
 - משמעות פונקציית ההפעלה היא:
 - רשמית מדובר במיפוי של כל הקלטים של נירון לפלט שלו.
 - בפועל: נרמול הפלט והחלטה לגבי הפעלת הנירון לטווח מוגבל.
 - פונקציית ההפעלה של השכבה היא sigmoid והיא מגבילה את הפלט לטווח שבין 0 ל-1.
 - . זאת אומרת, ככל שהערך יהיה קרוב יותר ל-1, הנירון יהיה יותר "מופעל".



כאשר:

- ערכים שלילים ינורמלו כך שהם יהיו קרובים ל-0.
- ערכים חיוביים ינורמלו ככה שהם יהיו קרובים ל-1.
- ערכים הקרובים ל-0 ינורמלו למספר כלשהו בין 0 ל-1.

5.5. Function loss:

- לפני שנבין מה היא פונקציית loss עלינו להבין מה הוא loss: ובכן, כאשר המודל שלנו יבצע forward propagation אחד בשכבה האחרונה הוא יפלוט מה הוא חושב, האם המשפט הוא חיובי או שלילי או ליתר דיוק את ההסתברות שמדובר במשפט חיובי או שלילי.

- במקרה שלנו ה-loss הוא חישוב של ההבדל בין מה שהמודל חושב על המשפט לבין מה שהוא בפועל (לפי התווית שנתנו לו) או למעשה loss הוא מרחב הטעות או אם נרצה ההפרש בין הניבוי של הרשת לבין הפועל.
- ה-Loss function אחראית על חישוב ה-loss.

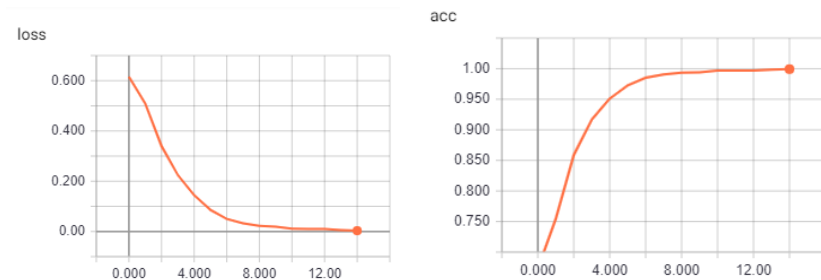
5.6. Optimizer:

- ה-Optimizer הוא לב הלמידה שלנו, מטרת ה-Optimizer היא למצוא דרך להקטין את ה-loss וזאת על ידי כיוון המשקלים וה-bias של כל נירון ברשת.
 - ה-Optimizer מבצע זאת על ידי חישוב וקטור הנגזרות החלקיות (gradient) והליכה כנגד הכיוון שלו (למעשה, חיפוש מינימום של פונקציה אשר למדנו בחדו"א 2).
 - אחרי שהעברנו ברשת את כל המידע שלנו (Forward propagation), מתבצע תהליך הפוך של (Backward Propagation) שבו מחושבים 'צעדים' שבהם המודל ילך לכיוון המינימום על ידי שינוי המשקלים.
- השינוי של המשקלים נעשה לפי הנוסחה:
- $$\text{new weight} = \text{old weight} - (\text{learning rate} * \text{gradient})$$
- ה-learning rate הוא גודל הצעד שעלינו לבצע בכל פעם.
- ה-Optimizer אשר נבחר כאן הוא adam - מדובר בווריאציה של GSD.

הרצה של ה-baseline ותוצאות

הרצה של ה-baseline הביאה לדיוק של 83.86060606060606 אחוז.

אם נתבונן על פונקציות ה-Loss וה-Acc נראה:

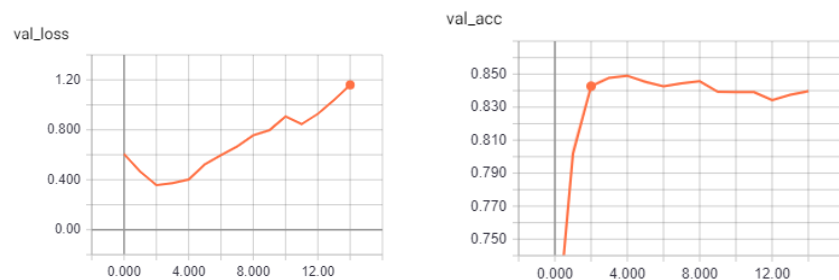


בסוף הצעד ה-15 ניתן לראות כי ה-acc 0.999253749847412 (כמעט 1) וה- loss הוא 0.00318928831256926 (כמעט 0) על פניו, נראה אידיאלי, אם זאת זה לא המצב.

יש לנו כאן אינטואיציה חזקה ל-Overfitting מצב שבו, המודל התאמן יותר מדי טוב על המידע שהועבר אליו כך שבמקום ללמוד ממנו הוא זוכר אותו בעל פה.

ניתן לדמות זאת למצב שבו סטודנט לומד את אותו סט של תרגילים שוב ושוב, עד שהוא זוכר אותו היטב אך ברגע האמת, במבחן הוא לא מצליח לפתור כמעט כלום או פותר בצורה ממש גרועה.

אם נתבונן ב val acc וב- loss acc (פונקציות אשר מחושבות על ה-validation data - אותם 20 אחוז שהושארו בצד) נראה שאכן זה המצב:



ניתן לראות שהמודל שלנו הצליח לצפות יפה את המידע שהוא אומן עליו אך בפועל, על מידע שהוא לא אומן עליו התוצאה משמעותית פחות טובה. משמעות הדבר, אכן יש לנו Overfitting.

המודל אומנם מביא תוצאה יפה, אך מהגרפים ניתן לראות כי אפשר להשתפר.

שינוי המודל

מחשבות לשיפור:

1. פרמטרים הנכנסים למודל:

לפני תחילת ביצוע שינויים במודל עצמו, החלטתי לבדוק את התפקוד של המודל עם שינוי בפרמטרים הבאים:

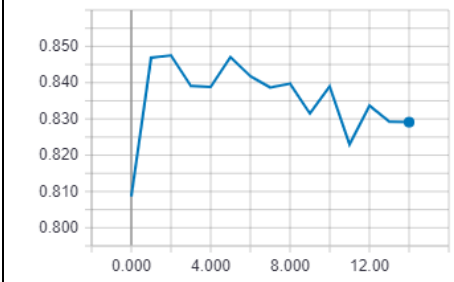
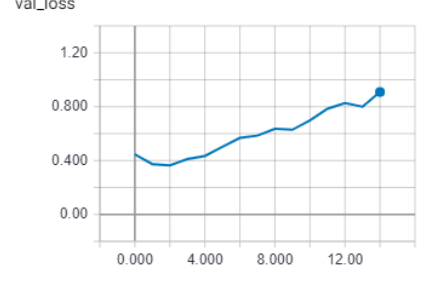
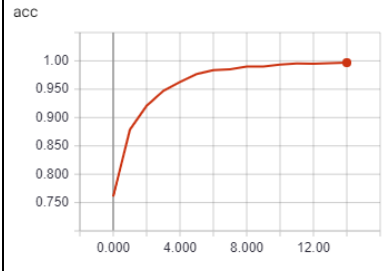
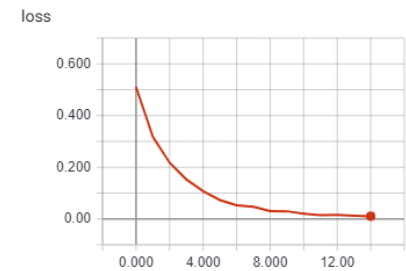
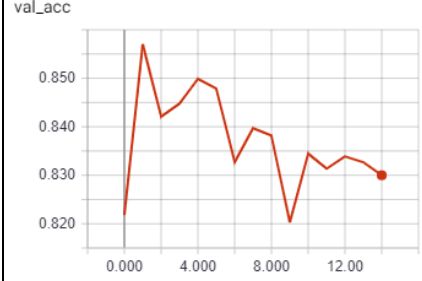
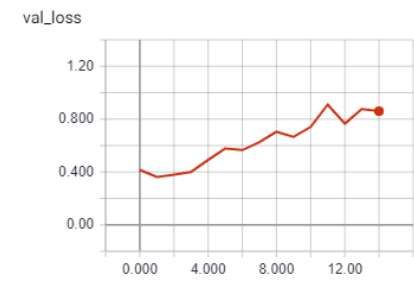
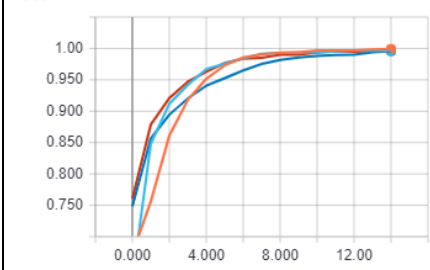
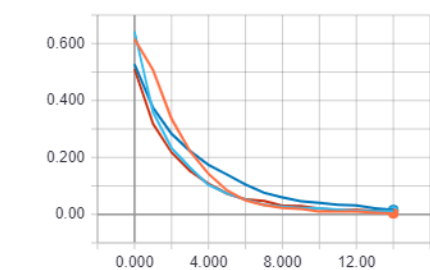
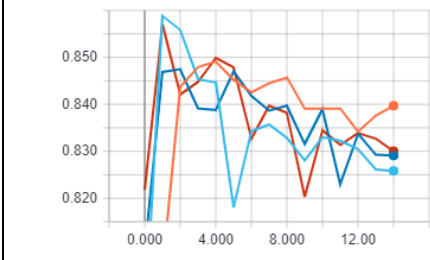
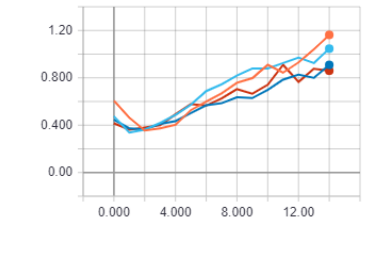
- כמות מילים מקסימלית אשר נלקחות בחשבון ע"פ תדירות מרבית:
- אורך מקסימלי של רצף: מדובר ב-Tradeoff בין הכנסת רעש מיותר לבין פספוס דברים חשובים, וכמובן על "הרעבת המודל"
- גודל של הייצוג הווקטורי של מילה (embeddings): הסיבה זהה לסעיף מעלה.

2. שינוי המודל עצמו:

- הגדלת כמות הנוירונים ב-LSTM.
- מעבר מ-LSTM ל-bidirectional LSTM-דבר זה עלול לתרום שכן, אז יהיה מעבר על ה'עתיד' ועל ה'עבר'.
- הוספת שכבות Convolution ו-Pooling ולמעשה הפיכת המודל לשילוב בין CNN ל-RNN.
- שינוי פונקציית אקטיבציה.
- שינוי פונקציית אופטימיזציה-עלול להשפיע לנו על השינוי בפונקציית ה-loss.
- שינוי גודל הצעד-דבר אשר יכול להשפיע לנו על קצב הלימוד/שינוי ב-Loss ולמנוע "דילוג" על ה-loss המינימאלי.
- שינוי גודל הקיבוץ (batch)-ב-tradeoff עם epochs יכול להשפיע על מספר דברים כמו מהירות חישוב, Overfitting, וכו'.
- שינוי כמות ה-(epochs)-עלול להשפיע לנו על קפיצות בגרף ועל איבוד אפשרי של מומנטום, Overfitting.
- שינוי ה-dropout או ויתור עליו לחלוטין, דבר זה יכול להשפיע על ה-overfitting.

שינוי גודל המילים אשר נלקחות בחשבון

גרפים	גודל
<div data-bbox="264 264 847 651"> <p>acc</p> </div> <div data-bbox="871 264 1374 651"> <p>loss</p> </div> <div data-bbox="264 674 815 1043"> <p>val_acc</p> </div> <div data-bbox="895 674 1358 1043"> <p>val_loss</p> </div>	20k
<div data-bbox="264 1077 767 1379"> <p>acc</p> </div> <div data-bbox="895 1077 1334 1379"> <p>loss</p> </div> <div data-bbox="264 1402 751 1688"> <p>val_acc</p> </div> <div data-bbox="919 1402 1342 1688"> <p>val_loss</p> </div>	10k
<div data-bbox="264 1688 679 1995"> <p>acc</p> </div> <div data-bbox="943 1688 1366 1995"> <p>loss</p> </div>	15k

<div>val_acc</div>  <div>val_loss</div> 		
<div>acc</div>  <div>loss</div> 	25k	
<div>val_acc</div>  <div>val_loss</div> 		
<div>acc</div>  <div>loss</div> 	כולם יחדיו	
<div>val_acc</div>  <div>val_loss</div> 		

לפי התוצאות הדיוק המיטבי הוא ב-20 אלף וההפסד המינימלי הוא ב-25 אלף.

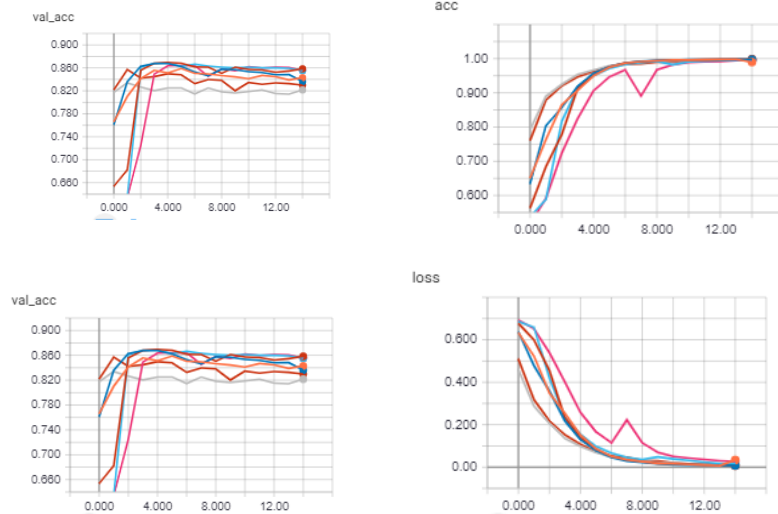
ההחלטה היא לזנוח את ה-20 אלף וללכת לכיוון ה-25, הסיבות לכך:

- ההבדל ביניהם מבחינת תוצאות חיזוי הוא זניח יחסית בעוד שההבדל בהפסד הוא גדול.
- חשוב לנו להוריד את ה-Loss למינימום האפשרי.

הערות:

- כרגע, בכל מקום שמצוין Loss ותוצאות חיזוי/דיוק מיטבי הכוונה לפרמטרים של val loss ו-val acc שכן, הם האינדיקציה האמתית להצלחת המודל.

שינוי גודל של כל רצף (maxlen)

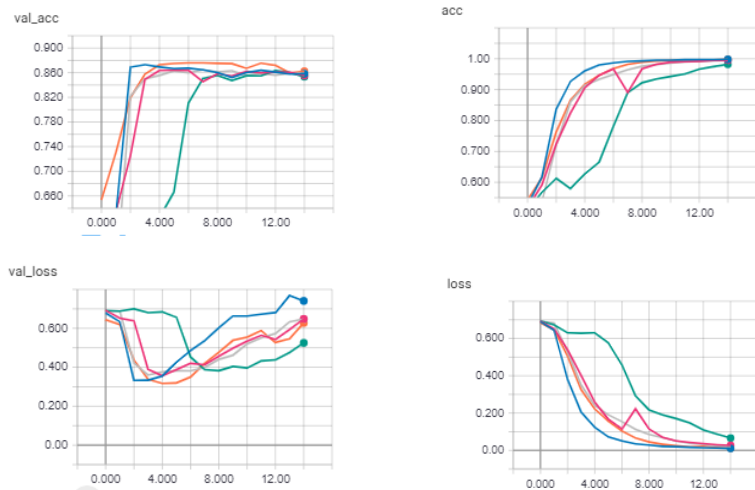


מקרא:



לאחר בחינת הנתונים ובהתאם לשיקולים כמו מעלה וכאלו אשר עוד מדובר עליהם מטה, הוחלט ללכת על גודל של 180.

שינוי גודל ייצוג של מילה (Emmbedings)



מקרא:



לאחר בחינת הנתונים ובהתאם לשיקולים אשר פורטו, הוחלט ללכת על גודל של 96.

החלפה מ-BLSTM ל-bi-BLSTM

ניסיון להחליף את כמות הנוירונים בשכבת ה-BLSTM ל-200 ומעבר לשכבת bi-directional-BLSTM.

ההבדל בין שכבה BLSTM ל-bi-BLSTM הוא היכולת לעבור על העבר כמו כן על העתיד, בעוד ששכבת BLSTM רגילה יודעת להתייחס אך ורק לעבר, שכבת ה-bidirectional יודעת להתייחס גם לעתיד או במילים אחרות, עוברת על העתיד ועל העבר יחדיו.

יש לכך יתרון משמעותי בחיזוי מילים ונסביר את זה בעזרת הדוגמא הבאה:

נניח שיש לנו את המשפט החלקי הבא:

The boys saw this spider __ movie and thought it was a great one

נניח שאנו מעוניינים שהרשת תזהה את המילה החסרה, לנו ברור שמדובר ב-man.

אבל אם נסתכל מ'עיניה' של שכבת LSTM רגילה נשים לב, שלה זה לא יהיה כל כך ברור.

שכבת LSTM רגילה עושה Forward LSTM בלבד ז"א בשביל לחזות את המילה יש לה רק את החלק הבא של המשפט:

The boys saw this spider , בהינתן חלק זה בלבד, נורא קשה להבין שהמשך אמור להיות man, ניתן בקלות לחשוב כי הילדים ראו עכביש אמיתי.

לעומת זאת:

שכבת bi-directional יודעת לעשות גם Forward וגם LSTM Backward או במילים אחרות כשהיא באה לחזות את המילה יש לה גם את החלק שבה אחריה.

יש לה גם את : The boys saw this spider וגם את movie and thought it was a great one.

ובמצב זה, יותר קל לראות מה המילה החסרה.

להערכתי, הקשרים בין המילים ילמדו כך יותר טוב על ידי הרשת.

הוספת שכבת conv1d ו- maxpooling ולמעשה הפיכה המודל ל- BiLstm-cnn .
לרוב, משתמשים במודל CNN בכדי לעלות על דפוסים מסוימים בעיקר בתמונות.

בעזרת שכבת ה- conv1D המידע מועבר דרך פילטרים מסוימים ובעזרת המעבר הזה של הפילטרים, נקלטים דפוסים כלשהם (למשל בתמונות: ככה ניתן לזהות אם מדובר בקצוות של אובייקט או ברקע וכו').

להערכתי, העברת המידע משכבת ה-RNN לשכבה זו יכולה לעזור לקלוט דפוסים מסוימים אשר יעזרו לזהות את הדפוסים הרלוונטיים לחיזוי נכון של ביקורת.

Max-pooling היא שכבה אשר נועדה להקטין את המרחביות של המידע, דבר אשר למעשה מקטין את הגודל של הפלט משכבת ה- conv , גורם לרשת להסתכל על אזורים גדולים יותר כל פעם שהיא מבצעת forward prop, מקטין את זמן החישובים ואת כמות הפרמטרים של הרשת, בסופו של דבר השימוש בשכבה זו יכול להקטין את ה-OVERFITTING.

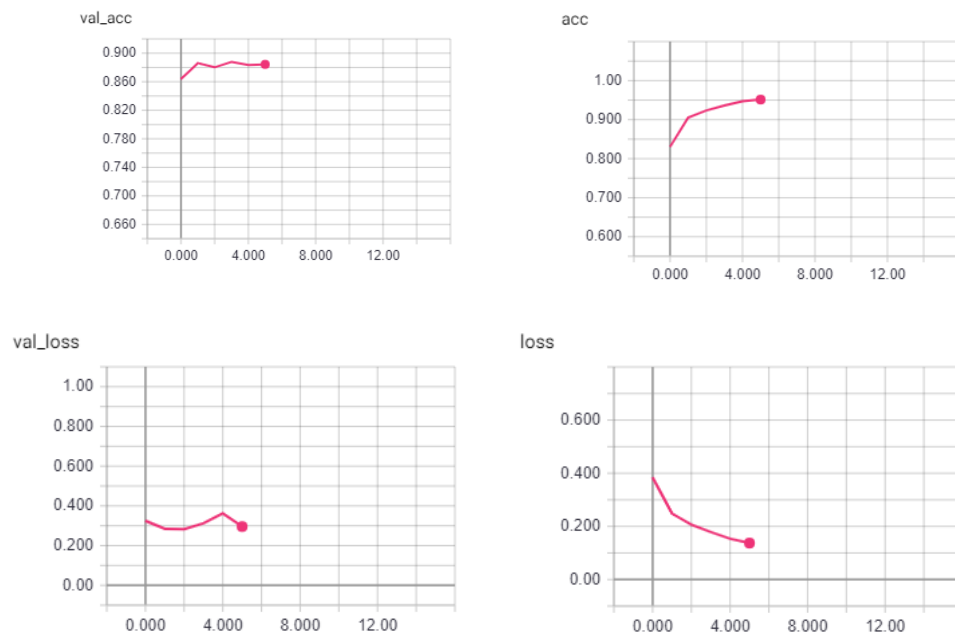
השכבה הזאת, לוקחת מכל 'pool' (חלון פעילות שהיא עוברת עליו כל פעם ומוגדר על ידי הפרמטר pool_size) את הערך המקסימלי (וזאת בניגוד ל- avg pooling שלוקחת את הערך הממוצע).

למעשה על ידי שימוש ב-Max-pooling אנו בוחרים בערכים הכי 'מופעלים' או משפיעים ומתייחסים רק אליהם, אם למשל היה מדובר בתמונה אז היינו לוקחים את הפיקסלים הכי משפיעים וממשיכים לעבוד עליהם.

נוסו מספר קונפיגורציות בעלות כמות פילטרים שונה ובעלות pool בגודל שונה.

כאשר הקונפיגורציה שנבחרה מבין כולם:

שכבת conv1D בעלת 200 פילטרים, ללא padding ועם אקטיבציה $\text{relu}(\max\{0, x\})$ ושכבת max-pool בגודל 4 ללא padding ו- strides דיפולטיבי. ופונקציית אופטימיזציה RMSProp.



תיקון טעות משלב העיבוד המקדים

התברר כי נעשתה טעות בשלב העיבוד מקדים וכחלק מהסרת ה-stop words הוסרו גם המילים 'not' ו-'nor'. אסור לנו להסיר אותן שכן, אנו צריכים שהמודל ידע לסווג נכון שלילה כפולה (not so bad למשל).

כתיקון, מילים אלו הוצאו מהמילון של ה-stop words ובנוסף, נעשו שוב מספר ניסיונות לשפר את המודל לפי הבעיות הנצפות:

1. קיום Overfitting.
2. דילוג על הנקודה האופטימלית עבור פונקציית ה-loss בכל הגרפים, בחלקם יותר ובחלקם פחות, ניתן לראות כי ה-loss קטן עד שלב מסוים שבו הוא מתחיל לגדול בחזרה, זו היא אינדיקציה לכך שאנחנו מדלגים על הנקודה האופטימלית וחשד ל- Exploding Gradient.

נבדקו מספר קונפיגורציות אשר כללו:

1. הוספת שכבת dropout לפני שכבת הפלט-יכול להשפיע על ה-overfitting.
2. שינויים בממדד הלמידה של האופטימיזר-דבר אשר יכול להשפיע על העובדה שאנחנו מדלגים על הנקודה האופטימלית.
3. שינוי גודל ה-pool.

לבסוף התצורה הבאה נבחרה:

```
self.model = Sequential()
self.model.add(Embedding(max_features, 96, input_length=180))
self.model.add(Bidirectional(LSTM(200, recurrent_dropout=0.2, dropout=0.2, return_sequences=True)))
self.model.add(Conv1D(filters=200, kernel_size=3, padding='same', activation='relu'))
self.model.add(MaxPooling1D(pool_size=2))
self.model.add(Flatten())
self.model.add(Dropout(0.5))
self.model.add(Dense(1, activation='sigmoid'))
```

```
self.model.compile(loss='binary_crossentropy',
                    optimizer=rmsprop(lr=0.0001),
                    metrics=['accuracy'])
```

כאשר max_features נותר ללא שינוי

ההרצה של המודל מה-epoch האחרון הביאה את התוצאה הבאה:

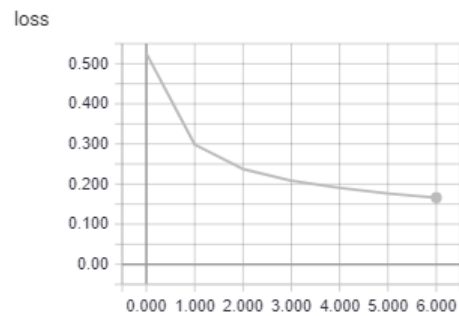
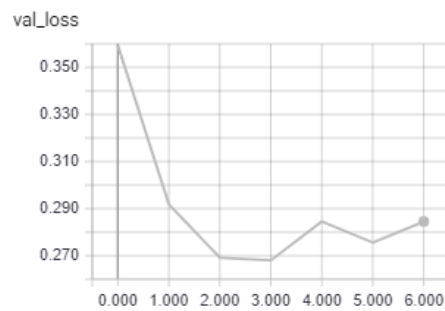
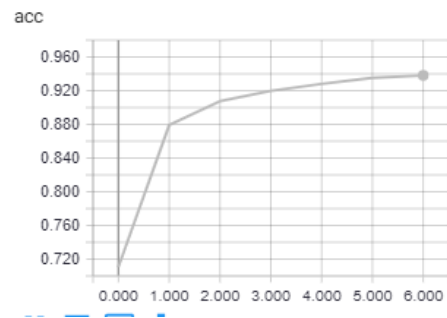
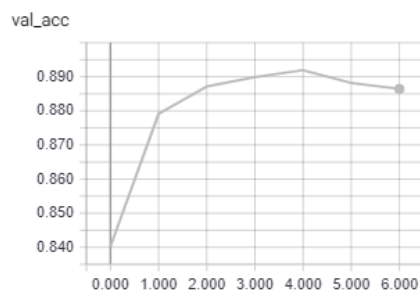
```
16500/16500 [=====] - 117s 7ms/step
Test accuracy: 89.1999999855503
```

בהרצה החשובה באמת שהיא של המודל אשר המחשב שמר (בשל val_loss הכי נמוך מכל ה-epoch-ים, הסבר מעמיק יותר ניתן למצוא בדף הסיכום) התקבלה התוצאה הבאה:

```
16500/16500 [=====] - 116s 7ms/step
Test accuracy: 89.41818181673685
```

ללא ספק, התוצאה הטובה ביותר מכל התוצאות שנצפו.

הגרפים:



תצורה זו הראתה את השיפור המשמעותי ביותר בכל הבעיות אשר דיברנו עליהן:

1. הקפיצות של המודל פחתו משמעותית .
2. ההבדל בין ה-loss ל-val_loss ובין ה-acc ל-val_acc איננו משמעותי, דבר אשר מצביע על ירידה ב-overfitting.

למרות התוצאה הטובה עדיין ישנן מספר בעיות במודל ועל כך ניתן לקרוא בדף הבא.

מחשבות על העתיד

לאחר בדיקת הנתונים על ה-test ולאחר הפעלת המודל על ביקורות חיצוניות לגמרי הגעתי למסקנות הבאות.

אומנם ישנה תוצאה חיובית אך ישנם כמה בעיות:

1. בעיית ה-Overfitting והקפיצות בגרף:
גם במודל הכי טוב שלנו עדיין מתקיים Overfitting קל וקפיצה של פונקציית ה-Loss:
 - הוספת מידע לאימון: ביקורות נוספות ממאגרים שונים ובגדלים שונים (Twitter, Amazon, Rotten Tomato's וכו') הביקורות לא דווקא חייבות להיות על סרטים. ובנוסף, הכנסת מידע נוסף שהוא לא דווקא ביקורות כמו למשל: עבור כל ביקורת ביצוע קידוד של עץ sentiment והכנסתו כפרמטר לכל רצף.
 - משחק עם גודל ה-learning rate והאופטימיזרים.
 - משחק עם ה-Dropout דבר אשר יכול להשפיע על מורכבות המודל ולהוריד מה-Overfitting.
 - חשוב לשים לב, שלא נפשט יתר על המידה את המודל בכדי שלא נגיע למצב Underfitting מצב שבו המודל מתקשה לזהות אפילו את המידע שהוא מתאמן עליו שלא לדבר על מידע לא מוכר.
2. בשלב כלשהו המודל שלנו מתחיל לאבד את הכיוון ולטפס בערך ה-Loss:
 - פתרון שכבר קיים הוא שמירה של המודל הטוב ביותר. למעשה, בכל epoch המחשב בודק אם היה שיפור ב-val_loss במידה וכן, הוא שומר את המודל.
 - פתרון נוסף שכבר קיים הוא עצירה מוקדמת, המחשב עוקב אחרי ה-val_loss ואם לא היה שיפור במשך 3 epoch-ים, המחשב עוצר את המודל.
 - פתרון ידני שניתן להכניס הוא ניסיון לכוון את כמות ה-epoch-ים לצד גודל ה-batch.
3. המודל שלנו נכשל לזהות מצבים של שלילה כפולה:
 - הכנסת מידע לאימון אשר כולל מצבים של שלילה כפולה וזאת בכדי שהמודל ילמד.
 - חישוב n-grams דבר זה אומנם קורה על ידי LSTM אך עם זאת אולי כדי לנסות בזמן העיבוד המקדים לבצע חישוב של 3-grams.
4. בשביל לתפוס קרבה יותר טובה בין מילים ניתן לבדוק את האפשרות של החלפת שכבת ה-embeddings באחד מהמודלים הבאים: word2vec, fastlearn, Glove.
5. המודל שלנו נכשל כאשר מדובר בביקורות ארוכות מדי שהפואנטה מגיע לקראת סוף הביקורת(אחרי השלב של הקיצוץ) או לחילופין כאשר מדובר בביקורת קצרה מדי.
 - כמובן שלא נוכל למצוא כאן פתרון אופטימלי, אך ניתן לנסות ולפתור זאת על ידי בדיקה של מגוון רחב יותר של אפשרויות עבור גודל ביקורת.

לדוגמא:

עבור הביקורת אשר בדף הבא המודל יחזה שהיא טובה, למרות שהיא קיבלה ציון 5. הסיבה לכך היא שהביקורת מחולקת כך שהחלק העליון(וזה שנקלט במודל) חיובי והתחתון שלילי.

Let's be honest! We were all, even marvel fan boys, waiting for this movie. The two most well known superheroes fighting... what could go wrong!? A lot.

The positives:

1. Batman & Alfred; Batfleck was amazing! Ben Affleck represented a batman that is damaged, changed his moral code(which if they play their cards right could be a center piece of solo batman films) and made him more bad-ass because of it.
2. The action: Amazing Action! It's thrilling, exciting and beautifully shot. That is what Zack does though and I expected nothing less.
3. Wonder woman: Supringly well done. Her as Diana prince was maybe a bit stiff sometimes but she can make the character her own along the way. I trust her that much. Her as wonder woman was amazing!! I got chills when she joined the fight!
4. Cinematography is general. It is a beautiful film to look at. Nicely shot and as I said the action is amazing. The CGI in certain places are a bit "meh" and one shot did take me out of the movie, but that's it.

Now the bad stuff why this movie went downhill:

1. the editing/pacing: Guys, this movie was a mess. The movie begins great with an introduction of batman and some stuff you saw in the trailer. Then we go to a scene... which tells us a story then it goes to another scene. You're invested in that particular and then it jump cuts to another scene. You are seeing 4 different story lines play out in 5 minute scenes after each other. As a general movie goer-ER you will lose track. As a DC-fan, as myself, I could understand everything but i needed to bring all my previous knowledge to actually understand. Really bad pacing
2. Superman: Not everyone will agree on this, but I think they handled superman horribly. He needs to be symbol of hope and yet he is never exactly that. There was certain montage, where it should shown that he was, but i never felt that he was. This representation was too dark. This was also more of a batman movie with superman playing a small part in that, yet batman's world is too dark for superman. If you want to combine their worlds, let Batman play a role in superman's world.
3. Lex Luther. I hated his representation of the power hungry and menacing villain. He played mark suckable and was a complete miscast. I don't care if he was the son of the "real" lex Luther. It didn't fit.
4. Doomsday: A complete waste of a great story arc. Why did they bring him in while there was so much going on..
5. The justice league set up: It felt rushed. The way they shoehorned all the other characters in was lazy writing.

6. the whole motivation of superman fighting batman. batman's motivation was clear and great. Superman's was horrible. They were one conversation away from understanding each other, but they had to fight of course.

7. Many many plot holes, but i cant go into those because i don't wanna spoil the whole movie.

verdict: If you love flashy action.. go watch it in Max. If you like great story.. wait till it releases on blue-ray.

6 out of 10 for a mediocre movie that should been an epic man. I was so disappointed. I see so many DC fans raging this movie was epic, but we're dealing with the phantom menace effect.

בכדי לחסוך מקום בדו"ח ולא ליצור דו"ח ארוך מדי לא כל הגרפים הוצגו ובנוסף, הורדו מספר בדיקות שלא תרמו כלל (כמו למשל החלפת LSTM ב-GRU או כמויות פילטרים שונות).
הרעיון שתמיד הנחה את בחירת המודל היא ניסיון להפטר מהבעיות המוסברות מעלה.