**דו"ח סיכום פרויקט – Deep Learning - Recurrent Neural Network**

מגישות: רז שיין 201054103 וליבנת גרשוני 301792792

בפרויקט זה התמודדנו עם חילוץ מידע מרצפים של טקסט, בעזרתו יצרנו מודל והצלחנו לחולל רצפים חדשים.

ה-Data Set איתו בחרנו לעבוד היה רשימה ארוכה של ציטוטים מפורסמים שונים. אורך Data Set הינו 2515 רשומות, כאשר כל רשומה מכילה ציטוט באורך שונה. את סט הנתונים הורדנו מהאינטרנט כקובץ .sql והמרנו אותו לפורמט csv.

בחרנו בסט הנתונים הזה כיוון שמשפטי חוכמה וציטוטים שם דמויות מובילות מהעבר כמו: בודהה, בנג'מין פרנקלין, המינגווי, שייקספיר ועוד. עניין אותנו לבחון עד כמה הרצפים יצאו קרובים למציאות ולמשפטים שסביר להניח היו נאמרים ע"י אותם אנשים.

האתגר העיקרי בעבודה עם סט הנתונים הנ"ל הוא שרצף המילים שנייצר לא יהיה באיכות טובה וכתוצאה מכך לא יהיה שימוש משמעותי במשפטי החוכמה שנייצר, למשפטים אלו אפשר שיהיה שימוש בנאומים עתידיים ועוגיות מזל.

תהליך ה-Data Preprocessing:

את הרשומות בפרויקט קראנו רשומה רשומה מקובץ ה-data.csv (נמצא תחת התיקייה data בפרויקט).

בחלק מן הרשומות היו תווים שלא מוכרים לשיטת ה-UTF8 ולכן היה עלינו להחליף תווים אלו בתווים מקבילים המוכרים לשיטה.

כל רשימה שהכילה ציטוט חולקה למספר משפטים במידה והציטוט הכיל מעל למשפט אחד (סה"כ עבדנו עם 3294 משפטים) וכל האותיות הוקטנו (case sensitive). לכל משפט הוספנו SENTENCE\_START ו- SENTENCE\_END על מנת ללמוד איזה מילים נוטות להופיע בתחילת משפט, ואיזה מילים נוטות להופיע בסיום.

ולבסוף כל משפט עבר tokenizing ובעצם יצרנו רשימה ארוכה של מילים – כ- 6411 מילים שונות.

יש משמעות למספר המופעים של כל מילה, לא נרצה ליצור רצף חדש עם מילה נדירה שהופיעה רק פעם אחת כי רוב הסיכויים שהיא לא תהיה רלוונטית – בעקר כיוון שלא ניתן ללמוד עליה יותר מידיי, לכן, היה צורך בלהגביל את גודל רשימת המילים בה נשתמש ליצירת הרצפים החדשים שלנו. לשם כך הגדרנו רשימה חדשה שנקראת vocabularyList , הרשימה מכילה את 1500 המילים השכיחות ביותר. את כל המילים שלא נכנסו ל- vocabularyList כיוון שהן נדירות יחסית לשאר, נחליף במילה UNKNOWN\_TOKEN וכמובן שאם בעתיד המילה UNKNOWN\_TOKEN תיבחר כחלק מרצף נדלג עליה.

מתוך הרשימה הזאת ניצור שני מילונים (data matrix) כפי שנלמד בשיעור: index\_to\_word ו- word\_to\_index שני המילונים האלה יעזרו לנו במשך ליצירת המשפטים

*לפני הרצה* : יש להוריד את הpackage nltk . ולכתבו בprogram . nltk.download() , רק בפעם הראשונה

בניית המודל:

חילקנו את כל המשפטים שלנו (כל ה-3294 ) ל- X\_train ול- Y\_train מהצורה הבאה:

X\_train: SENTENCE\_START It takes time to build a castle

Y\_train: It takes time to build a castle SENTENCE\_END

ולכן עבור המילה הראשונה ב-X המילה הראשונה ב-Y תהיה הבאה בתור.

מהחלוקה הזאת יצרנו מודל Simpel\_rnn - RNN אשר מבוסס על השיטות של Numpy. המודל מקבל כקלט את מספר הקודקודים בשכבה הנסתרת, אנחנו בחרנו ב-100.

לאחר שהמודל בנוי ניתן למדוד את הצלחתו במספר דרכים:

* יצירת משפטים ובדיקת תקינותם ע"י המשתמש
* בחינת ה-LOSS של המודל
* יצירת משפטים חדשים והשוואתם באמצעות דימון למשפטים בסט הנתונים

בחינת ה-LOSS של המודל:

על מנת לבחון את ה-LOSS של המודל עלינו תחילה לחשב את ה-LOSS של מודל רנדומלי על מנת שנוכל להשוות את התוצאות, התוצאה שיצאה לנו במודל הרנדומלי הייתה - 7.313220 כעת היה עלינו לחשב את ה-LOSS של המודל שלנו.

את הLOSS מחשבים בעזרת SGD אלגוריתם לאופטימיזציה של תוצאות, האלגוריתם משתמש בגרדיאנט אותו נייצר באמצעות אלגוריתם לפעפוע לאחור המיועד לRNN - *Backpropagation Through Time (BPTT)* . לבסוף נאמן את אלגוריתם ה-SGD על ה-train set שלנו ונראה שככל שהאלגוריתם רואה יותר דוגמאות הוא לומד יותר וה-LOSS שלנו קטן, כמו כן משתנה מאוד חשוב בחישוב הSGD הוא nepoch שבעצם מסמל את מספר הפעמים שהרשת לומדת , לפרמטר הזה בחרנו 100 איטרציות – הLOSS לאחר שהאלגוריתם ראה את כל המשפטים ב data set הוא 4.238144 .

יצירת משפטים:

ניצור משפט בעזרת המודל, תחילה נבחר את המילה הראשונה במשפט - SENTENCE\_START ונשתמש ב- forward propagation  שזה בעצם חיזוי של ההסתברות של כל מילה להיות המילה הבאה, מתוך זה נבחר את המילה בעלת ההסתברות הגבוהה ביותר. כך בעצם נשרשר מילים חדשות לרצף המשפט שלנו עד שהמילה SENTENCE\_END תצא עם ההסתברות הגבוהה ביותר. במידה ובמהלך יצירת המילים תיבחר המילה UNKNOWN\_TOKEN או מילים ללא משמעות כמו > \ וכו', נתעלם ונעבור להגרלת מילה חדשה, כמו כן טיפלנו בכינוי השייכות 's וקיצורים כמו 've.

לבסוף יתקבל משפט אותו נקרא ונחליט האם הוא תקין או לא מבחינה תחבירית והגיונית.

אלגוריתם לדמיון

את אלגוריתם הדמיון שלנו ביצענו באמצעות Minhash בשלוב עם Jaccard אשר מחזיר מספר בין 0 ל-1 לגבי הדמיון של משפט למשפט אחר.

**Simple Algorithm:**

|  |  |  |
| --- | --- | --- |
| Minhash & Jaccard Similarity | Original data sentence | New sentence |
| 0.2857142857142857 | He that falls in love with himself will have no rivals. | one can love falls who the it we knows a kindness it if the before if the departure too |
| 0.2857142857142857 | Everyone thinks of changing the world, but no one thinks of changing himself | it every know world if there in the and it he who and from itself everyone the all he |
| 0.25 | We don't love qualities, we love persons; sometimes by reason of their defects as well as of their qualities | if the too n't if technological how persons one a he before too good love if god in by with |
| 0.25 | everyone only at he he proof one a he before too good love if god in back he choose though | Confidence in the goodness of another is good proof of one's own goodness |
| 0.25 | Character is who you are when no one but GOD is looking. | one fate many fate god the it the and success he the and one all it the without |
| 0.2727272727272727 | Call it Nature, Fate, Fortune; all these are names of the one and selfsame God | one fate many fate god the it the and success he the and one all it the without |

לסיכום – האלגוריתם הפשוט

לאחר ההרצה של תוכנית ה-python שבנינו עם משתנים שונים כמו מס' הקדקודים בשכבה החבויה וגודל האוצר מילים שלנו, קיבלנו תוצאות שונות אך לא מצאנו תוצאה שהייתה בצורה מובהקת טובה מהאחרות, לאחר מחקר באינטרנט הגענו למסקנה כי אלגוריתם ה-RNN שנלמד בשיעור לא מתוחכם כמו לדוגמא אלגוריתם נוסף המבוסס עליו – LSTM, לכן פנינו ללימוד ובניית אלגוריתם ה-LSTM. תוצאת האלגוריתם הנ"ל היו בהחלט טובות יותר (פה גם היה שימוש בחבילה של theano )

בניית אלגוריתם חדש המבוסס על LSTM:

לאחר שבחנו את תוצאות המודל, והן לא היו כל כך טובות, חזרנו אחורה ומימשנו מודל נוסף LSTM\_rnn המבוסס על LSTM וגם הוא מקבל כקלט את מספר הקודקודים בשכבה הנסתרת, אנחנו בחרנו ב-100.

שמות הפונקציות במודל simple ו LSTMזהות, והמטרה של כל אחת מהן גם כן זהה, אך עדיין קיים שוני במימוש.

לאחר שהמודל בנוי ניתן למדוד את הצלחתו במספר דרכים:

* יצירת משפטים ובדיקת תקינותם ע"י המשתמש
* בחינת ה-LOSS של המודל
* יצירת משפטים חדשים והשוואתם באמצעות דימון למשפטים בסט הנתונים

בחינת ה-LOSS של המודל:

על מנת לבחון את ה-LOSS של המודל עלינו תחילה לחשב את ה-LOSS של מודל רנדומלי על מנת שנוכל להשוות את התוצאות, התוצאה שיצאה לנו במודל הרנדומלי הייתה - 7.313220 כעת היה עלינו לחשב את ה-LOSS של המודל שלנו.

את הLOSS מחשבים בעזרת SGD אלגוריתם לאופטימיזציה של תוצאות, נאמן את אלגוריתם ה-SGD על ה-train set שלנו ונראה שככל שהאלגוריתם רואה יותר דוגמאות הוא לומד יותר וה-LOSS שלנו קטן, כמו כן משתנה מאוד חשוב בחישוב הSGD הוא nepoch שבעצם מסמל את מספר הפעמים שהרשת לומדת , לפרמטר הזה בחרנו 100 איטרציות – הLOSS לאחר שהאלגוריתם ראה את כל המשפטים ב data set הוא 0.40379.

יצירת משפטים:

ניצור משפט בעזרת המודל, תחילה נבחר את המילה הראשונה במשפט - SENTENCE\_START ונשתמש ב- forward propagation  שזה בעצם חיזוי של ההסתברות של כל מילה להיות המילה הבאה, מתוך זה נבחר את המילה בעלת ההסתברות הגבוהה ביותר. כך בעצם נשרשר מילים חדשות לרצף המשפט שלנו עד שהמילה SENTENCE\_END תצא עם ההסתברות הגבוהה ביותר. במידה ובמהלך יצירת המילים תיבחר המילה UNKNOWN\_TOKEN או מילים ללא משמעות כמו > \ וכו', נתעלם ונעבור להגרלת מילה חדשה, כמו כן טיפלנו בכינוי השייכות 's וקיצורים כמו 've.

לבסוף יתקבל משפט אותו נקרא ונחליט האם הוא תקין או לא מבחינה תחבירית והגיונית.

אלגוריתם לדמיון

את אלגוריתם הדמיון שלנו ביצענו באמצעות Minhash בשלוב עם Jaccard אשר מחזיר מספר בין 0 ל-1 לגבי הדמיון של משפט למשפט אחר.

**LSTM Algorithm:**

|  |  |  |
| --- | --- | --- |
| Minhash & Jaccard Similarity | Original data sentence | New sentence |
| 0.7 | Good men have the fewest fears. He has but one great fear who fears to do wrong; he has a thousand who has overcome it | he has but one great fear who fears to do wrong he has a thousand who has overcome it |
| 0.625 | Animals feed; man eats. Only the man of intellect and judgment knows how to eat | only the man of intellect and judgment knows how to eat |
| 0.5555555555555556 | Animals feed; man eats. Only the man of intellect and judgment knows how to eat | realities the man of intellect and judgment knows how to eat |
| 0.4 | Time is the stuff life is made of | at some time in the life who is nature |
| 0.36363636363636365 | There is no such thing as a self-made man. We are made up of thousands of others. Everyone who has ever done a kind deed for us, or spoken one word of encouragement to us, has entered into the makeup of our character and our thoughts, as well as our success | who we has ever done a kind for us , or spoken all one word or has man |
| 0.3333333333333333 | A celebrity is one who is known to many persons he is glad he doesn't know | he does not know how many one that from one beautiful and one |

לסיכום – אלגוריתם LSTM

לאחר ההרצה של תוכנית ה-python שבנינו עם משתנים שונים כמו מס' הקדקודים בשכבה החבויה וגודל האוצר מילים שלנו, תוצאות האלגוריתם הנ"ל היו בהחלט טובות יותר (פה גם היה שימוש בחבילה של theano ), המשפטים נראו הגיוניים יותר, וגם הדמיון לסט הנתונים שלנו היה גדול יותר