שיטות בעיבוד שפה טבעית

דו״ח פרויקט

שם: יונתן קויפמן ויהב כהן

תז: 212984801, 207261983

תאריך הגשה: 23.03.2023

המודל הבסיסי

בפרויקט שלנו, בחרנו להשתמש במודל המאומן מראש t5-base. בחרנו במודל זה משום שהוא אומן מראש על משימת תרגום מאנגלית לשפות אחרות, וביניהן גרמנית. לכן, המודל כבר מכיר את המבנה של השפה הגרמנית כשפת יעד, ובאמצעות שימוש ב fine-tuning המודל ילמד איך להתייחס לגרמנית כשפת המקור ולאנגלית כשפת היעד והלכה למעשה, לבצע את המשימה שהוטלה עלינו בפרויקט.

לאחר הבחירה במודל, התחלנו לבנות את אלגוריתם האימון.

ראשית, קראנו את הנתונים מקובץ האימון לפי הפונקציה read_file שהסגל סיפק. לאחר מכן, המרנו את הדאטה לפורמט של CSV כדי שנוכל לטעון אותו לאחר מכן לצורה המובנית של datasets.

כדי לעזור למודל להתאמן ולהגיע לתוצאות טובות, ביצענו prompting – הוספת תחילית לכל input אשר תכוון את המודל. התחילית שהוספנו היא: Translate German to English:. כעת, כאשר יש בידינו את הקלט הרצוי, קודדנו אותו באמצעות tokenizer מובנה שמגיע מהמודל המאומן t5-base. לאחר הקידוד, בנינו את הליך האימון באמצעות Trainer של huggingface. הגדרנו כי אורכו של תהליך האימון יהיה 4 אפוקים מתוך חשיבה שאנו לא רוצים לבצע over fit ולאבד למעשה את יכולותיו של המודל הראשוני שקיבלנו.

Hyper parameters נוספים שהגדרנו הם גודל המשפט המקסימלי, כמות ה beams וגודל ה batch.

אם לא מגדירים ל Trainer גודל מקסימלי של ג'ינרוט, הוא ייצר משפטים קצרים באופן מובנה. לכן הגדרנו את הגודל המקסימלי להיות 512. בנוסף, הגדרנו שהמודל ישתמש ב beam search בגודל 2. כנלמד, בשיטת beam search האלגוריתם לא בוחר את המילים בצורה גרידית (יבחר את המילה הבאה שתביא את ההסתברות הגבוהה ביותר), אלא יחזיק ביד כמה היפותזות ויבחר את זו שתניב את ההסתברות הגבוהה ביותר לכל אורך ההיפותזה. בנוסף, beam search תמיד יבחר משפט עם הסתברות גבוהה יותר ממשפט שנבחר על ידי greedy search, אך עדיין אין הבטחה לבחירה במשפט האופטימלי מבין כולם. גודל ה batch שנבחר הוא 4 שכן זהו הגודל המקסימלי (מניסויים שערכנו) שהמכונה מסוגלת להתמודד איתו ללא קריסה.

האופטימייזר שעבדנו איתו הוא adamw_torch. אלו התוצאות שקיבלנו במודל הבסיסי (פלטים של התחזית על קובץ המבחן בכל אפוק בתהליך האימון):

Epoch	Training Loss	Validation Loss	Bleu	Gen Len
1	1.592700	1.364280	30.483900	87.179000
2	1.456300	1.307419	32.491600	86.927000
3	1.371100	1.285050	33.038100	86.275000
4	1.329900	1.281122	33.104600	86.850000

ניתן לראות בטבלה את ה loss על מדגם האימון, ה loss על מדגם המבחן, ה BLEU והאורך הממוצע של משפט שהמודל ג'ינרט.

לאחר שהיה בידנו מודל שעבר fine-tuning והותאם למשימה, ביצענו ניסויים על פרמטרים נוספים שמשפיעים על התחזיות כגון: top k, top p ו do sample. נסביר על כל אחד מהם בקצרה:

- Top k: במקום לבחור מבין n מילים בצעד הבא, נבחר מבין k המילים הסבירות ביותר
- Do sample: הכנסת אקראיות בבחירת המילה הבאה במקום בחירה במילה הסבירה ביותר.

- Top p: בשונה מ top k שבו קבענו k אחיד לכל הצעדים, כאן נאפשר בחירה דינמית של כמות המילים בהן נתחשב. בשיטה זו, נבחר מתוך כמות המילים המינימלית שסכום הסבירויות שלהן גדול או שווה ל p.

קיבלנו את ה BLEU הגבוה ביותר בשילוב 3 השיטות עם הפרמטרים שהוגדרו קודם לכן (אורך מקסימלי ו num beams). לאחר הרצת כמה ניסויים עם הצבות שונות של ערכים בשיטות, קיבלנו את ה BLEU הגבוה ביותר עבור: top k = 10, top p = 0.8 ו Do sample = True. הערך שקיבלנו הוא: 33.8.

המודל התחרותי – חלק 1

במודל התחרותי שילבנו את ה roots ואת ה modifiers שניתנו לנו. השתמשנו בספריה Spacy ובמודל המאומן "en_core_web_sm". זהו מודל ייעודי למציאת עצי תלויות למשפט קלט באנגלית. המודל מקודד כל מילה ל token אשר מכיל את המידע הבא: המילה עצמה, סוג התלות שלה בעץ התלויות, ה POS של המילה ומי הילדים שלה בעץ התלויות. אזי, בהינתן משפט קלט באנגלית נוכל לחלץ ממנו את הroot ואת ה modifiers שלו בכך שנחלץ את המילה שסוג התלות שלה הוא ROOT ורשימת הילדים של המילה תהיה ה modifiers במשפט. נקודה מעניינת ששמנו לב אליה: המודל לוקח פיסוק כגון נקודה או פסיק כ modifiers אפשריים. חשבנו על זה והחלטנו שאנו רוצים להוריד את הפיסוק מרשימת ה modifiers שכן הוא לא יתרום למודל ולא יעזור לו להבין טוב יותר אילו מילים צריכות להיות במשפט. במדגם האימון המקורי אין roots ו modifiers ועל כן ייצרנו כאלה בעצמנו באמצעות המודל שתואר לעיל. שמרנו קובץ אימון חדש כך שאת השורשים ואת ה modifiers כתבנו אל הקובץ לפי הפורמט של קבצי התחרות. כמו כן, כדי שיהיה לנו נוח, יצרנו קובץ val חדש שבו יהיו הפסקאות בגרמנית, התרגומים באנגלית והשורשים וה modifiers שנתנו בקובץ val.unlabeled.

בשלב הבא קראנו את הקלט מהקבצים האלה ובנינו את הקלט למודל. בחרנו לשלב את ה roots ואת ה modifiers בשלב הבנייה של הקלט למודל. עשינו את זה בצורה הבאה: יצרנו template שחלקים ממנו יהיו זהים לכל הקלטים (החלק השחוק) וחלקים ממנו ישתנו בהתאם לקלט (החלק הכחול):

"For the following German paragraph: {de_p}, translate each sentence with the corresponding English root and modifiers: {Root_sen_1} {(modifier_1_sen_1, modifier_2_sen_1,....)} , {Root_sen_2} {(modifier_1_sen_2, modifier_2_sen_2,....)}, ... "

החלקים השחורים זהים לכל הקלטים בעוד שהחלקים הכחולים משתנים בהתאם לקלט.

בנינו datasets לאימון ולמבחן על סמך הtemplate לעיל והרצנו את האימון בזהה למה שעשינו בחלק של המודל הבסיסי. קיבלנו שיפור ניכר בתוצאות:

Epoch	Training Loss	Validation Loss	Bleu	Gen Len
1	1.371400	1.261358	34.058200	84.541000
2	1.233100	1.205643	35.926900	85.540000
3	1.151900	1.186782	36.424700	85.146000
4	1.103700	1.184719	36.738400	85.131000

נשים לב כי בהוספת ה roots וה modifiers כבר ב epoch הראשון הגענו לערך BLEU של 34.05 אשר גבוה מהערך שקיבלנו במודל הבסיסי בהוספת השיטות השונות לייצור משפטים.

המודל התחרותי – חלק 2

כדי לשפר תוצאות אלו, ניסינו לחקור את הנתונים שהביאו לנו. ראינו כי הן בקובץ התחרות והן בקובץ comp נמצאים רק 2 modifiers ואילו המודל שלנו מחזיר יותר. לאחר קריאה בפורום השאלות הבנו כי ה modifiers שמופיעים נבחרו באקראי מבין כל ה modifiers האפשריים של המשפט. אזי, המודל שלנו מתאמן על רשימה גדולה של modifiers בעוד שבקובץ ה val מסופקים לו רק זוג modifiers. החלטנו להוריד את כמות ה modifiers ל 2 גם כן מתוך השערה כי:

א. נוכל לאמן מודל על התפלגות שתהיה קרובה יותר להתפלגות הנתונים שהמודל יבחן לפיהם ועל כן ביצועיו ביחס למדגם המבחן ישתפרו.

ב. ריבוי modifiers מייצר קלט ארוך למודל וייתכן כי המודל יתפקד טוב יותר עם קלט קצר יותר.

אזי השיפור שהכנסנו הוא בחירה של 2 modifiers בצורה אקראית מבין ה modifiers שמצאנו. ואכן, השערתנו אוששה וקיבלנו תוצאות טובות יותר:

Epoch	Training Loss	Validation Loss	Bleu	Gen Len
1	1.469300	1.237601	35.122100	84.872000
2	1.331400	1.183306	37.108100	85.889000
3	1.247800	1.159439	37.863900	85.607000
4	1.205900	1.157485	38.072600	85.875000

לבסוף, ברגע שהיה בידנו מודל מאומן יצרנו פרדיקציות באמצעות הוספת השיטות שתוארו בחלק של המודל הבסיסי.

ערך ה BLEU הסופי שהתקבל מהמודל התחרותי שלנו על קובץ ה val הוא: 40.04.

משום שיש אלמנט רנדומלי במודל שבא לידי ביטוי בשיטות הג'ינרוט שהשתמשנו בהן (do_sample וכו'), אנו צופים כי ערך ה BLEU שנקבל על הקובץ comp יהיה קרוב ל39.8.

חלוקת העבודה בין 2 חברי הקבוצה

יונתן קויפמן: מציאת מודל תיוג עצי תלויות וכתיבתו באלגוריתם שלנו, כתיבת הדו"ח, בחינתhyper parameters בג'ינרוט וכתיבת הקוד שממיר את הקלט הנתון לקלטים של המודל (בין אם זה למודל הבסיסי ובין אם זה למודל התחרותי).

יהב כהן: כתיבת כל הפונקציות שכותבות אל קובץ ויוצרות אותו, בין אם זה יצירת קובץ אימון חדש עם השורשים וה modifiers ובין אם זה יצירת הקובץ התחרותי עם הפרדיקציות. כתיבת קוד שבוחר 2 modifiers מתוך כל הרשימה ושומר אותם ואת הקלט המקורי לתוך קובץ חדש.

את מרבית הקוד, הרעיון למודל התחרותי והשיפורים הנדרשים בקוד עשינו ביחד.

הורדת המודל התחרותי

המודל התחרותי שלנו שמור בתיקייה our_dir אשר נמצאת בקישור הבא:
https://drive.google.com/drive/folders/1WiMZxMq6kfNHqS-Zt9LpmzP2w5U2y1jv?usp=share_link
בתיקייה יש תיקייה פנימית שנקראת checkpoint-10000 והקוד טוען את המודל מתוך: our_dir/checkpoint-1000.