בס"ד
אסנת אקרמן 315747204 ackermo1
שירה יאיר 315389759 Yairshi

Assignment 4 – implementing an SNLI paper

במשימה התבקשנו לממש מודל שמשימתו היא להכריע עבור 2 משפטים- premise, hypothesys, האם משפט אחד נובע מהשני (entailment), האם המשפטים סותרים (contradiction), או האם אין קשר ביניהם (neutral).

עבור משימה זו נכתבו מאמרים שונים שמתארים מודלים שונים שהשתתפו בתחרות של סטנפורד על הדאטה שהם ייצרו – SNLI corpus.

בתחילה בחרנו מאמר בשם "*600D (300+300) Deep Gated Attn. BiLSTM* encoders" שנוקט בשיטה של להפוך כל משפט לווקטור בגודל קבוע ומשם לעבור לסיווג. מכיוון ששיטה זו משתמשת ברשתות עמוקות עם הרבה פרמטרים לאימון, הבנו שתהליך האימון יהיה ארוך מדיי וכבד מדיי ביחס לזמן ולמשאבים שיש לנו. לכן החלטנו לבחור במאמר אחר: *"A Decomposable Attention Model for Natural Language Inference"*, *vanilla approach*.

הרעיון המרכזי במאמר הוא להתייחס למבנה המשפטים עם תשומת לב לנושא ולפועל, ולמעשה לבצע soft alignment בין hypothesis לpermise באמצעות attention.

<u>תיאור המודל:</u>

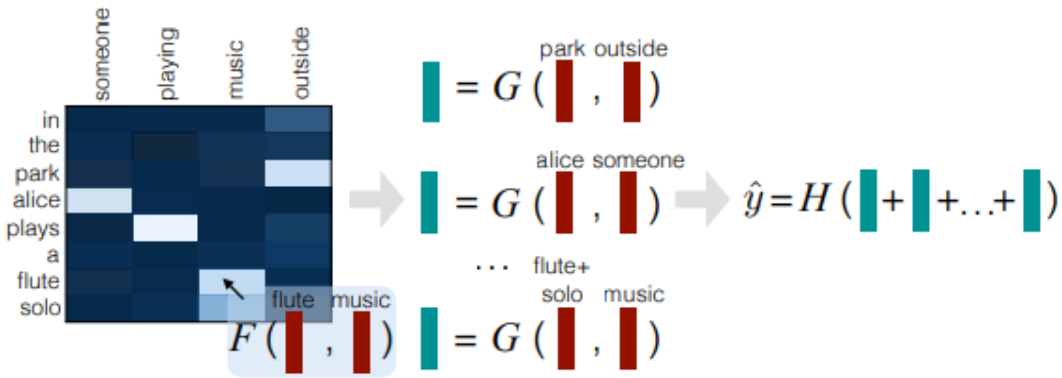
כדי לממש את המאמר בצורה טובה ביותר פעלנו כך:

1. **טעינת הדאטה- torchtext:**
כדי לטעון את הדאטה השתמשנו בספריית torchtext שטוענת אוטומטית את כל הדאטה מתוך הקבצים של train, dev, test אל טנזורים בפייתון שאפשר להשתמש בהם בנוחות. באמצעות הדגל Pre_proccesing הוספנו לכל משפט NULL בתחילתו כפי שהתבקש במאמר.
2. על הדאטה שטענו בנינו bucket_iterator שממיין את הדוגמאות לפי אורכי המשפטים מהקצר אל הארוך כך שנדרש פחות padding בכל batch ובעקבות כך זמן העיבוד מתקצר משמעותית.
3. על בסיס הקבצים שנטענו בנינו את אוצר המילים עבור טבלת הembedding וכל מילה קיבלה אינדקס. (הסבר מפורט על שיטת הembedding בהמשך)
4. עבור כל דוגמא בdata_set נבצע את השלבים הבאים:

- **שלב הENCODE:** כל מילה בכל משפט (hypothesis, premise) קיבלה את הייצוג שלה לפי טבלת הembedding.את הוקטורים מטבלת הembedding הגדרנו להיות במימד 300 והעברנו אותם תחת שכבה לינארית כדי להקטין את גודלם ל200.
- **שלב הATTEND:** מחשבים עבור כל מילה במשפט הראשון את המשקלים אל מול כל המילים במשפט השני (באמצעות מעבר ברשת נוירונים בעלת 2 שכבות, פונקציית אקטיבציה relu). לאחר מכן אנו מבצעים כפל בין המשקלים למשפטים המקוריים ויוצרים את שני הוקטורים alpha וbeta ושולחים לשלב הבא.
למעשה מה שהתבצע הוא soft_aligment בין שני המשפטים הנתונים.
- **שלב הCOMPARE:** בהינתן וקטורי alpha וbeta נשרשר אותם עם המשפטים המקוריים בהתאמה ואת התוצאה מעבירים בעוד שכבת נוירונים.
- שלב הAGGREGATE: בהינתן וקטורי v2,v1 מהשלב הקודם נסכום כל אחד מהוקטורים ונשרשר אותם יחד, את השירשור נעביר ברשת נוירונים.
- את התוצאה שקיבלנו מפונקציית הaggregate נעביר בשכבה לינארית נוספת שהoutput שלה הוא וקטור בגודל מספר הlabels שלנו.

אסנת אקרמן 315747204 ackermo1
שירה יאיר 315389759 Yairshi

- הכניסה שקיבלה את הציון הגבוה ביותר נבחרה להיות הפרדיקציה.



הדמיה של התהליך עבור כל השלבים שלו המוגדרים לעיל (מהמאמר)

<u>הסיבה שבחרנו במאמר:</u> כמות הפרמטרים קטנה יחסית והמשאבים שלנו מאפשרים את האימון הנדרש. בנוסף, ההתייחסות למבנה המשפט ולא רק למשמעות המילים נתנה פרספקטיבה חדשה על איך להתמודד עם המשימה, ועניין אותנו לשחזר את ההצלחה של מאמר זה.

<u>הסבר על שכבת הEMBEDDING:</u>

לאחר טעינת הדאטה יצרנו את אוצר המילים עבור טבלת הembedding באמצעות אותה ספרייה (torchtext). אל אוצר המילים הוספנו וקטורי Pre_embedding של glove . עבור כל מילה באוצר המילים שנבנה ניתן אינדקס בהתאם לטבלה שנטענה מראש. Padding קיבל את אינדקס 1, ומילים לא ידועות קיבלו אינדקס 0.
בתחילה טענו את הווקטורים מקובץ glove.6B.300D . קובץ זה מתייחס למילים בlower_case ולכן כשטענו את המשפטים הפכנו את כולם לlower_case כדי להתאים לאוצר המילים.
לאחר בדיקות באינטרנט גילינו שהקובץ glove.840B.300D יותר רחב וגם נותן התייחסות שונה לlower ולupper מה שמאפשר למידה מדויקת יותר למשפטים שנצפים בדוגמאות. שינוי זה אכן שיפר משמעותית את אחוזי הדיוק על הdev_set.
את הווקטורים שנטענו נרמלנו שוב (max_norm=1, נרמול l2) כפי שנתבקשנו במאמר. בנוסף, טבלת הembedding לא נלמדה במהלך האימון והווקטורים נשארו כמו שהם.

<u>OOV:</u>

לפי הוראות המאמר, עבור כל מילה שלא נמצאת בטבלת word_embedding (וקיבלה אינדקס 0) הוגרל וקטור אקראי מבין 100 וקטורים מאותחלים נורמלית בגודל 300 ששורשרו לטבלת הword_embedding המצויה.

<u>נקודות בתהליך שיחזור ביצועי המאמר:</u>

1. Optimizer – על פי המאמר בחרנו את AdaGrad עם LR=0.05, וinitial_accumulator_value=0.1. אחד מניסיונות השיפור שלנו כלל להגדיר את Adam הפופולרי כoptimizer עם אותו קצב למידה, אך לא קיבלנו שיפור ונשארנו עם מה שהוצע במאמר.
2. כל הhidden_layers בכל אחת מהרשתות הן בגודל 200 חוץ מהשכבה האחרונה, שכבת הoutput.
3. לכל אחת מהרשתות ביצענו רגולריזציה של dropout בהסתברות של 0.2 , הdropout בוצע לפני כל הפעלת פונקציית האקטיבציה-relu, אך לא עבור השכבה הלינארית האחרונה.

בס"ד
אסנת אקרמן 315747204 ackermo1
שירה יאיר 315389759 Yairshi

4. את כל הפרמטרים של השכבות הלינאריות נתבקשנו לאתחל באיתחול גאוסייני (std=0.01,mean=0), כשעשינו זאת ידנית האחוזים ירדו משמעותית וקצב הלמידה היה קטן,. דווקא האתחול הדיפולטי של Pytorch השיג תוצאות טובות יותר ולכן נשארנו איתו (init.uniform_).
5. Batch- במאמר דווח שהשתמשו בbatch בגודל 4, מכיוון שתהליך האימון מאוד איטי עם גודל כזה, השתמשנו בbatch בגודל 32 (שגם הוא הומלץ במאמר).
6. בטעינת הדאטה בחרנו לדרוס את שיטת הtokenize הדיפולטיבית עבור המשפטים ולהשתמש דווקא בספריית spacy שמפרקת את המשפטים באמצעות אלגוריתם מדויק יותר עבור כללי הדקדוק בשפה האנגלית. שינוי זה הביא לשיפור משמעותי של לפחות 3% על הdev_set.
7. Weight_decay- עבור הoptimaizer שבחרנו הוספנו רגולזיציה מסוג L2 לחישוב הloss עם מקדם 0.000001 דבר ששיפר במעט את הדיוק.
8. Epochs- אימנו את המודל על כל קבוצת האימון במשך 300 אפוקים כאשר בכל אפוק חישבנו את הדיוק על כל הdev_set. הבחנו כי בערך באפוק ה170 אנחנו מתכנסים לאותם אחוזים ולכן בהרצה הסופית אימנו את הפרמטרים במשך 170 אפוקים שלאחריהם עברנו על הtest וחישבנו את הדיוק.

<u>מידת ההצלחה:</u>

במאמר דווחו 86.3 אחוזי דיוק על test, 89.5 על הTrain. המודל שלנו הצליח לשחזר את ההצלחה עבור הtest ואף להתעלות- הגענו ל86.5 אחוזים. לעומת זאת על הtrain הגענו רק ל85.3.

ייתכן כי הוספת הרגולריזציה ללוס איפשרה יותר טעויות מה שהוביל להכללה טובה יותר על הtest. אך בתמורה הדיוק על הtrain ירד.

<u>learning curve graphs:</u>