פרוייקט גמר בהשראת המאמר Character-level Convolutional Networks for Text Classification

כותב: אדם יערי, אוניברסיטת בר-אילן

מנחה: ד"ר יואב גולדברג, אוניברסיטת בר-אילן

מבוא:

קיטלוג (קלסיפיקציה) של טקסט היא משימה קאלסית בתחום עיבוד שפה טבעית (*NLP*) במדעי המחשב, בה נעשה קיטלוג של טקסט, הניתן כקלט למערכת, לאחת מקבוצת הקטגוריות המוגדרות מראש. במשימה זו המחקר העכשיוי מתמקד בשאלות מגוונות וברמות מורכבות שונות, משינויים יחסית קטנים של מאפיינים (פיטצ'רים) במערכת בכדי למצוא את הכיול האופטימלי ללמידת המערכת, וכלה בבחירת ועיצוב מקטלגים שונים למשימה, המתבססות על תיאוריות שונות להבנת הכתוב. נכון להיום, כמעט כל הטכניקות לביצוע קיטלוג מתבססים על שיטות להבנת הטקסט המתבססות על מילים מוטבעות (*Embedded words*) ע"י טכניקות כגון *Word2Vec* ו- *Lookup Table*. בעבור הארכיטקטורות האלה ניתן לדמין את הבנת הכתוב באופן דומה לדרך בה אנחנו כבני אדם קוראים משפט ומסוגלים להגיד אם הטקסט מדבר על נושא כזה או אחר. בשיטות אלו סטטיסטיקות פשוטות, של רצף מילים מסודר כלשהו, (כדגומאת *n-gram*) לרוב מתפקדות הכי טוב [2].

הבעיה בשיטות אלה היא שהן דורשות ידע מקדים על השפה שאינו זול – מילון מוכן מראש, דרך התמודדות וקיבוץ של מילים עם משמעות דומה, הפרדה בין מילים זהות בעלות משמעות שונה וכודמה. דרישות אלה הופכות את המשימה של הבנת הכתוב למוכוונת לשפה ספציפית, כך שכאשר נרצה להשתמש בכלי ששימש להבנת טקסט בשפה א' עבור הבנה של שפה ב', נאלץ לעשות שינויים רבים ומשמעותיים בארכיטקטורת הכלי.

מצד שני, מחקרים רבים בתחום רשתות הנוירונים הראו של- *CNN) Convolutional Neural Networks*) יכולות גבוהות בחילוץ מידע מאותות פשוטים (*raw signals*) [3,4] בתחומים רבים ומגוונים כגון: ראייה ממוחשבת (*Computer vision*), הבנת דיבור (*Speech recognition*) ועוד. ברשתות קונבולוציה המידע נאסף בעזרת מעבר בין מספר שכבות קונבולציה, כאשר בכל שכבה יש מספר רב של פילטרים הסורקים את האות הניתן להם (האות הראשוני או הפלט של אחת משכבות הקונבולוציה הקודמות). הפילטרים הללו מורכבים ממשקולות המאומנות עם הזמן למציאת מאפיינים שונים, מורכבים יותר ואינפורמטיבים יותר מבשכבה הקודמת להם, כך שבסופו של דבר הם מסוגלים לרמז על שיוך לקטגוריה כזו או אחרת.

בפרוייקט זה התבססתי על המאמר "*Character-level Convolutional Networks for Text Classification*" והוא הראשון המתייחס למשפט כתוב כאות פשוט של אותיות (*Character level*), ומימש רשת קונבולציה חד-מימדית הסורקת את המשפט הנתון ומקטלגת אותו לאחת מהקטגוריות הנתונות לה מראש. כותבי המאמר ציינו כי למרות, שבמאמרם הם הדגימו את הצלחת רשת הקונבולוציה על משימות קיטלוג בלבד, הם מאמינים כי היכולות של רשתות *CNN* לטובת הבנת הטקסט כאות פשוט הן רבות ומגוונות יותר מהמוצג במחקר הנ"ל, בהינתן קלט אימון גדול מספיק.

בנוסף לכך, שימוש ב- *CNN* לטובת הבנת טקסט הינה אפשרות שנחקרה בעבר והוכחה כיעילה עבור שימוש ישיר על מילים מוטבעות [5,6,7], גם ללא ידע מקדים על מבנה סינטקטי או סמנטי בטקסט, ונמצא כמתחרה עבור שיטות הקלסיפיקציה המסורתיות.

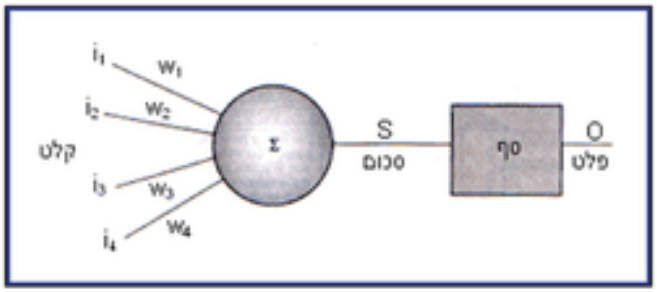
שיטת ה- *Text Understanding From Scratch* שמציע המאמר הנ"ל מאפשרת שיפור הן של גנריות והשימושיות של השיטה, והן הפחתה של מנגנונים מורכבים בדרך להבנת הכתוב באופן הבא:

1. יש שימוש באותיות ולא במילים – חוסך את השימוש המיותר בשיטות הטבעת המילים לווקטור, שהינן מסובכות להפקה בפני עצמן ומייצרות מרחב גדול מאוד של ווקטורים אפשריים (*High dimensionality*).
2. אין צורך בידע סינתקטי או סמנטי מקדים על השפה – הסקה לרמות גבוהות יותר של עיבוד מספיקה. מה שמפריך את ההנחה כי יש צורך שהנחות מובנות ומודלים של שפה נדרשים לצורך הבנת הטקסט, ומאפשר שימוש באותה שיטה עבור שפות רבות.

חומר רקע

בחלק זה של הדו"ח אתן מעט רקע והגדרות למושגים מעולם רשתות הנוירונים בהם אשתמש בהמשך בכדי לתאר את הנעשה בפרוייקט. אתחיל בהסבר קצר על מה היא למעשה למידה ברשת נוירונים וארחיב על מספר מושגים חשובים בשפה המקצועית, שבהם גם אשתמש בהמשך.

נוירון מלאכותי הוא למעשה גוף חישובי פשוט ביותר המבצע שתי פעולות: סכימת הקלט הנכנס אליו והפעלת פונקציית אקטיבציה (לרוב לא לינארית) על סכום זה. כל קלט בודד אותו סוכם הנוירון הוא מכפלה של ערך התא הקודם לו (בין אם קלט הרשת או פלט השכבה הקודמת לו) והמשקולת בין התא הקודם לנוירון הנוכחי. כך למשל, בדוגמא למטה, הקלט לנוריון הוא $\sum_j i_j * w_j$, עליו תפעל פונקציית האקטיבציה שתקבע, בסופו של דבר, את פלט הנוירון. ברשת נוירונים מדובר בכמות גדולה של משקולות העוברות בין תאים בצורות רבות ומגוונות, ופלט אותה רשת נקבע ע"פ הפלט של הנוירונים האחרונים והרחוקים ביותר מהקלט ברשת זו, לאחר פעפוע החישוב הנ"ל משכבת הקלט הראשונית ועד לשכבת הפלט בתהליך הנקרא *forward propagation*.



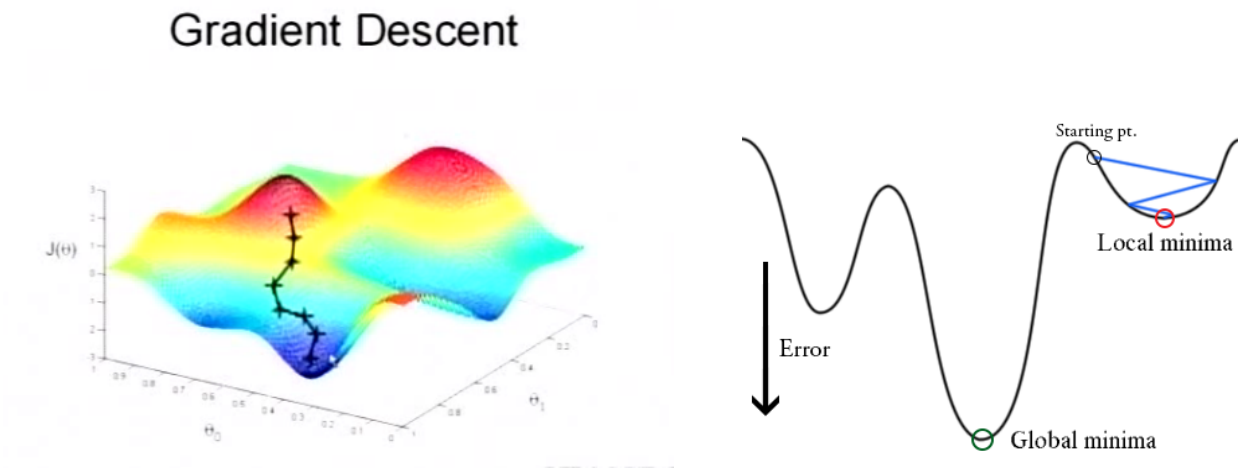
*איור 1: אילוסטרציה של נוירון מלאכותי יחיד.

תוצאת הרשת ניתנת להבנה ולאימון בצורות רבות ומגוונות, אך בכדי להסביר בצורה פשוטה המאפיינת גם את מבנה הרשת שתתואר בהמשך דו"ח זה, אתמקד ברשת שמטרתה הסופית היא קיטלוג. ברשת המקטלגת בין C קטגוריות יהיו C נוירונים בשכבת הפלט, כך שכל אחד מהם יאפיין קטגוריה מסוימת ונרצה שיפעל רק כאשר הקלט הנוכחי של הרשת שייך לאותה הקטגוריה. כלומר, נרצה שעבור דוגמא בה יש 4 קטגוריות, כאשר הקלט הנוכחי שייך לקטגוריה מספר 1, ווקטור הפלט יהיה [1,0,0,0], ואילו כאשר הקלט שייך לקטגוריה 3 הפלט יהיה [0,0,1,0]. במציאות אנחנו כמעט ולא רואים מצבים עד כדי כך נקיים ומוחלטים, ולכן מאפיינים את הקטגוריה אותה חזתה הרשת כזו בעלת הערך המקסימלי מבין C האפשרויות (מה שמאוד תואם את ההיגיון, מכיוון שטקסט המדבר על ספורט עולמי למשל, יכול להיות משוייך גם לקטגוריית עולם וגם לקטגורית ספורט במידות שונות). חשוב לציין כי ברשת נוירונים, המבנה של הרשת, פונקציית האקטיבציה והקלט אינם דברים הניתנים לשינויי, ולכן האלמנט היחיד עליו אנו משפיעים במהלך הלמידה הוא סט המשקולות.

אם כך, נוכל להגדיר למידה של רשת נוירונים כיצירת סט משקולות, שבהינתנו, הרשת יודעת לבצע מספר מקסימלי של סיווגים מוצלחים (או הצלחות בכל משימה אחרת אליה הרשת מותאמת), וסט משקולות א' טוב יותר מסט משקולות ב' עבור רשת כלשהי וקבוצת דוגמאות נתונה, אם סט א' מבצע יותר סיווגים מוצלחים של קבוצת הדוגמאות הניתנות כקלט לרשת.

הלמידה מתבצעת בתהליך אימון בו ניתנות לרשת מאותחלת במשקולות (רנדומליות לרוב) סט של דוגמאות עם הפלט הצפוי עבורן. כל אחת מהדוגמאות ניתנת כקלט לרשת, שמבצעת עליה חישוב *forward propagation* עד הגעה לפלט כלשהו. אותו פלט מושווה לפלט הצפוי עבור אותה דוגמה, ובעזרת ההפרש ביניהם מחושב ה- *loss*, ההפסד שיש לרשת מהאופטימום אליו ציפינו. בהתאם לגודל אותו אובדן (*loss*) נבצע שינויי של כל המשקולות ברשת, כך שבפעם הבאה שנראה את אותה דוגמה נקבל תוצאה טובה יותר וקרובה יותר לאופטימום.

באופן תיאורטי נוכל להסתכל על ערכי המשקולות כערכים בהיפר-מישור שבו קיים שילוב ערכים אופטימלי כלשהו, ושבו תוצאת הרשת היא הטובה ביותר, כשאנו נמצאים בתחילת הריצה בנקודה התחלתית רנדומלית כלשהי ושואפים להגיע למינימום הגלובלי כמו שמומחש באיור 2.



*איור 2: אילוסטרציה של משקולות עם מינימום גלובלי בהיפר מישור של 2 (2D) ו-3 מימד (3D).

אזי בכדי להגיע למינימום מהנקודה בה אנחנו עומדים נרצה לבחור את הכיוון עם ההתכנסות המהירה ביותר לאותו מינמום, כיוון שאותו נוכל לקבל (ע"פ היוריסטיקה שהדרך הכי תלולה היא גם הדרך הכי מהירה למטה) בעזרת נגזרת. אותה הנגזרת נקראת גרדיאנט (*Gradient*), שהוא כאמור הכיוון אליו נרצה ללכת את הצעד הבא גודלו של אותו צעד ייקבע ע"פ גודל האובדן, כך שעבור אובדן גדול נרצה לרדת צעד גדול ולהפך. לכן עדכון המשקולות יהיה מכפלה של הגרדיאנט, האובדן וקבוע הלמידה הנקבע מראש ועליו ארחיב בחלק המושגים.

לאחר חישוב השינוי שנרצה לעשות למשקולות הנכנסות לשכבת הפלט נעשה פעפוע לאחור של אותו שינויי לכל משקולות הרשת בעזרת חישוב הנגזרת החלקית שלהן ובהתאם "לגודל התרומה" שלהן לאובדן הסופי (משקולות בעלות ערך קטן יזוזו מעט ולהפך) בתהליך הנקרא *Backward Propagation*.

התיאור לעיל היה תיאור שטחי ומהיר לחישוב הנעשה ברשת נוירונים מלאכותית, והכיל את מרבית המושגים בהם אשתמש בדו"ח זה. אפרט עוד על 5 מושגים נוספים שלא נגעתי בהם עד כה, וחשוב להכיר:

- *Gradient descent*: תהליך הירידה במורד הגרדיאנט לכיוון המינימום הגלובלי או מינימום מקומי. לתהליך זה אנו למעשה קוראים "למידה", מכיוון שיש ירידה בערכי האובדן ושיפור בערכי החיזוי של הרשת.
- באטץ' (*Batch*): מספר דוגמאות שאנחנו מריצים אחת אחרי השניה, אך לא מפעפעים לאחור את ערכי האובדן שלהן עד סוף הבאטץ', בו נפעפע לאחור את ערך האובדן הממוצע. מאפשר למידה יותר חלקה ולא תזזיתית לפי טעות בודדת של כל דוגמה לחוד. כשאשתמש במושג באטץ' בדו"ח זה כוונתי היא למעשה למיני-באטץ' (באטצ'ים קטנים הנותנים את הפשרה בין למידה ע"פ דוגמה אחת ולמידה ע"פ כל הדוגמאות יחדיו).
- רגולריזציה (*Regularization*): מושג שמטרתו לאפשר יציאה מאותו מינימום מקומי שלא נרמה "להתקע" בו, לרוב ע"י הכנסת "רעש" לערכי שינוי המשקולות והוספה של רנדומליות מסוימת שיכולה לשנות מסלול לא יעיל.
- *התאמת יתר (Over fitting)*: מצב בו יש למידה מאוד ספציפית לסט האימון, הנותנת ערכי אובדן קטנים מאוד (וכביכול מוצלחים), שאינם מתבטאים בהצלחה בסט הבדיקה. קורה למשל, במצבים כמו סט אימון קטן מידי בהם הרשת מתמקצעת בסט הדגימה ולא בבעיה הכללית.
- קבוע למידה (*Learning rate*): קבוע הלמידה הוא קבוע שנוסיף למכפלת הגרדיאנט כדי לצמצם את צעד ה-*Gradient descent*, שלא לבצע צעדים גדולים מידי שיפספסו את המינימום המבוקש.

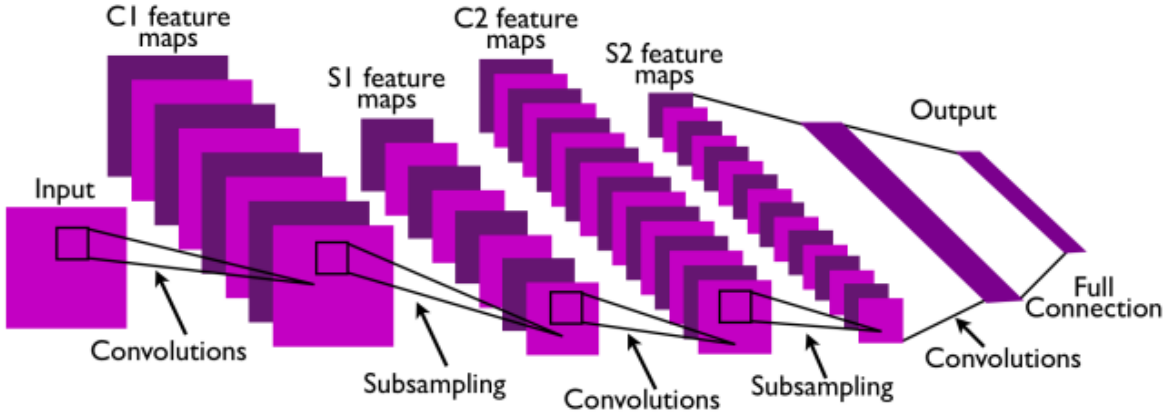
עיצוב רשת ה- CNN

בחלק זה אתאר בקצרה מבנה של רשת *CNN* סטנדרטי ולאחר מכן ארחיב יותר על פירוט רשת ה-*CNN* שתוארה במאמר המקור, ושאותה מימשתי בפרוייקט שלי.

רשת CNN סטנדרטית:

באופן כללי *Convolutional Neuron Networks* מתבססות על 3 סוגי שכבות עיקריות:

- *Convolutional Layer*: זו השכבה המקנה לרשת את שמה ומהווה את השכבה העיקרית ברשת *CNN* סטנדרטית. בכל שכבה מסוג זה ישנם מספר פילטרים/חלונות בגודל זהה אך בעלי משקולות שונות הנלמדות במהלך ה- *Back Propagation*. פלט השכבה מיוצר ע"י קונבלוציה של כל אחד מהפילטרים הנ"ל עם מטריצת הקלט של השכבה, ובכך מייצר פלט תלת-מימדי שמימדיו תלויים בגדלי הפילטר, הקלט ומספר הפילטרים[1]. פלט זה תוצר מועבר דרך פונקציית אקטיבציה, שהינה לרוב לא לינארית (כגון: *ReLU*, *tanh*, *sigmoid* וכו'), טרם העברת התוצאה כקלט לשכבה הבאה. במהלך האימון ניתן לראות כי כל אחד מהפילטרים לומד לזהות מאפיין תבניתי כלשהו, שרמת מורכבותו עולה עם עומק השכבה בה הפילטר נמצא.
- *Pooling/Subsampling Layer*: שכבה זו מהווה צמצם של פלט הקונבולוציה ע"מ למזער את כמות המידע[2] שיש צורך להעביר בין השכבות ולהוריד את זמן החישוב. מטרת שכבה זו היא לבחור את האיברים החשובים בכל חלון בו היא מסתכלת, למשל ברוב המצבים נחשיב את האיבר הגדול ביותר בחלון מסויים להיות המשמעותי ביותר בחלון, ולכן נשתמש ב- *Max Pooling*.
- *Dense/Fully-Connected* Layer: שכבה זו היא למעשה שכבת רשת נוירונים רגילה והפשוטה ביותר, בה כל נוירון בשכבה ה- i מחובר לכל נוירון בשכבה ה- $i+1$. שכבה אחת או מספר שכבות כאלה תמיד יהיו האחרונות ברשת *CNN* ובהן יבוצע מעין "סיכום" של תוצר הרשת, שבסופו של דבר גם ייתן את התוצאה הסופית, למשל בעזרת פונקציית *Softmax* עבור קלסיפיקציה.



*איור 3: תמונה להמחשה של מבנה רשת CNN בעלת שתי שכבות מכל סוג המצויין לעיל.

חשוב לציין כי יתרונות הרשת הן חיסכון בזיכרון (ביחס לרשת *Fully Connected* באותו הגודל), היות והמשקולות חוזרות על עצמן עבור כל איבר בקלט[3], ויש לה יכולות גבוהות באבחון מאפיינים מרחבים ספקטרלים כאשר קלט הרשת הוא יחידות בסיס קטנות ופשוטות, כגון פיקסלים בתמונה או פונמות בקלט אודיטורי, בדומה לאופן פעולת מערכת הראייה האנושית המזהה אובייקטים מורכבים מקלט של נקודות אור נפרדות.

---

1. בגרסה בה השתמשתי לא נעשה ריפוד אפסים בקצוות הקלט ולכן גודל תוצר הקונבולוציה המתקבל עבור קלט בגודל $l\times h$ ו- k פילטרים בגודל $d\times r$ הוא $(l-d+1)\times(h-r+1)\times k$.
2. עבור חלון בגודל 3×3 ממנו נבחר רק איבר 1 נוכל לצמצם את גודל המטריצה פי 9.
3. מספר המשקולות תלויי בגודל ומספר הפילטרים שקטן משמעותית מגודל הקלט, שמהווה את מספר המשקולות בשכבה *Fully Connected*.

Character-level CNN עבור קיטלוג טקסט[4]:

- **מרכיבי מפתח:**
  - הקונבולוציה במערכת זו הינה חד מימדית, כלומר שעבור כל פילטר ניתן רק מימד אחד (d) כגודל הפילטר ומספר פילטרים (k), ומשמעותו הוא שעבור מטריצת קלט, בגודל $l\times h$, נבצע קונבולוציה עם k פילטרים בגודל $d\times h$, שתייצר פלט דו-מימדי בגודל $(h-d+1)\times k$.
  - שיטת ה- *Pooling* ברשת זו היא *Max Pooling*, בה נעשה שימוש גם ב- *CNN* בראייה ממוחשבת.
  - פונקציית האקטיבציה היא *Threshold ReLU*[5] בכל שכבות הרשת (קונבולוציה ומלאות), כך שהסף המינימלי נקבע להיות 10^{-6}.
  - שיטת למידת המשקולות היא *Stochastic Gradient Decent* עם גודל מיני באטץ' של 128, מומנטום של 0.9, קצב למידה 0.01 וקבוע רגולריזציה של 10^{-6}.
  - המשקולות ההתחלתיות הן רנדומליות עם התפלגות גאוסיאנית ושונות של 0.05.

- **קוונטיזצית (*Quantization*) האותיות:**

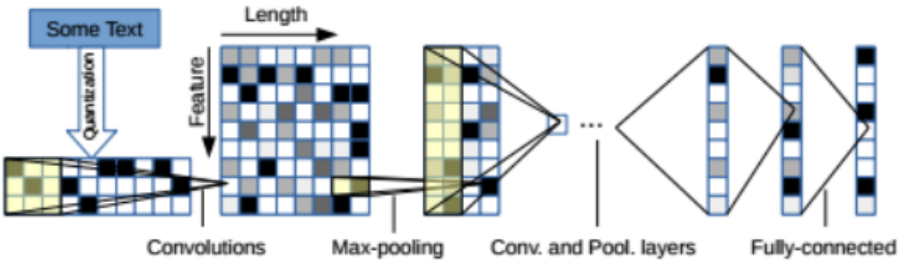
במודל זה ישנה למעשה קוואנטיזציה של אותיות ממאגר ידוע מראש לווקטורים בגודל קבוע, המחליף את מקומן של ווקטורי המילים המוטבעות. בהינתן שישנן m אותיות שונות המתקבלות כקלט לרשת, הקוואנטיזציה נעשית בשיטת *1-to-m*[6] ומייצרת m ווקטורים שונים. עבור כל אות שנקראת במשפט ואינה חלק מאותן m אותיות הקוואנטיזציה תחזיר ווקטור אפסים בגודל m. במאמר השתמשו ב-70 אותיות ותווים מהא"ב הלטיני (26 אותיות אנגלית, 10 ספרות ו-34 תווים הכוללים גם את תו הרווח). להלן 70 התווים (ללא הרווח):

abcdefghijklmnopqrstuvwxyz0123456789 -,;.!?:’’’/\|_@#$%ˆ&˜‘+-=<>()[]{}*

בנוסף, הם הגדירו אורך $l_0 = 1014$ שהגביל את אורך הקלט הנקרא, וכל אות מעבר לאותם 1014 תווים לא נקראה, כך שאורך הטקסט לא יהיה אחד מהגורמים המרמזים לקלסיפיקציה ולא יהיה צורך להתמודד עם אורך משתנה (לא היו טקסטים קצרים מ-1014 תווים בסט האימון והבדיקה) במעבר בין הקונבולוציה לשכבות המלאות.

- **עיצוב המודל:**

המודל הכיל 6 שכבות קונבולוציה ו-3 שכבות מלאות.



*איור 4: אילוסטרציה של המודל.

כאמור לעיל, גודל הקלט היה 1014×70, עקב מספר שיטת הקוואנטיזציה והגבלת אורך הקלט. כותבי המאמר ציינו כי מבדיקה אמפירית נראה כי 1014 תווים מסוגלים לתפוס את רוב עניין הטקסט הנתון. בנוסף לכך, בין 3 השכבות המלאות הוסיפו שתי שכבות השמטה (*Dropout*)[7] עם הסתברות 0.5 בכדי ליצור רגולריזציה.

---

4. במאמר המקורי פורטו שתי רשתות שונות, זהות מבחינת הארכיטקטורה אך שונות בגודלן, בפרוייקט זה מימשתי ועבדתי רק עם הרשת הקטנה מבין השתיים עקב אילוצי חומרה של מחסור במעבד גרפי *(GPU)* שייצרו זמני ריצה ארוכים במיוחד שלא התכנסו לזמני ההגשה.
5. $Threshold\ ReLU(x) = \max(threshold, x)$.
6. מקרה של שיטת *"One Hot Vector"* המייצרת m ווקטורים בגודל m כך שעבור האות ה- i האיבר ה- i בווקטור יהיה 1 וכל השאר אפסים.
7. שכבת השמטה משמיטה את הפלט שלה (מאפסת אותו) בהסתברות הנתונה לה (0.5 במקרה שלנו).

טבלה 1 מפרטת את קונפיגורציית חלק הקונבולוציה של הרשת:

מספר שכבה	מספר פילטרים	גודל פילטר	גודל חלון *Pooling*
1	256	7	3
2	256	7	3
3	256	3	-
4	256	3	-
5	256	3	-
6	256	3	3

בפילטר הקונבולוציה גודל הצעד היה 1, וה- *Pooling* היה ללא חפיפה.
טבלה 2 מפרטת את קונפיגורציית חלק השכבות המלאות של הרשת:

מספר שכבה	גודל פלט
7	1024
8	1024
9	מספר הקטגוריות

במודל זה ע"מ לחשב את גודל הקלט לשכבה 7 (השכבה המלאה הראשונה) ניתן לעשות חישוב בנוסחה הבאה: $l_6 = (l_0 - 96)/27$, כך ש- l_i הוא אורך הפלט של השכבה ה- i.

את הקוד ניתן למצוא בקישור הבא: https://github.com/AdamYaari/NLP_CNN_From_Scratch/

מאגר המידע לעבודה:

בפרויקט זה השתמשתי באחד ממאגרי המידע בו השתמשו במאמר המקורי ע"מ שאוכל להשוות את תוצאות הריצות שלי לתוצאות שהושגו ע"י צמד החוקרים המקורי. המאגר בו בחרתי להשתמש הוא - *AG's news corpus*[8] המכיל 496,835 מאמרי חדשות מקוטלגים לפי נושאים מיותר מ-2000 מקורות עיתונות. במאגר המידע שנלקח במאמר, ובו גם אני השתמשתי, נלקחו 4 הקטגוריות הגדולות ביותר, ומכל אחת נדגמו 30,000 מאמרים לסט האימון ו-1,900 לסט הבדיקה.

הטקסט שהועבר כקלט לרשת ה- *CNN* הוא שרשור של כותרת המאמר ותיאור הכתוב במאמר, שנקטע לאחר 1014 תווים מהסיבות המפורטות לעיל.

8. *http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html*

תהליך העבודה והמודל הסופי

בחלק זה אתאר תחילה את תהליך הלמידה לפני קבלת המשימה הספציפית אותה אני מגיש, ולאחר מכן אתמקד יותר במשימה שעשיתי, במכשולים ובמימוש הסופי שביצעתי.

עבודת ההכנה ומימוש הפרוייקט

ע"מ להבין את התכנים איתם אני עובד במהלך הפרוייקט, ובכדי להיות מסוגל לממש את הנדרש ממני, הייתי צריך ללמוד שני קורסים ייעודיים העוסקים ברשתות נוירונים. הראשון הוא הקורס הבסיסי של ד"ר גדי פנקס הנקרא "רשתות נוירונים" ומתעסק ברמות יותר בסיסיות של סוגי למידת מכונה בעזרת רשתות נוירונים, שהיה חלק מתוכנית הלימודים המתוכננת בתואר שלי, והשני הוא קורס מתקדם יותר של ד"ר יואב גולדברג, מנחה פרוייקט זה, שנקרא "עיבוד שפה טבעית בעזרת רשתות נוירונים עמוקות", קורס שלא היה חלק מהסילבוס של התואר שלי, ושביצעתי כחלק מהפרוייקט כדי להעמיק את הידע שלי בתחום ולהבין את המושגים איתם אתעסק לעומק. במסגרת שני הקורסים הנ"ל, ובנוסף למימוש במפורט בפרוייקט זה, ביצעתי מספר מימושים של רשתות נוירונים רבות המבצעות משימות שונות, מהרמה הפשטנית ביותר של שני נוירונים לינארים הפותרים את בעיית XOR ועד רשת Bi-LSTM המקטלגת טקסט לקטגוריות.

במהלך פרוייקט זה ביצעתי שני מימושים נפרדים לרשת הנוירונים המפורטת לעיל, בשתי שפות *Open Source* ייעודיות לעבודה עם רשתות נוירונים עמוקות – *Keras*[9] ו- *Pycnn*[10]. כאשר מטרת הפרוייקט הייתה לממש את מודל רשת ה- *Character-level CNN* ב-*Pycnn*, *ולנסות לשפר את התוצאות המוצגות במאמר עצמו*.

Keras הינה חבילה בעלת פונקציונאליות גבוהה יותר מבחינת כמות הפונקציות ונוחיות העבודה בה, ולכן מימשתי את הרשת בעזרתה, טרם מימוש ה-*Pycnn*. *Keras* איפשרה לי להגיע לתוצאות מהירות יחסית מבחינת מימוש (ומבחינת זמן ריצה), לוודא את ההבנה הבסיסית שלי לארכיטקטורת הרשת ולבצע מספר ניסויים מקדימים למציאת פיצ'רים ושינויים מבניים יעילים יותר. לעומת זאת, כמו כל חבילה בעלת פונקציות ייעודיות מרובות ונוחיות עבודה טובה, *Keras* מאפשרת פחות נגיעה בפרטים הקטנים של הרשת ופחות גמישות בכל מה שאינו במסגרת העבודה שלה. אזי, בכדי להגיע לרמת הבנה עמוקה יותר של המימוש, ולוודא כי אכן המנגנון פועל כמו שאנו מצפים ממנו, השקעתי את מרבית זמני במימוש הרשת גם ב- *Pycnn*.

במהלך העבודה, היות ומימוש ה- *Keras* הראה למידה מהירה ויעילה, האתגר המרכזי היה לגרום למימוש ה- *Pycnn* ללמוד באופן זהה, או במילים אחרות ליצור ב- *Pycnn* רשת שקולה לזו של ה- *Keras*. מה שהוסיף לפרוייקט מטרה נוספת הנובעת משתי המטרות הראשוניות, לבצע השוואה בין המתודות הקיימות ב- *Keras*, שבהן עשיתי שימוש, לבין המתודות שכתבתי ב- *Pycnn*.

מודל ה- *Pycnn* הסופי:

כאמור לעיל, מספר הפונקציות המוכנות מראש של- *Pycnn* קטן יחסית, ולא עונה על כל דרישות רשת ה- *CNN* אותה רציתי לממש. מצב שיוצר אתגרים לא פשוטים בהתחשב בעובדה שהחבילה עובדת עם טיפוס מסוג *Expression* כמחלקת הבסיס שלה, טיפוס יחסית דל בפונקציונאליות וקשה להרחבה (בשונה מעבודה עם טיפוסים פרימיטיבים למשל, כגון *Integer*, *String* וכו'). בחלק זה אפרט את האתגרים העיקריים איתם נאלצתי להתמודד, ואת דרך הפיתרון שאותה אני מציג בסופו שלדבר:

1. אחד האתגרים האחרונים שאיתם התמודדתי , הוא דווקא הראשון שאציג פה, מכיוון שהייתה לו השלכה גדולה על אופן פעולת הרשת הסופית.
   הבעיה: ב- *Pycnn* עבודה עם מטריצה כקלט היא בעייתית בשני מישורים, הראשון הוא שלא כמו בעבודה עם ווקטורים, במטריצה אין גישה לאיבר בודד כלל (כמו למשל באופן הבא *Matrix[i][j]*), ושנית צורת העבודה דרשה פעולות *reshape* ו- *transpose* רבות כדי למשל את פונקציות ה- *Pooling* והקונבולוציה, שהן פעולות כבדות מאוד בזיכרון ואיטיות בזמן הריצה בחבילה הנ"ל.

---

9. https://github.com/fchollet/keras
10. https://github.com/clab/cnn

הפתרון: במקום עבודה עם מטריצה בחרתי לעבוד רק עם ווקטורים, כך שבמקום מטריצה בגודל $m \times n$ עבדתי עם ווקטור בגודל $1 \times (m * n)$, שבו האיבר ה- $i * m + j$ *מתאים לאיבר ה- (i, j)* במטריצה המקורית. פתרון זה איפשר פונקציית קונבולוציה ופונקציית *Pooling* פשוטות ומהירות משמעותית, בנוסף לצמצום של צריכת הזיכרון לעשירית מהכמות הקודמת[11].

2. **הבעיה:** ל- *Pycnn* יש אפשרות לפונקציית *ReLU* סטנדרטית אך ללא אפשרות ה- *Threshold* (שהוכחה בניסויי מקדים ב- *Keras* כי היא הכרחית ללמידה), ובשל העבודה עם טיפוס ה- *Expression* אין אפשרות לבדוק את הערך המספרי של תוצאה במהלך אמצע הריצה בזמן עבודה סביר.
הפתרון: מכיוון שערך ה- *Threshold* מאוד קטן (10^{-6}) השתמשתי בפונקציית ה- *ReLU* הקיימת ולכל התוצרים הוספתי את ערך ה- *Threshold*[12].

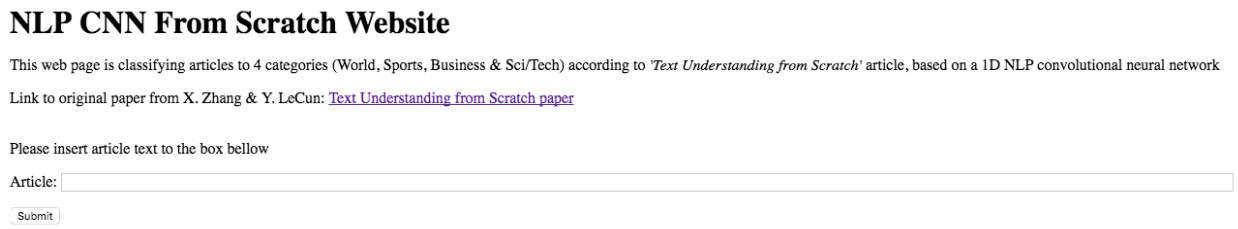
3. **הבעיה:** ל- *Pycnn* אין פונקציית *Max Pooling* שבעזרתה נעשה צמצום המימדים ב- 3 מתוך 6 שכבות הקונבולוציה, בנסוף לבעיית הערך המספרי של ה- *Expression הנ"ל*.
הפתרון: ל- *Pycnn כן יש פונקציית KMax Pooling* הבוחרת את *k* האיברים הגבוהים ביותר מהווקטור הנתון לה, לכן השתמשתי בה באופן הבא: לאחר כל *d* צעדים של הקונבולוציה, כאשר *d* הוא גודל חלון ה- *Pooling* (במקרה שלנו 3 בכל השכבות), שרשרתי אותם לווקטור *v* והעברתי אותו כקלט לפונקציה *KMax Pooling עם k = 1*. מה שבסופו של דבר נתן *Max Pooling לכל d* אברים.

4. **הבעיה:** ל- *Pycnn* אין פונקציית קונבולוציה כלל (לא חד-מימדית ולא דו-מימדית).
הפתרון: עבור גודל פילטר - *fl*, מספר פילטרים בשכבה - *fn* וגודל מסגרת הקלט (גובה המטריצה) - *h*, ייצגתי כל פילטר כווקטור באורך $h * fl$ וקיבצתי אותם ביחד למטריצה מסדר $(fl * h) \times fn$, וביצעתי כפל מטריצה בווקטור עבור כל מיקום *i* בווקטור הקלט של השכבה, כך ש- *i* גדל בקפיצות של *h*. כל הכפלה כזו של המטריצה הנ"ל בווקטור בגודל $1 \times (fl * h)$ יצרה ווקטור חדש בגודל *fn*, אותו שירשרתי עם שאר תוצרי הקונבולוציה ליצירת ווקטור הפלט של השכבה.

<u>ממשק ה- *Web* לשימוש בתוצר הלמידה:</u>

בנוסף לאפשרות הלמידה של הרשת יצרתי ממשק של דף *Web*, המאפשר טעינה של משקולות המודל לאחר תהליך הלמידה, וביצוע קיטלוג למשפט חדש, הניתן בדף כקלט, לאחת מהקטגוריות של מאגר המידע עליו נעשה האימון (כרגע מותאם ל- *AG's news corpus* אך בשינויים קלים בלבד יכול להיות מותאם לכל מאגר מידע אחר).

קוד הדף מבוסס על שפת *Bottle* המאפשרת ממשק *Web* נוח לשפת *Python*, שהינה גם הבסיס לשפת *Pycnn*, כך שבעת קבלת הקלט מהדף האינטרנטי, בהינתן משקולות המודל המאומנות, כל שנותר הוא לעשות קוואנטיזציה פשוטה של הקלט, להזין אותו ל- *Forward* Propagation ברשת ולהחזיר את תשובתה.



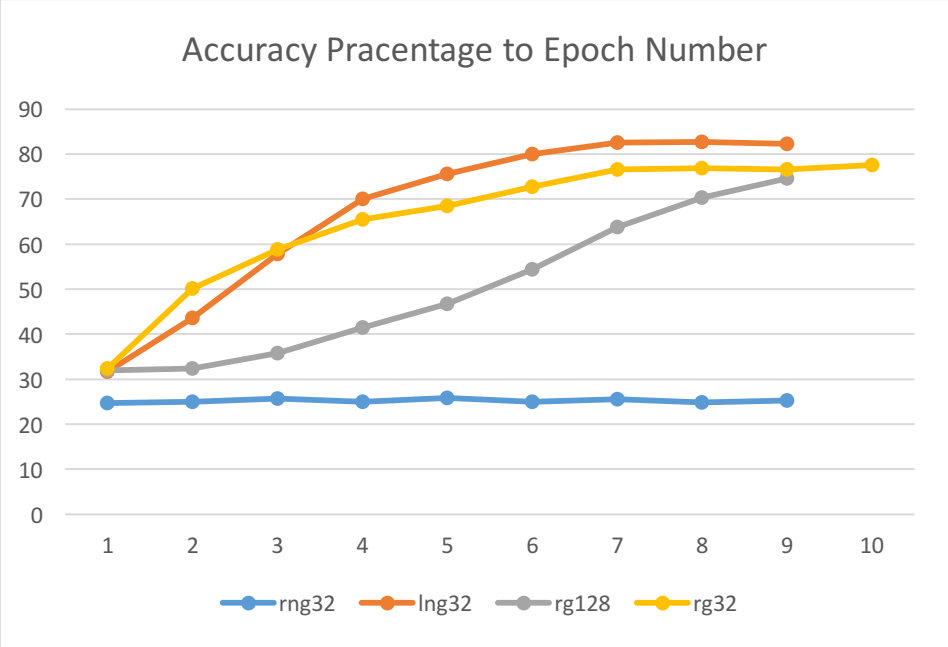
*איור 5: תצוגת הדף לפני הזנת קלט.

11. במימוש המטריצה מספר הדוגמאות המקסימלי שניתן היה להריץ יחדיו ב-*Batch* אחד היה 3 (עבור 2 GB של זיכרון לריצה), לעומת 35 דוגמאות ב- *Batch* במימוש עם ווקטור על אותה כמות זיכרון.
12. כלומר החלפתי את הפונקציה $\max(threshold, x)$ ב- $\max(threshold, x + threshold)$.

תוצאות

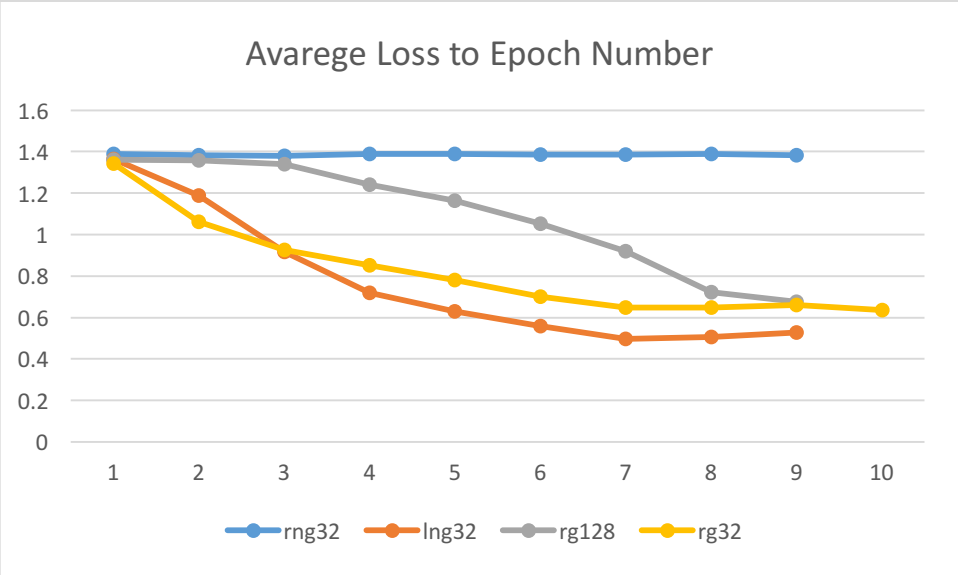
תחילה אציין שעמדתי במטרת הפרוייקט הראשית, שהייתה חיקויי מוצלח ועובד של המודל המוצג במאמר *Character-level Convolutional Networks for Text Classification [1]* בעזרת ספריית *Pycnn*, כך שנוכל לדעת האם המוצג במאמר אינו מטעה או חסר בפרטים. כאשר התוצאה הסופית המוצגת במאמר (עבור הרשת המקבילה לזו שמימשתי) היא 84.35%, ואילו התוצאה הסופית שלי היא 82.32%, חשוב לציין כי התוצאה אליה הגיעו צוות החוקרים ברשת המקורית התקבלה לאחר 30 סבבי למידה (*Epochs*)[13], ואילו התוצאה הסופית ברשת שלי התקבלה לאחר 9 סבבים בלבד, כנראה בשל מספר שינויים שעשיתי ואותם אפרט בהמשך. צמצמתי את מספר הסבבים אותם בדקתי משתי סיבות: הראשונה, היא איטיות הריצה (קצב ממוצע של כ- *Epoch* 1 ליום) הנובעת מהמחסור ב- *GPU* במחשבים עליהם עבדתי ומהמחסור באופטימיזציות של מהירות הריצה ב- *Pycnn*, והשניה היא התכנסות התוצאה לערכים יחסית קבועים, כך שלא היה נראה שעוד 20 סבבים ישנו את התוצאה עוד הרבה (ניתן לראות בתרשימים 1 ו- 2 בהמשך).

הניסויים העיקריים שביצעתי וייצרו למידה היו: שינויי גודל ה- *Batch*, ביטול אתחול המשקולות הגאוסיאני ושינויי פונקציית האקטיבציה בשכבות המלאות (ה-3 האחרונות) ללינארית (*במקום Threshold ReLU*). את שינויים אלה ביצעתי במספר שילובים אחד עם השני ובחנתי בעיקר אל מול קצב ההתכנסות ופחות אל מול התוצאה הסופית של סט הבדיקה בסוף הלמידה, עקב אותו שיקול זמן המצוין לעיל. חלק מהשינויים יצרו מצב בו לא הייתה למידה כלל, אך בשילוב שונה דווקא יצרו התכנסות מהירה וטובה יותר, כמו למשל ביטול אתחול המשקולות הגאוסיאני עם פונקציית אקטיבציה של *Threshold ReLU* על כל שכבות המערכת (קו כחול בתרשימים 1 ו- 2) לא התכנסה ללמידה כלל, אך בשילוב שינויי פונקציית האקטיבציה בשכבות המלאות ללינארית (קו אדום בתרשימים 1 ו- 2) יצרה את ההתכנסות המהירה והטובה ביותר בערך הסופי.



*תרשים 1: גרף של דיוק תחזית הרשת (באחוזים) על סט הבדיקה אל מול מספר ה- *Epoch*. בסוף כל *Epoch* נעשתה בדיקת קטגוריה חזויה מול צפויה, והערך המוצג בתרשים הוא מספר הצלחות החיזוי חלקי מספר הדוגמאות הכולל באחוזים, כפונקציה של מספר *Epoch*. הקווים המוצגים בתרשים מייצגים 4 ריצות שונות של הרשת עם שינויי הפרמטרים הבאים:

- rng32 (כחול) – *Threshold ReLU* גם על השכבות המלאות, ללא אתחול משקולות גאוסיאני ובאטץ' בגודל 32.
- lng32 (אדום) – שכבות מלאות עם פונקציית אקטיבציה לינארית, ללא אתחול משקולות גאוסיאני ובאטץ' בגודל 32.
- rg128 (אפור) – *Threshold ReLU* גם על השכבות המלאות, אתחול משקולות גאוסיאני בהתפלגות 0.05 ובאטץ' בגודל 128.
- rg32 (צהוב) – *Threshold ReLU* גם על השכבות המלאות, אתחול משקולות גאוסיאני בהתפלגות 0.05 ובאטץ'בגודל 32.



*תרשים 2: גרף של אובדן הרשת הממוצע על סט הבדיקה אל מול מספר ה- *Epoch*. בסוף כל *Epoch* במהלך בדיקת קטגוריה חזויה מול צפויה, חושב גם האובדן הממוצע על פני כל דגימות סט הבדיקה, ואלו התוצאות המוצגות בתרשים. הקווים המוצגים תואמים בשם ובצבע את אלו המתוארים בהרחבה בתרשים 1.

בהשוואה בין הגרפים, ניתן לראות באופן לא מפתיע כי קיימת התאמה בין ערכי האובדן של הרשת על סט הבדיקה לבין אחוזי ההצלחה שלה בסוף כל *Epoch*. בנוסף, בהשוואה בין ערכי הניסויים השונים, ניתן לראות כי ההתכנסות של באטץ' בגודל 128 איטית מזו של באטץ' בגודל 32 (בהשוואה בין הקו האפור והצהוב בהם כל שאר הפרמטרים זהים), וכי כאשר גודל הבאטץ' זהה השילוב של אקטיבציה לינארית לשכבות המלאות וללא אתחול גאוסיאני מביא לתוצאה טובה יותר מאשר אתחול גאוסיאני ואקטיבצית *Threshold ReLU* (בהשוואה בין הקו האדום והצהוב בהם גודל הבאטץ' זהה).

מבין כל השינויים הנ"ל השילוב המוצלח ביותר שבדקתי היה גודל *Batch* של 32 (לעומת 128 במקור), ללא אתחול משקולות גאוסיאניות ועם אקטיבציה לינארית בשכבות המלאות, והתוצאה הסופית של רשת זו היא זו שהוצגה קודם לכן. תוצאה דומה התקבלה עבור שינויים אלה במימוש בשתי החבילות *Keras* ו- *Pycnn*. למרות שע"פ הגרפים המתקבלים מהמדידות נראה כי כאשר הרשת רצה עם באטץ' בגודל 128 הלמידה איטית יותר, משמע ש- 10 אפוקים לא בהכרח מספיקים לה כדי להגיע לערך סופי, ויכול להיות שהייתה מגיעה לערכי דיוק גבוהים יותר בסופו של דבר.

השאלה האחרונה שהתעסקתי בה הייתה – האם מתודת הקונבולוציה הנתונה ב- *Keras* וזו שכתבתי ב- *Pycnn* שקולות? שאלה שהטרידה אותי רבות בשלב בו עוד לא הצלחתי להגיע להתכנסות הפרמטרים למצב של למידת הטקסט. לשאלה זו לא אציג תוצאות גרפיות או מספריות, מכיוון שהן חסרות עניין, ורק אציין שתחליף הקונבולוציה שהצגתי אכן זהה לפונקציית ה- *Keras* *Convolution1D* המשתמשת מאחורי הקלעים בפונקציית *Convolution2D* עם חלון בגדול הקרנל החד ממדי וגודל מסגרת הקלט (גובה המטריצה).

13. במאמר הם פירטו כי ביצעו 30 סבבים, כך שלאחר כל 10 סבבים חילקו את קבוע הלמידה ב- 2.

דיון

בנוסף ללמידה האישית שלי, תוצאות פרוייקט זה חיזקו את ממצאם של כותבי המאמר' הטוענים שניתן לבצע קיטלוג של טקסט בעזרת רמת ה"אות" בלבד ואין צורך במאמץ הנוסף הנדרש בכדי לקרוא את הטקסט ברמת ה"מילה". מה שנראה כמתאפשר במיוחד ברשת *CNN* המתמחה בחילוץ תבניות מורכבות מאות פשוט (*Raw Signal*), ויכול להיות שאף מאפשרת אבחון דקויות שאנו מאבדים כאשר נקרא את הטקסט ברמת המילה. כוונתי היא שבשכבות הגבוהות של רשת *CNN* יכול להיות שנוצרו פילטרים ה"מחפשים" מילים ספציפיות כתבנית ספקטרלית של אותיות, באותו האופן שקו ישר בזווית כלשהי היא תבנית ספקטרלית של פיקסלים בתמונה (כשאת אותו קו ישר פילטרים לומדים לחפש בסריקה של התמונה ומאפשרים ראייה ממוחשבת ברמה גבוהה), מה שלמעשה נותן לנו פונקציונאליות דומה לזו של הזנת מילים בשיטות כמו *Word2Vec* או *n-gram* (רק ללא הצורך בעיבוד מקדים). בנוסף לאפשרויות שונות ומשונות, שלא נוכל לקבל כאשר נסתכל רק על מילה כמבנה שלם, כשלדוגמא אולי הסיומת *tion_* מספר מסויים של פעמים בתוך טקסט מרמז שהוא שייך לטקסט המדבר על עסקים, או התחילית *re_* מרמזת על טקסט המדבר על נושאים טכנולוגים, שהן שתיהן דקויות שלא היינו רואים אם המילה הייתה מאופיינת כווקטור שמנסה לתפוס את משמעותה הסמנטית של המילה ולא כרצף אותיות שיכול להכיל גם משמעות סינטקטית כלשהי.

לעומת זאת, חלק ממסקנות הפרוייקט מנוגדות לכתוב במאמר, הדוגמה המובהקת ביותר לכך היא שכותבי המאמר טוענים כי שיטה זו דורשת מספר רב מאוד של דוגמאות בכדי לייצר למידה אפקטיבית, וכי 30,000 דוגמאות מכל קטגוריה המוצגות במאגר הנ"ל, הן כמות לא מספקת (במאמר הראו מצב של *Over Fitting* על אותו סט אימון ובדיקה בו אני השתמשתי). כאשר בתוצאות פרוייקט זה ניתן לראות איך בשינויים לא גדולים במיוחד ניתן לשפר את קצב ההתכנסות ואת איכות הלמידה ללא תוספת של דוגמאות נוספות. למרות שעדיין קשה לאמוד באופן מוחלט האם היה אפשרי בסופו של דבר להגיע לדיוק גבוה יותר מזה שהוצג במאמר, מכיוון שמדד זה לא נבחן לעומק מספיק במסגרת פרוייקט זה. במדידות שהוצגו בדו"ח זה הראיתי כי ניתן להגיע לאותן התוצאות בפחות עבודה, אך תהליך המדידה הופסק טרם הגעה לשיפור סופי או התייצבות חדש משמעית של התוצאות.

מסקנה נוספת, שלא נבעה מפרוייקט זה, אך אני רואה בה ערך רב לגבי יעילות השיטה, היא שהמודל איננו מוגבל רק לשפה אחת כמו מודלים המתבססים על רמת המילה, המתבססים על הקשר בין המילים השונות ומבנה השפה. החוקרים כותבי המאמר הראו שבעזרת תהליך פשוט של לטיניזציה (המרת מילה לאותיות לטיניות ע"פ הצליל שלה כך: "מילה" => *"mila"*), ניתן לבצע למידה של טקסטים בכל שפה ללא שינויי כלל לרשת הקיימת.

לסיכומו של דבר, עיבוד שפה טבעית ברמת האות ובעזרת רשתות נוירונים עמוקות הינו מעניין ביותר ויכול לספק פתרונות למכשולים רבים בעולם ההבנה הממוחשבת של טקסט. יהיה מעניין לבדוק את החלפת שכבות הקונבולוציה בשכבות *Bi-LSTM*, אך להישאר ברמת עיבוד בסיסית של אותיות ולא מילים, בנוסף לאפשרויות שיפור נוספות של רשת ה- *CNN*.

ביבליוגרפיה

1. *X. Zhang, Y. LeCun. Text Understanding from Scratch. 4 April 2016.*
2. *T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In Proceedings of the 10th European Conference on Machine Learning, pages 137–142. Springer-Verlag, 1998.*
3. *Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, November 1998.*
4. *J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal, 2014.*
5. *C. dos Santos and M. Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 69–78, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics*
6. *R. Johnson and T. Zhang. Effective use of word order for text categorization with convolutional neural networks. CoRR, abs/1412.1058, 2014.*
7. *Y. Kim. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.*