

טנספורמרים

הקדמה

חלק מעבודת התזה שלי התעמקתי לאחרונה בארכיטקטורת הטנספורמרים. כשקרأت את המאמרים הראשונים בתחום (Attention is all You Need, ViT) התקשתי לעבוד את הרעיון ה konkurrensial שליהם מובסים הטנספורמרים לרעיונות מופשטים. כתוצאה לכך צלתי לנושא, ניתרתי קוד מספר מקורות באינטרנט לצד הוראות המתאמים, ולבסוף ניהلت ChatGPT סיבוב שאלות-תשובות כדי לחזור את ההבנה. לאחר שבוע וחצי של מחקר מאומץ, הגעת למסקנה שאני רוצה לשחרר את הדעת שצברתי עם העולם. ולכן, החלמתי לכתוב סדרת פוסטים שיאכל להביע דרכה את ההבנה שלי, ואולי לפחות תחילך למי שאין לו הזמן שהיה לי להשקיע בלמידה זו. חלק מהכתיבה, אסביר את ההיסטוריה של התחום, מבנה הרשת, יתרונות וחסרונות, כיצד מאמנים אותה. בשלב השני אסביר מאמרים בתחום הראיה הממוחשבת המציגים את השימוש ברשת למשימות כגון סגננטציה, הדבקת תמונות (Image Matting), העתקת מנה גוף (Gait Transfer), סיוג וכדומה. בכלโพסט אוסיף קישור לקובץ מתעדכן המכיל את כל המידע המתוקצר בפוסטים.

prtach, במשך זמן מה התלבטתי האם לכתוב את התוכן הזה בעברית או באנגלית. לבסוף בחרתי בעברית מכיוון שישנו חומר בנושא באנגלית,ומי שירצה יכול בחיפוש למצוא אותו. לעומת זאת בעברית אני מצליח להביא את הקול הייחודי שלו. מחלל לכם קריאה מהנה, ולី הצלחה.

טנספורמרים הוצעו בהתחלה כפתרון ניתוח טקסט. המאמר שבו הטנספורמרים הופיעו לראשונה נקרא *Attention is all You Need*, הכותבים השתמשו בארכיטקטורה זו למשימת תרגום מאנגלית לצרפתית. נאמר זה למעשה פותח את הסכר למחרפת ה-NLP שאנו רואים היום. טנספורמרים כולו מספר רעיונות חדשים בתחום השפה. השניים המרכזים, שוררים אחד בשני בארכיטקטורה, ומהווים את אבני הבניין הקונספטואלים של הרשת (בנוסף לעוד חידושים שהמאמר הציג). הראשון היה עיבוד מידע מקביל, שהוביל ליעילות חישובית באימון המודול ביחס למודלים קודמים (UVMSTANRNN), ואפשר בכך הראשמה לפזר את מחסום הלמידה התלויה בזמן של קלט סדרתי. כלומר, ניתן ללמוד במקביל תלויות קצרות וארוכות טווח בקלט סדרתי. נציין שגם טנספורמרים מוגבלים ביכולתם לעבד קלט מקביל, אולם זהי מוגבלת התלויה במשאבי חישוב כדוגמת זיכרון זמני ויחידות עיבוד (כרגע מספר הטוקנים המקסימלי הוא 1024 [1])

בנוסף, הרשת הצינה שני מנגנוני תשומת הלב(attention). הראשון הוא תשומת לב- עצמית (self-attention) שאפשרה למודל להתמקדם במידע החשוב ביותר באופן סלקטיבי. המנגנון השני הינו תשומת לב מוצלבת (cross-attention) שאפשרה מידע תלויות וקשרים בין חלקיו הקלט והפלט השונים. תוכנות אלו הרכזיות במשימות כדוגמת תרגום ומענה על שאלות סיקום טקסט, שדורשות בחירה מודעת בחלקים החשובים ביותר של הקלט והפלט. רעיונות אלו הלהבו אותי, ונגרמו לי לרצות להבין מה הוביל להסתפקותם. על מנת לעשות זאת התחלתי לסקור את הארכיטקטורות שקדמו לטנספורמרים, כיצד הם עבדו, ומדוע לא הצליחו במשימה שהטנספורמרים כן הצליחו בה.

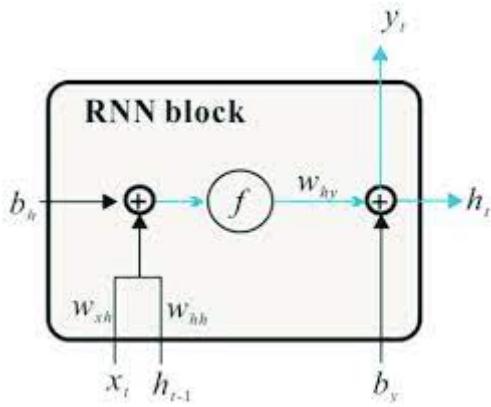
פרק 1 - ארכיטקטורת RNN

לפנינו שנדון בפתרונות שהוצעו לבעה, נפתח בהצגה שלה. כאשר אנו מדברים על בעיה, עלינו להתייחס למורכבות (complexity) שלה. מורכבות של משימה הנשענת על ניתוח נתונים יכולה להתבטא בכמות המימדים, גודל הקלט, רעש, דגימות חריגות (outliers), ודפוסים מורכבים החובים בדעתה. בהקשר זה, שפה טבעיות היא בעיה עם מורכבות גבוהה, שכן היא מכילה תלות ארוכה וקצרת טווח בין מרכיבי הקלט השונים, סלנג, כפל משמעות (כדוגמה סרקיום), וסמנטיקה חביה וגלואה. שפה מקיימת גם כללי תחביר ודקוק, כאשר משמעות של מילה מסוימת תלולה בהקשר בו היא נאמרת, ועל כל אלו שפה אינה שומרת על

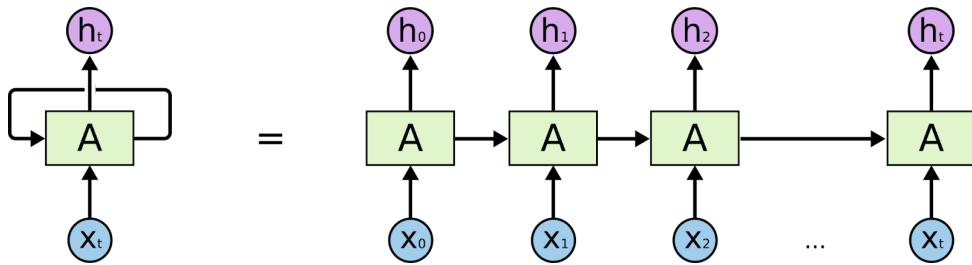
רציפות באופן מוחלט. אלגוריתמים בעלי חוקיות מוגדרת מראש (rule-based) כדוגמת Cyc שנייה לפרמל פתרון לנитוח שפה באמצעות סט חוקים פנימיים, נכשלו בעקבות כך.

נדגים את מורכבות ניתוח שפה, באמצעות בעיית התרגומים, ואת הקשיים שהיא מציפה. נניח ואנו מעוניינים לתרגם את המשפט הבא מאנגלית לעברית: "The dog sleeps" - "הכלב ישן". ניתן לתרגם כל מילה במשפט ללא תלות במילים האחרות. הכלל היחיד שעליינו להתחשב בו הינו שהמילה "the" תוסיף תחילית "ה" למילה שבאה אחריה. וכך, נוכל לשמר מילון המכיל לכל מילה באנגלית את התרגום שלה בתוספת לחוקים שהצענו ולקבל תרגום תקין. אולם, בהינתן המשפט "We're all mad here" (עליסה בארץ הפלאות) תוצאה התרגומים שבוצע בהתאם לחוקים אליו תהיה "colsno את כל מטורף כאן". שורש הבעיה הוא תרגום המילה הנוכחית ללא התחשבות בהקשר(context), לעומת במהלך התרגומים אנו חיברים להתחשב בתלויות בין המילה שאנו מתרגמים לבין המילים האחרות במשפט. ניתן להציג פתרון המשך את המילון שהצענו, ולשמור בנוסף על המילים והתרגומים שלהם, גם צמדים מילים, אך זהו פתרון נקודתי לבעה מבנית, וכי מהר נצטרך להרחיב את המילון לביטויים בעלי 3, 4 ועוד מילים. ולבסוף, נתכנס לכישלון המשותף לכל האלגוריתמים שניסו לפתור את הבעיה בצורה (rule-based), והוא חוסר יכולת למדל את חוקיות הבעיה בצורה ייעלה.

הארQUITקטורה הראשונה שניסתה למדל את התלות בין מילים בקלט מתוארת לוסף שנות השמונה של המאה הקודמת [2], ונקראה RNN. הרעיון המרכזי מאחריה הוא ניתוח הקלט הנוכחי, בתוספת מידע שהתקבל מהימים האחוריים במשפט. הרשות מקבלת מילה במשפט ומיצרת שני פלטים, הראשון הוא פלט המתאים למילה ביחס למשמעות (תרגום, תקציר טקסט, יצירת טקסט וכו'), והשני הוא ההקשר שנמצאה עד עתה ומיצג תמצות של כל המידע שהרשות קיבלה עד אז (bijzog yossa). פתרון פשוט לבעה זו הינו שרשור של יחידות חישוב כ אלה, כך שכל יחידת עיבוד (שנקרא לה בלוק מטעני נוחות החל מכאן) מקבלת קלט בזמן t (כלומר את הקלט בPOSITION t-1 במשפט/טעט טקסט) ואת ההקשר מזמן t-1 ומיצרת את הפלט המתאים לו. אולם, משפטים שונים מכילים מספר מילים שונה, ולכן, לא ניתן להזיז מספר בלוטי מוגבל של בלוקים כך שכל בלוק יתאים לקלט מסוימת במשפט. בנוסף בארQUITקטורה שכזו הבלוקים הראשונים יאומנו הרבה יותר מהבלוקים האחרונים, דבר הנובע מכך שבמשפטים, קיצרים מספר הבלוקים המרבי, ירפסו באפסים, והבלוקים האחרונים לא ישתתפו באופן מפורש בתהליכי הלמידה. ולכן, הפתרון הינו הפעלה איטרטיבית של הרשות. כלומר שימוש בבלוק יחיד וכתוצאה לכך בסיס משקלות מסווג, כך שהרשות מקבלת בכל איטרציה קלט חדש ואת ההקשר שהתקבל ב叵ונזה האיטרציה הקודמת. הייצוג הטורי מהו זה פרישה (unfolding) של הרשות (ראה איור 2), ביצוג הקומפקטי: הרשות מקבלת קלט A ואות המצב הפנימי מהאיטרציה הקודמת שמתוארך -_{i-1}H, ומוציאה פלט Z, בעוד שהמצב הפנימי H מוחזר בחזרה לרשות באיטרציה הבאה.



איור 1: מבנה רכיב RNN



איור 2: RNN ביצוג טורי (מיימי) וביצוג קומפקטי (unfolded)

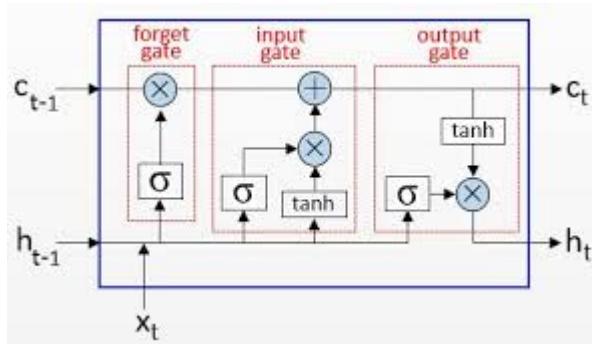
אז כיצד משתמשים ברשת כזו לניתוח שפה? טרם השימוש מקודדים את המילים/חלקי מילים (word/subword embedding). פועלות הקידוד היא המרת מילה בשפה טבעית, לווקטור בעל k ממדים, כאשר הייצוג הווקטורי מכבץ מילים בעלות משמעות דומה תחת ערכיהם קרובים, ומרחיק מילים בעלות משמעות שונה אחת מהשנייה. **ב ksiישׂוֹר** הנ'יל ניתן לראות שיטות שונות לבצע זאת, כאשר הנפוצה ביותר היא שימוש ברשת נוירונים (לדוגמה word2vec). הבעה באלגוריתמים אלו, הינה שקידוד מילים בעלות כל משמעות יהיה זהה, דבר שמקשה על הלמידה. בהקשר זה, אחד החידושים שאריכטקטורת הטרנספורמרים הצינה היה שילוב שיכון (embedding) המילים כחלק מהאריכטקטורה, דבר שאפשר במידה של שיכון המילה ביחס למילים קרובות למילה הנוכחית, אלא גם מילים רחוקות ממנה (כלומר שיכון מילים במרחב תלוי הקשור או contextualized embeddings).

על פניו, נראה שהרשת מספקת פתרון לבעה, היא מסוגלת לאנוור מידע ולmdl את התלות בין חתימות הזמן השונות. איז מדוע זה לא עובד? ישן מספר סיבות האחראיות לכך. הראשמה נועצה במבנה הטורי של הרשת, שהוביל כמעט תמיד לדעיכת הגרדיאנט לאפס (vanishing gradient). כפי שהוזכר בהתחלה, במקרים מסוימים השפה הפלט עבור קלט מסוים מלא בסדרת קלטים שקדמו לו, ולכן, בעת אימון הרשת המשקولات מעודכנות רק לאחר העברת הסדרה כולה. בנסיבות איטרטיביות, יש לחשב את הגרדיאנט של המשקولات גם כן בסדר איטרטיבי אולם הפעם הוא הפוך. כדי לחשב את השפעת המשקولات על השינוי בשנייה, علينا לכפול בטור את הנגזרות החלקיות של כל מצב פנימי ? ביחס למצב הפנימי -1-. מכיוון שהשניינו מחושב עבור אותו סט משקولات, הגרדיאנט יכול לדעוך או להתבדר לאורך האימון. ניתן להמיחס זאת באמצעות הערכים העצמים של מטריצת המשקولات; במידה והרטנים מ-1, הגרדיאנט יידרך מכיוון שמטריצת המשקولات מקטינה אותו בכל פעם שמתבצע עדכון, במידה והערכים העצמים גדולים מ-1.

הגרדיינט יתבדר מכיוון שמטריצת המשקولات מגדילה אותו בכל איטרציה ([דיעיכת הגרדיינט ופתרונות](#) [ב-RNN](#)). הבעה השנייה נובעת מהיחס בין גודל המוצבים הפנימיים והיכולת שלהם לדוחס מידע, לבין גודל הסדרה שהרשת נדרשת לנתח. המצב הפנימי "דוחס" את כל המידע עד לנקודת הזמן הנוכחיית, ולכן, כאשר הסדרה עוברת סוף מסויים בוגדלה ומתחילה להיווצר תוליות ארוכות טווח בין מרכיביה, המצב הפנימי הופך להיות צואואר-בקבוק בראשת והוא מתחללה "לשוכח" פרטימ שרחוקים מהקלט הנוכחי שלה. הבעה האחרונה מצהה במלין השימוש בראשת לאחר האימון (inference). הרשת אינה מנצלת את מלאה הפוטנציאל החישובי שלה (ושל המכונה שהיא רצה עליה) כתוצאה מאופי קבלת הקלטים באופן טווי, דבר הנורא זמן הסקאה איטי (כلمיר latency נבוה) של המוצבים הפנימיים במנוען משימות NLP. בעיה מहותית זו נפתרת באמצעות הטרנספורמים שנדרש עליה בהמשך.



פרק 2 - ארכיטקטורת LSTM



איור 2: בלוק LSTM

כפי שראינו, דעיכת הנרדיאנט וכשל בדיחסת מידע מוגנים מרשת RNN את יכולת להתרמס בחלקים החשובים ביותר בסדרה. כתוצאה לכך, הרשות מתקשה לאתר את התלות בין מרכיביה השונים שזו כאמור המטריה העיקרית בעיבוד שפה טבנית. כיצד ניתן להתגבר על בעיות אלה ברשומות המקובלות כקלט מידע סדרתי?

על מנת שנוכל לענות על שאלה זו, נבחן כיצד בני אדם מתמודדים עם סוג מידע שכזה. נציג הפיטה של הרעיון: בני אדם משתמשים בשני אלמנטים עיקריים: הראשון הוא זיכרון לטוווח ארוך וקצר, והשני הוא יכולת להפריד בין טפל ועיקר. שני מנגנונים אלו קשורים אחד בשני באופן הפעולה שלהם. כלומר, אנחנו מסוגלים להפריד בין טפל לעיקר בזכות ניסיון עבר והשלכתו על סיטואציה חדשה.

זהו בסיס הרעיון שהרשת הבאה מנסה לישם. [LSTM](#) (ונRSAה האלטרנטיבית GRU) היא ארכיטקטורה איטרטיבית שבדומה ל-RNN מכילה זיכרון לטוווח קצר H_t , (המצב הפנימי ב-RNN) ובנוסף, רכיב זיכרון ארוך, לטוווח C_t . בניגוד ל-RNN שמתמצבת את המידע שהתקבל עד לנקודת הזמן הנוכחי מזכיר קצר לטוווח, ההפרדה לזכרון לטוווח קצר וארוך ב-LSTM מאפשרת לרשות לשמור את מאפייני הקלט העיקריים גם בטוווח הארוך וпотרטת את בעית צואර הבקבוק של RNN. מבנה הרשות מאפשר לה לשמור את שני סוגי הזיכרונות והמילה החדש בכל איטרציה, ולנתח את המילה החדשה באמצעותם.

ניהול זיכרון מתבצע באמצעות שלושה שערים: שער השכחה, שער הקלט ושער הפלט (איור 2). **שער השכחה** אחראי על הסרת מידע מהזיכרון לטוווח הארוך לאחר שווה התNELLA כלא רלוונטי יותר. **שער הקלט** אחראי על הזरמת המידע לתוך תא הזיכרון. שער זה משתמש במידע המתתקבל מהמצב הפנימי הקודם ומקלט חדש, ומחליט מה המידע החדש רלוונטי ויש להוסיפו לתא הזיכרון. **שער הפלט** אחראי על חישוב מוצא הרשות $-H_t$, המהווה למעשה את הזיכרון קצר הטוווח.

היתרון של LSTM מtabטא בעיקר במשפטים ארוכים, לדוגמה, במשפט "She left", תרגום המילה "left" על ידי RNN תהיה "הלה" ולא "הלה" (או "שמאל") מכיוון שההקשר הוא קרוב. אולם, אם ניקח את המשפט "The goalie was determined to protect the goal" את המילה האחורה ניתן לתרגם כ"משימה/מטרה" או "שער (כדורגל)", במקורה זה, מילא ההקשר "goalie" (שער) מרוחקת מהמילה שאנו רוצים לתרגם, ולכן RNN עלולה לטעות בתרגום, במקרה זה יבוא לידי ביטוי הזיכרון ארוך הטוווח, שמתחשב במילה "goalie" בתרגום המילה "goal".

מדוע LSTM סובלת פחות מדעיכת הנרדיאנט? הסיבה נובעת בכך שריכב הזיכרון אינו מתעדכן בפלט של פונקציית האקטיבציה כמו ב-RNN, אלא בשימוש ב모ץ שער השכחה ושער הקלט. שער השכחה מסיר חלקים מהקלט על ידי יצירת קטור של ערכים רציפים במרקוטע של $[0,1]$, כאשר 0 משמעו שכחה מוחלטת,

ו-1 מהו זה שימוש מוחלט של המידע. הקלט והמצב הפנימי קבועים כמה מידע יש לשכוח, אולם הם אינם קבועים מה יהיה תוכן תא זיכרון לאחר העדכון. מן הצד השני, הוספת מידע חדש לזכרון אינה מתבצעת כד:rightה של התוכן על ידי הקלט, אלא כחיבור שלו. שער הקלט מייצר וקטור שמתווסף למידע הקיים בתא זיכרון בפעולת חיבור (element-wise addition) ובכך מאפשר את העברת האינפורמציה ברשות ומונע את דעיכת הנדרידיאנט, מכיוון שפעולה זו פחות רגישה לשינויים בקלט כפי שהוא פונקציית האקטיביציה ולא מתאפשרות, רגישה אליהם. כתוצאה לכך, הנמנחות שאין מושפעות באופן ישיר מפונקציות האקטיביציה ולא מתאפשרות, מאפשרות את עדכון המשקלות (בשונה מ-RNN שהמנחות מתאפשרות די מהר). בנוסף, מכיוון שעור השכחה מייצר וקטור הקובע אילו חלקים ברכיב הזיכרון יש לשכוח ואילו לשמר, כאשר ערך הוקטור קרוב ל-1, המידע יכול לזרום בראשת מבלי להיזוק באופן משמעותי או במיללים אחרים, מתאפשרת שמירת מידע אחר טווח (מידול מתמטי של נושא זה ניתן למצוא [בקישור](#)). בזאת השימוש ברכיב זיכרון, המצב הפנימי מצטמצם להיות זיכרון לטווח קצר בלבד והשפעת צוואר הבקבוק שנוצרת ב-RNN פוחתת גם היא.

למרות היתרונות המובהקים של הרשת על פני RNN במשימות בהן ישנו צורך לנתח תלויות ארכוכות טווח, קיימות לה מנבלות בניתו שפה טبيعית הנובעת מmorphemes הארכיטקטורה ואופן הפעלה האיטרטיבי שלה. הפעלה הראשונה משותפת לכל הרשות האיטרטיביות, והוא חוסר יכולת להפעיל את הרשת במקביל, דבר הנורר הפעלה איטית של הרשת ואי ניצול מספק של משאבי החישוב, בנוסף, לא מאפשרת למדל תלויות בין מקטעים שונים של הקלט.

מכיוון שכל תא LSTM מכיל מספר גדול יותר של פרמטרים בהשוואה ל-RNN, אימון והפעלת הרשת דרישים משאבי חישוב (זיכרון וכוח עיבוד) נוספים יותר. בנוסף, על מנת שהרשת תמדו את התלוויות ארכוכות הטווח באופן אפקטיבי ישנו צורך בכמות משמעותית של נתונים בעלת מספר TOKENS גובה (=אורך הסדרה). בנוסף, משך האימון הוא גם כן חסרון משמעותי של הרשת. חיפוש תלויות ארכוכות טווח בסדרה הינה משימה מורכבת יותר ביחס לחיפוש דפוסים מקומיים (local patterns) בDATA, וכן נדרש זמן אימון ארוך יותר.



פרק 3 - מנגןן תשומת הלב לפני עידן הטרנספורמרים

לאחר שראינו כיצד רשתות איטרטיביות כנו LSTM/RNN מודלות את התלויות בקלט סידרתי, נציג כעת את המנגנון המהווה את לבה של ארכיטקטורת הטרנספורמרים: מנגןן תשומת הלב (Attention mechanism). במהלך פרק זה נענה על השאלות הבאות:

- מהו מנגןן תשומת הלב?
- מדוע הוא נדרש?
- כיצד מנגןן תשומת הלב הפותח במכונות לומדות?

מהו מנגןן תשומת הלב ?

לפנינו שנגזר מהי תשומת לב במכונות לומדות, נבחן כיצד היא באה לידי ביטוי בקוגניציה האנושית. תשומת לב הינה התמקדות סלקטיבית בחלקים הרלוונטיים ביותר של מידע שאנו חווים וסינון חלקים בעלי חשיבות פחותה. תשומת הלב מאפשרת לנו למקד את המשאים העומדים לרשותינו בצורה עיליה בכל סיטואציה, ובכך אנו יכולים להתמצא ולהבין טוב יותר את מלאה האינפורמציה הזמין לנו. מנגןן תשומת הלב הוא יכולת מולצת של בני אדם, אולם היא גם יכולה לנלמדת הנינתנת לשיפור במהלך חיינו, (לדוגמא [מיינדרולן](#)).

האופן המופשט שבו מנגןן תשומת הלב פועל במוחנו הינו :

1. קבלת קלט על ידי הסנסורים החושיים שלנו (כגון מערכת ראייה, מערכת שמע וחוש הריח).
2. עיבוד מקדים וסינון המידע על ידי המוח.
3. בחירת החלקים החשובים ביותר של המידע בהתאם למידע קודם והקשר הנוכחי.
4. המידע שנבחר כרלוונטי עובר עיבוד, ובסיום נשמר בזיכרון.

מנגןן תשומת הלב למעשה מנע לנו בתור בני אדם, מכיוון שהוא מאפשר לנו **להתעלם** ממידע שאינו חיוני לנו (המקיף אותנו הרבה יותר מאשר המידע חיוני). דוגמה לכך היא "מסיבת קוקטייל" (cocktail party) שבה אנו נוכחים במפגש חברתי שבו מספר רב של אנשים מדברים בו זמני, ואנו מעוניינים להתמקד באדם אחד שמדובר. המוח שלנו מסייע לנו למקד את תשומת הלב שלנו באדם זה ולהתעלם משאר הקולות שהופכים לרעש רקע.

הנקודה המזקקת את העיקרון של תשומת הלב הינה היכולת לשער מידע (information assessment) במלילם אחריות לנו **לומדים להעניק משקל** לכל פיסת מידע, ביחס לכל פיסת מידע אחרת, בהתבסס על החשיבות שלה, והקשר שלה עם מידע שכבר ידועים (הנאנגר בזיכרון), או חווים ברגע נתון.

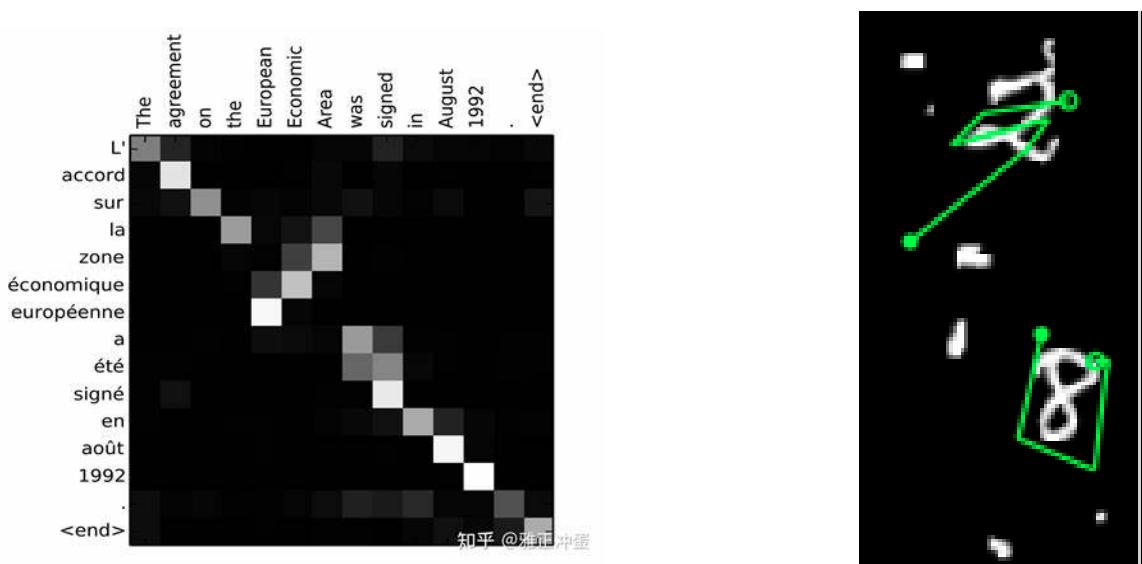
וזה מהו החידוש שמנגןן תשומת הלב מביא אליו כאשר הוא משולב ברשתות נירוניים? התשובה לכך היא שמנגןן תשומת הלב הוא **פונקציה הנלמדת** כחלק מתהליכי אימון הרשת, שמטרתה לחשב מהו המשקל היחסי של כל יחידת DATA בהינתן הקשר ומידע שנאגר עד אותה הרגע. מנגןן תשומת הלב יוצר ייצוג וקטורי רציף תלוי-הקשר (contextualized representation) עבור יחידת המידע (כגון טוקן או מילה), ותוכאטו הינה פונקציה רציפה שנitinן למזהר אותה ביחס לפרמטרים של מנגןן תשומת הלב (כלומר soft attention). מנגןן תשומת הלב משכך יחידות DATA במרחב וקטורי כאשר ייצוג של יחידת DATA במרחב זה מתחשב בעוצמת הקשר **הרציפה (לא דיסקרטית)** ביןיה לבין שאר היחידות DATA. לאחר והמשך

של ייחidot דاطה הינה פונקציה נירה ביחס לפרמטרים של מנגנון תשומת הלב (ונם ביחס לייצוג ההחלטה של מנגנון זה) אנו יכולים לאמן את הרשות לייצג את הקשרים האמייטיים בין בין החלקים השונים של הטקסט.

איור 1 (שמאל) המציג את מפת תשומת הלב של משפט אנגלית ביחס לתרגומו לצרפתית, ממחיש את עקרון הריצפות של פונקציית תשומת הלב. האיור מציג משקל של הטוקנים בסדרה הראשונה ביחס לכל הטוקנים בסדרה השנייה (מנגנון זה נקרא תשומת הלב המוצלבת אשר נרחיב עליו בהמשך). איור זה לquo מהמאמר שהציג את מנגנון תשומת הלב לאשונה עבור משימות שפה טבعت. נדון במאמר זה בפרק העוסק בתשובה לשאלת "יכן מנגנון תשומת הלב הפותח במכונות לומדות?".

אולם, לא כל מנגמוני תשומת הלב נולדו שווים. [תאמיר זה](#), שהציג את השימוש הראשון במנגנון תשומת הלב (במאמר הופיעה לראשונה המונח attention), עבר סיווג תסבוכות, השתמש במנגנון תשומת הלב דיסקרטי (hard attention). אופן פעולה הרשות דומה למשחק חיבור נקודות באמצעות קוים היוצרים צורה. הרשות מחפשת את הפיקסלים המקיימים את האזור הרלוונטי ביותר בתמונה (איור 1 מימין), כך שגם לחבר אותם בקוים ישרים, נקלט תמונה חדשה המכילה את המידע החשוב ביותר הדרוש לסיווג התמונה. בכל הפעלה של הרשות, מנגנון תשומת הלב בוחר פיקסל חדש, ומוסיף אותו לפיקסלים שנבחר באיטרציות הקודמות, כאשר בחירת הפיקסלים הללו יוצרת בסופו של דבר את האזור התוחם. מכיוון שפונקציית תשומת הלב הדיסקרטית אינה גירה (ומקבלת ערכים דיסקרטיים בלבד) לא ניתן למטב (optimize) אותה עם שיטות משפטת מورد הנגדיאנט (gradient descent). עקב כך המאמר עשה שימוש בשיטת אימון משטר מבוסס החלטות (policy opt) השאלה מ透ם הלמידה מבוססת החיזוקים (reinforcement learning), שלא דורשת מזרות של פונקציית מטרה, על מנת לאמן את מנגנון תשומת הלב.

בעולם ניתוח השפה, מנגנון תשומת הלב הדיסקרטי, לימד למצוא את הטוקנים החשובים ביותר (בהתאם למשימה) לבנייה של וקטור הייצוג עבור טוקן נתון. לדוגמה, נניח והמטרה של הרשות היא למצוא את האובייקט החשוב ביותר במשפט "The cat is playing with the toy, it is soft". הרשות תיתן את המשקל המקסימלי למילה "cat" ומשקל נמוך לשאר המילים.



איור 1 - בחירת הפיקסלים החשובים ביותר בתמונה (ימין) מפת תשומת הלב וציפה בתרגום משפט מאנגלית לצרפתית (שמאל)

תשומת לב "לא מפורשת" לעומת תשומת לב "מפורשת"

המנגנון שדנו בו עד עתה נקרא תשומת לב מפורשת (explicit attention), שבו אנו משערכים קשרים בין יחידות מידע שונות באופן יומי. לעומת זאת, ישנו מנגנון תשומת הלב נוסף הנקרא תשומת לב "לא מפורשת" (implicit attention). מנגנון זה הינו "творך לא מכון" של רשותות עמוקות. רשותות אלו נוטות להתמקדש בחלקים מסוימים של המידע ולהתעלם מאחרים. לדוגמה, בסיווג של מכח גוף מתוך סרטונים (pose estimation), תהיה לרשותה הנטיתיה להתמקדש באוזרים שבהם מופיעים חלקים חלקיים גוף ולהתעלם מאוזרים בהם לא מופיע אדם בכלל (ורק או אובייקטים דומים). ניתן להמחיש באופן ויזואלי את תשומת לב המרמזות על ידי איור 2, המגע מההרצתה הבהא. האירור ממחיש את מיקוד הרשות בעת כניסה מבוססת בעבר אימון סוכן במשחק. הסוכן לומד לנסוע בדרך השביל ולהימנע מפוגעים באמצעות למידה מושגת חיזוקים. החלקים הבוהקים באדם מייצגים את האוזרים בהם הרשות מתמקדת על מנת לקבל את החלטה הבאה. מכיוון שהתקדמות בשביל מוביליה לעלייה בנקודות הרשות מתמקדת באופק, ובולוח התוצאה שמציג את ההחלטה. **מעתה, בכל מקום בטקסט שנתייחס לתשומת לב, נתכוון לתשומת לב מפורשת.**



איור 2 - תשומת לב מרמזות ברשות מבוססת חיזוקים

מדוע מנגנון תשומת הלב נדרש?

כפי שהסבירנו בפרקם הקדומים, ארכיטקטורות איטרטיביות סבלו מבעיה מרכזית המשותפת לכלן, והיא רכיבי זיכרון הקבועים בגנודלים, ומאידר, קלט בעל אורך משתנה. כתוצאה לכך אנו נאלצים לפחות מושפטים באורךים שונים לוקטור בגודל קבוע. ולכן, בעת קידוד קלטים העולים על גודל מסוים, נתכנס לביעית מידול של תלויות ארוכות הטווח. עיקר הבעיה בא לידי ביטוי בכך שלא ניתן להשתמש בכל ייחדות הקלט באופן מפורש לבניה של וקטור ההקשר והעbor ייחידת DATAה. בambilים פשוטות, מטרתנו לאפשר לוקטור המוקוד את הקלט לגשת לכל חלקי (של הקלט) במקביל בעת בניית הייצוג.

על מנת להמחיש את הנושא, נסתכל על הפסקה הבאה:

"**צח, מהנדס תוכנה**, עבד מהבית בשנתיים האחרונים. הוא מתגורר ביישוב קטן בצפון הארץ עם אשתו ושני ילדים. היישוב שקט ורגוע ויש בו תחוות קהילתיות. צח נהנה לבളות עם משפחתו ולצאת לטווילים ארוכים בעיר הסמוכה. הוא מעריך את הగמישות **שבבודה מרוחק** מצעה לו. בשבועו שעבר נלה צח כי החברה שלו מתכוננת

לישם מדיניות חדשה שתחייב את כל העובדים לעבוד מהמשרד. צח שוקל כתת האם **לעbor לנור בקרבת המשרד** שנמצא במרקם הארץ או **לחפש עבודה חדשה**.

מודל ניתוח וסיכון טקסט המבוסס RNN או LSTM עשוי להתקשות להבין מהו המידע החינוי בטקסט זה וליצור סיכון תמציתית איקוטי. מכיוון שאפונן עיבוד הקלט הוא טוקן-אחררי-טוקן, תוצאה אפשרית של מודל איטרטיבי יכולה להיות:

"צח, מהנדס תוכנה, שעבודך מרוחק, עBOR לנור ליד המשרד, או לחפש עבודה חדשה, בוגל שניי במדיניות החברה"

למרות שהרשות אכן "דחסה" את כל המידע החשוב, עדין חסירה נהייה (קוורנטו) בפלט.

לעומת זאת, רשות המקימת את התנאים הבאים:

- בעלת מנגנון תשומת הלב.
- מייצרת וקטור הקשר גדול מספיק.

יכולת לספק את התמצאות הבא:

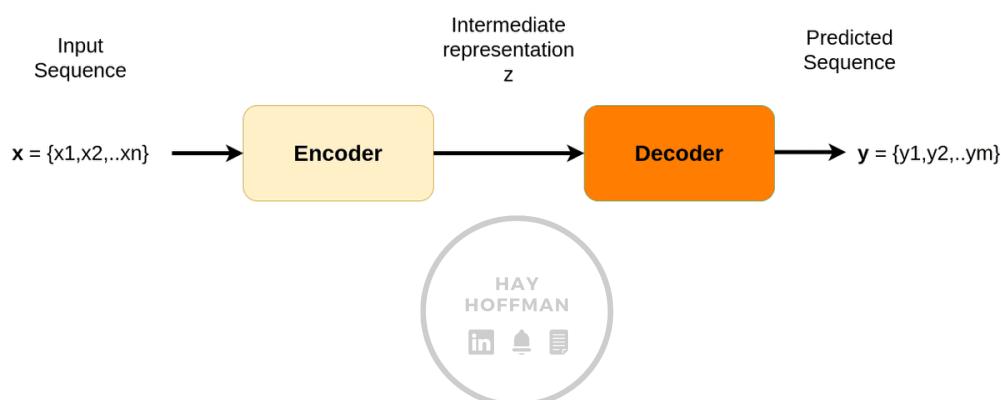
"צח, מהנדס תוכנה שעבודך מרוחק, שוקל האם לעbor לעיר הקרובה לעבודתו במשרד או למצוא משרה חדשה המאפשרת עבודה מרוחק בשל מדיניות החדש בחברה"

נקודות לסייע הפוך:

- מנגנון תשומת הלב במכונות לומדות הינו פונקציה נלמדת, השואבת השראה מתשומת הלב בكونכיזיה האנושית, וממשקית חשיבות של קלט ביחס לקלט אחר.
- מנגנון תשומת הלב שהוא דנים בו נקרא soft attention שמהווה פונקציה רציפה. תcona זו נובעת מהעובדת שימוש של יחידת>Data נבדק נסרך לב מפורשת, הינה פונקציית תשומת לב המומשת ישן שני סוגים של מנגנון תשומת לב. תשומת לב מפורשת, הינה פונקציית תשומת לב הגדירה על יוזם חלק מארQUITקTOT המודול. לעומת זאת, תשומת לב לא מפורשת הינה תוצר של עיבוד>Data על ידי רשתות עמוקות, הלומדות חשיבות של אזורים מסוימים בקלט ללא הכוונה יוזמה.
- רשותות בעלות מנגנון תשומת לב יכולות למדו קשרים מורכבים יותר בקלט וליצור פלט קוורנטו יותר ביחס לרשותות איטרטיביות.

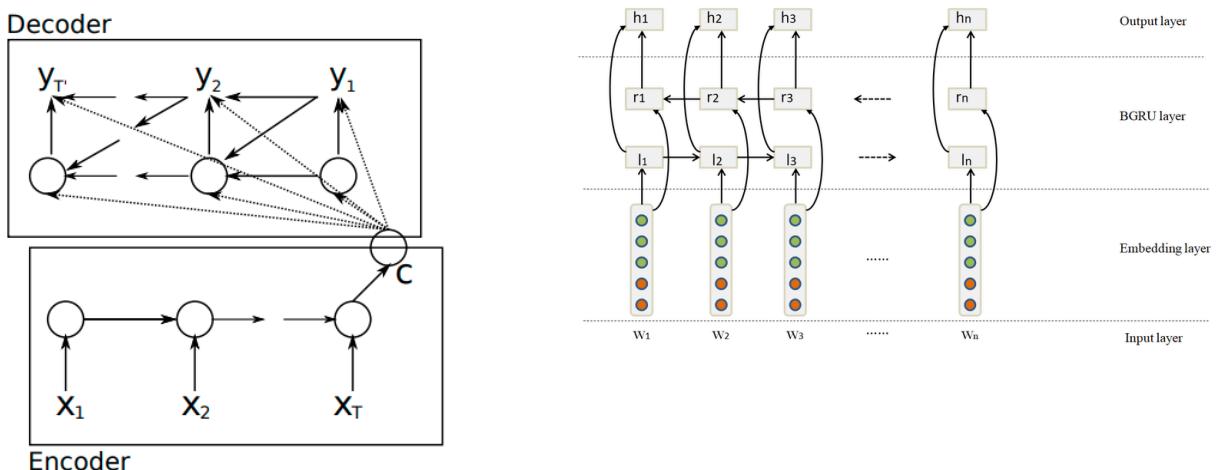
כיצד מנגנון תשומת הלב הפתח במכונות לומדות?

המאמר הראשון שהציג שימוש במנגנון תשומת הלב עבור משימות של ניתוח שפה טבעית השתמש בארכיטקטורת מקודד-מפענח (encoder-decoder architecture), המთוארת באיור 3, לטובות תרגום מאנגלית לצרפתית. על מנת להסביר את הצורך במנגנון תשומת הלב, נציג כתת מושג חשוב בתחום עיבוד השפה הטבעית הנكرة "יישור" (alignment). מושג זה מתאר בהקשר של תרגום, את התאמיות בין מילה/מילים משפט המקור לבין מילים בשפט היעד (איור 1 משמאל מדגים את היישור בין מאנגלית לצרפתית). במילים אחרות, זהו יציג של "עוצמת הקשר" בין קבוצות של מילים בשפט היעד לבין מילים בשפט המקור.



איור 3 - הפענה של ארכיטקטורת מקודד-מפענה

כאן המקודד מקבל מפענה (או קטע של טקסט) בשפה א' כסדרת טוקנים, ומיצג את המידע במשפט כוקטור במינימד חבוי (latent representation). מן הצד השני, המפענה מחלץ מתוך הווקטור שהתקבל את המידע הרלוונטי ומפיק מתוכו את המפענה בשפה ב'. הרשות מאומנת ממוקשה אחת, ולומדת לקודד ולפענה באותו הזמן. בamar גם המפענה וגם המקודד מומשו על ידי יחידות GRU משורשיות (אולם ניתן להשתמש גם ב-RNN או LSTM). כאשר המקודד מורכב מיחידות GRU דו-כיווניות ([bidirectional GRU](#)). ארכיטקטורת GRU דו-כיוונית מורכבת משני שרשרים של יחידות GRU כאשר השרשור הראשון משמש להעברת הקלט מתחילה לסיום, והשרשור השני של יחידות GRU משמש להעברת הקלט מהסוף להתחלה (ראה איור 4). הסיבה לשימוש בארכיטקטורה דו-כיוונית נובע בכך שאנו יכולים לקבל מידע מקיים על הקלט, כמו גם מתחילה וגם מסויים, שכן כל המידע (מפענה המקורי) נתון לנו בזמן הפעלת המודל לאחר האימון (inference). לעומת זאת הפלט במפענה נוצר באופן אוטורגרסיבי (מילה אחריה מילה), כאשר פלט הנוכחי הופך לקלט עתידי לאחר שנוצר) בזמן ההסקה, שהופר את השימוש ברשות דו-כיוונית במפענה במהלך משלול הינון.



איור 5 - ארכיטקטורות מקודד מפענה

איור 4 - GRU דו-כיוונית

בארכיטקטורות מסווג מקודד-מפענה שקדמו למאמר, הקלט שהמפענה מקבל בכל איטרציה (יצירת יחידת DATAה חדשה) הינו המצב הפנימי h_i והפלט y_{i-1} מהאיטרציה הקודמת (של המפענה). על מנת לחבר את המידע שהמקודד למד מהקלט, המפענה מקבל בנוסף את וקטור המוצא של המקודד שנקבע לו מעתה C . בארכיטקטורות מקודד-מפענה בסיסית, C הינו שרשור המ מצבים הפנימיים שחושבו באיטרציה האחרונה מכל אחת מהרטשות המרכיבות את הרשות הדו-כווניות במקודד ($[h_i; h_i \leftarrow h_i] = h_i$) שכן הוא מכיל מידע על כל הקלט. איור 5 מציג את הארכיטקטורה שהסבירנו בפסקה זאת.

כיצד בא לידי ביטוי מנגנון תשומת הלב במאמר?

כפי שהזכרנו קודם לכן, השימוש בתשומת הלב מיועד לפתור את בעיית היישור בין הקלט לפלט. עקב אכילים של הארכיטקטורות שקדמו לו של המאמר, הייתה שימוש במצבים הפנימיים האחרונים של המקודד (הפלטים של האיטרציה האחרונה של מושני הכווינים של GRU). מכיוון>Statusים אלו היכלו מידע דחוס על כל הקלט, לא ניתן היה למדוד את התלותות המוקומית בין המצבים הפנימיים של המפענה, לאלו של המקודד. כמו כן, וקטור הקשר המתקבלים בכניסה למפענה דוחסם את כל המידע מסדרת הקלט של המשמעות. לעומת זאת, הקשרות בקשר שבין יחידות מידע בסדרה א' ליחידת מידע הנבנתה בסדרה ב'. לעומת זאת, בשיטה המוצעת במאמר, וקטור הקשר שהמפענה מייצר בעת בניית יחידת פלט, מקבל את המידע על

כל יחידות הדאטה של הקלט שנבנו על ידי המקווד. וקטור ההקשר נוצר כסכום משוקל של כל הממצבים הפנימיים של המקווד, כאשר המשקלים ממדלים את הקשרים בין כל יחידות הקלט ליחידה פlut.

מנגנון תשומת הלב: המידול המתמטי

המושג הראשון שהמאמר מגדיר הינו **עוצמת היישור** (alignment score), שמייצג את הקשר בין המצב הפנימי i - במשמעותו לבין מצב פנימי j כלשהו במקווד. על מנת למןעו בלבול, נגידר את הממצבים הפנימיים של המפענה s - i (כפי שהוא מובא במאמר) ואת הממצאים הפנימיים במקווד נשאיר C - h . האינדקסים j , i , מייצנים את המספר הסידורי של יחידות הקלט והפלט i - j . כתע עברו יחידת plut i נגידר וקטור T $e_{ij} = e_j$ באופן הבא:

$$(1) \quad e_{ij} = \text{attention}(s_{i-1}, h_j) = v_a^T * \tanh(W * [s_{i-1}; h_j]), \quad j = 1, \dots, T$$

משוואה 1 - חישוב עוצמת היישור (מנגנון תשומת הלב)

כאשר:

- e_{ij} - עוצמת היישור לא מנורמלת.
- h_j - המצב הפנימי של יחידה j של המקווד.
- s_{i-1} - המצב הפנימי מהיחידה $i-1$ של המפענה.
- W - מטריצת המשקלות של מנגנון תשומת הלב.
- v_a - וקטור המשקל של פונקציית תשומת הלב.
- T - מספר יחידות הדאטה במקווד.

כאמור, מנגנון עוצמת היישור הינו פונקציה נלמדת המחשבת את עוצמת הקשר שבין המצב הפנימי של המפענה לממצאים הפנימיים של המקווד. מכיוון שהמכפלות בתוך פונקציית הטננס והיפרבולית \tanh יוצרם וקטור בגודל 1 \times 1, וערך תשומת הלב בין שתי יחידות דאיתה צריך להיות סקלר, המכפלה בוקטור v_a יוצרת סקלאר בפועל. נשים לב כי המטריצה W והוקטור v_a הינם פרמטרים הנלמדים (מאומנים) של המודל.

המושג השני שהמאמר מגדיר הינו **משקל תשומת הלב** (attention weight). מטרת מנגנון תשומת הלב הינה ליצור משקל החשיבות של טוקן אל מול כל טוקן אחר, כפונקציה רציפה ומירה. המשמעות של רציפות בהקשר שאנו מדברים עלי, הינו משקל של עוצמת הקשר e_{ij} (המקושרת למצב הפנימי i מהמקווד) ביחס לכל עוצמות הקשר האחרות. עוצמות אלו מיצגות את הקשר שבין כל שאר הממצאים הפנימיים של המקווד ביחס למצב פנימי הנתון של המפענה. על מנת לעשות זאת, אנו משתמשים בפונקציית softmax המופעלת על עוצמות הקשר. חישוב זה למעשה פותר את בעיית היישור שפתחנו איתה את הפרק, מכיוון שפונקציית softmax תהפוך את תוכאת היישור, שעמדה בפני עצמה, להיות פונקציית צפיפות הסתברות התלויה בכל הממצאים הפנימיים של המקווד. אנו מבצעים פעולה זו עברו כל מצב פנימי של המקווד אל מול אותו מצב פנימי של המפענה, ובכך מקבלים את עוצמת הקשר הרציפה שהזכרנו.

$$\alpha_{ij} = \exp(e_{ij}) / \left(\sum_{k=1}^{T_x} \exp(e_{ik}) \right) \quad (2)$$

משוואה 2 - חישוב משקל תשומת הלב עבור זוג יחידות דאיתה i,j



כאן \times הוא מספר ייחודת חישוב במקודד כלומר אורך מקסימלי של סדרת DATA שnitן להכנס בז'ר מכקה Achter.

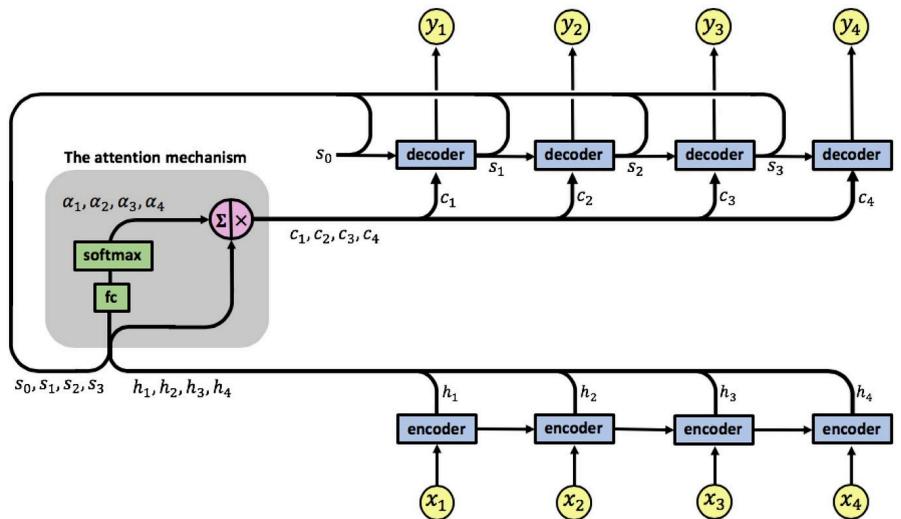
החלק האחרון בהסבירו שלנו הוא **בנייה וקטור הקשר דינמי C**. אנו משתמשים במשקלות תשומת הלב שיחסבנו, ומרכיבים את הוקטור המכפלה של משקלות זו במצב הפנימי של המקודד המקשר אליו. בניה זו של הוקטור הקשר מאפשרת לנו לחת למפענה את המידע הרלוונטי ביותר ביחס במצב הפנימי הנוכחי שלו. מצבים פנימיים של המקודד בעלי עצמת קשר נמוכה ביחס במצב הפנימי של המפענה, לא ישפיעו על הוקטור C (שכן אם ערך משקלות תשומת הלב שלהם נמוך, חלקם היחסי בוקטור C יהיה נמוך גם כן).

$$C_i = \sum_{j=1}^{T_x} \alpha_{ij} * h_j \quad (3)$$

משווה 3 - בניית וקטור הקשר הדינמי C

הקלט של המפענה הינו שכלל של המצב הפנימי הקודם שלו, וקטור הקשר, ומוצא הפלט הקודם המזומנים לתוך הרשות לבדוק לרשות אוטרטיביות קודמות שריאנו (LSTM&RNN). פונקציית השקלול מרכיבת מפונקציות האקטיבציה הפנימיות של בלוק LSTM (בדומה ל GRU ישם שער שכח ועדכו). בניית פלט המפענה מתוארת באירור 6.

אם נחזר לאיור 1, נוכל לראות כי לאורך רוב האלכסון, רק מילה אחת מהפלט מיושרת (קשורה) באופן מובהק למילה בקלט, ולכן המצב הפנימי מהמchodד המתאים יועבר כמעט בשלמותו לווקטור הקשר C (נוסחה 3). לעומת זאת, במקרים שבהם מילה בפלט תלויה (בהתאם למשקלות תשומת הלב) בכמה מילים מהקלט, וקטור הקשר יהיה מרכיב מסכם משוקל של מצבים פנימיים של המקודד. איור 6 מציג את הארכיטקטורה כפי שהיא מובאת במאמר בשמותה.



איור 6 - תיאור הארכיטקטורה בשמותה.

או במה מנגן תשומת הלב מסויע לנו?

מלבד יכולת לגורם למפענה להתרכז בקלט מסוים בעת החיזוי של המילה הנוכחי, מנגן תשומת הלב עובד באופן דומה למנגנון skip connection. אנו מספקים גישה ישירה בין המצבים הפנימיים של המchodד למפענה באמצעות הוקטור הדינמי שאנו יוצרים, ובכך מאפשרים למידע הקיים בהם "לזרום" כמעט ללא שינוי בדומה לאופן פעולה skip connection. וזאת בשונה מארכיטקטורות מקודד-מפענה ללא מנגן זה שחלק מהמידע אובד בעת יצירת הוקטור ביציאת המקודד. מנגן זה נותן

מענה לשתי החסרונות המרכזים של הרשותות האיטרטיביות: צוואר הבקבוק של ייצוג סדרות דאותה אורך וឌיעת הנדריאנט. בנוסף, מנגנון תשומת הלב מאפשר "פרשנות" ([Interpretability](#)) טוביה יותר לרשות.פרשנות היא מושג המתאר את יכולת שלנו בטור בני אדם להבין את התהליכים המתורשים בתוך רשותות לומדות. בהקשר של הארכיטקטורת שימושת שמשתמשות במנגנון תשומת לב, ניתן להבין כיצד המודל מייצר את הפלט באמצעות המשקל שכל יחידת מידע קיבלה, ומתוך כך למדוד את מגבלות הרשות ולמצוא דרכי לשפר אותה.

מה هي החסרונות של הארכיטקטורה שראינו עד כה?

אך על פי שהארQUITקטורה שהציגנו הייתה התקדמות עצומה בתחום ניתוח שפה טبيعית, היו לה מספר מגבלות.

המגבלה הראשונה נבעה מכמות המשאים שנדרשו לחישוב וקטור ההקשר C, שהוסיף על עומס החישוב הקיים גם כך בراتות איטרטיביות. כתע' נדרשות ($m * n$) הפעולות של פונקציית היישור (*alignment*) כאשר m מייצג את מספר הtokנים בקלט ו- n הוא מספר הtokנים בפלט. דבר זה גרם לזמן אימון והסיקה (*inference*) ארוכים במקודם.

המגבלה השנייה של הרשות הינה ייצוג ההקשר המוגבל שלה (*limited context representation*). הרשות לומדת את הקשר בין קלט לפולט. אולם, היא אינה לומדת את ההקשרים שישיחות המידע יוצרות אחת עם השניה (תליות פנימיות) בקלט ובפלט. לכן, היא לא יכולה "להבין" סמנטיקה מורכבת כגון סLANG, סרקיום, כפלו משמעויות, ומערכות יחסים עיקריות החוביות בקלט (שבתוון בני אדם אנו מבינים בקלות). דבר זה מוביל לכך שביצועי הרשות פוחדים ביחס ישיר לאורך הסדרה.

כיצד ניתן לפתור בעיה זו?

מנגנון תשומת הלב אותו תיארנו עד כה נקרא תשומת לב מוצלבת (*cross attention*). שכך, הינה (תשומת הלב) מצליבה את המידע מהמקודם עם זה שהתקבל מהמפענה. עתה, נציג מנגנון תשומת לב נוסף, **תשומת לב עצמית (self attention)** שהזנג לראשונה [במאמר זה](#). תשומת לב עצמית מ Chapman, נזכיר, שבארכיטקטורות מוקודד-מפענה, גם העצמת הקשר בין כל טוקן עם כל טוקן אחר באותה סדרה. וכך, נציג לשבץ את מנגנון תשומת הלב העצמית בכל אחד מהם. בהקשר זה **אחד החידושים המרכזיים של ארכיטקטורת הטרנספורמרים היה השימוש של שני מנגנונים אלו.**

או מדוע תשומת לב עצמית נדרשת מლכתחילה?

כאמור, אחת המגבלות של הארכיטקטורה הקודמת שהציגנו הייתה יכולת מוגבלת לעבד טקסטים מורכבים. מכיוון שהרשות **דק את עצמת הקשר** שבין מילה בשפה א' לקבוצת המילים בשפה ב' הדרישות לתרגם שלה (או במלחים אחרות בנתה את וקטור ההקשר רק מתוך המצביעים הפנימיים של המקודם). אולם, כאשר אנו ניגשים למשמעות תרגום, אנו צריכים למפות את התလיות המורכבות במשפט המקור על מנת לתרגם בצורה נכונה. על ידי שילוב של מנגנון תשומת הלב העצמית, ניתן לשקל את הקשר שבין מצב פנימי אחד לאחר, בנוסף על בחינת הקשר שבין מצב פנימי במקודם למפענה. נמחייב בעיה זו באמצעות דוגמא המצינה תרגום מאנגלית לעברית, ונראה כיצד ארכיטקטורת המשלבת את שני מנגנוני תשומת הלב תתרגם את המשפט לעומת ארכיטקטורה עם מנגנון תשומת לב מוצלבת בלבד.

נניח והמשפט אותו אנו מעוניינים לתרגם הוא:

"Despite the stormy weather causing some delays, the couple, who were accompanied by their close friends, managed to reach the mountaintop and enjoy the breathtaking view."

ארQUITקטורה המשתמשת בשני המנגנונים תתרגם את המשפט באופן הבא:



"למרות האיחורים שנרגם מזג האויר הסוער, הוג, שלוו אותו חברים קרובים, הצליח להגיע לפסגת ההר ולהנוף עוצר נשימה"

לעומת זאת ארכיטקטורה משתמשת רק במנגנון תשומת הלב המוצלבת עלולה לתרגם את המשפט כך:
"הוג, שלוו אותו חברים קרובים, הצליח להגיע לפסגת ההר, למרות האיחורים ומזג האויר הסוער, ולהנוף מהנוף עוצר נשימה".

למרות שני התרגומים קוהרנטיים ושמרו על כללי תחביר ודקדוק, התרגום השני נכשל בהבנת הקשר שבין האיחור למזג האויר. לעומת זאת, ארכיטקטורה שכן משתמשת בתשומת לב עצמית הצלחה למצוא קשר זה, ולהביא אותו לידי ביטוי בתרגום.



פרק 4 - ארכיטקטורת הטרנספורמרים

כפי שראינו בפרק הקודם, הארכיטקטורות שעשו שימוש במנגנון תשומת הלב לבדוק לא שרדו את מבנן הזמן, והוחלפו על ידי טרנספורמרים בכל המשימות הקשורות לניתוח שפה טבעי.

בפרק זה אנו נבצע ניתוח עמוק של ארכיטקטורת הטרנספורמרים, ונענה על השאלות הבאות:

- כיצד הטקסט מזון למודל הטרנספורמר?
- מהו קיוד תליי מיקום, ומהו משחק תפקיד חשוב בארכיטקטורת הטרנספורמים?
- מהם תפקידם של המזודד והפענה בטרנספורמרים, וכייזה הם עובדים יחד?
- כיצד מנגנון תשומת הלב בא לידי ביטוי בטרנספורמרים?
- מהו הייחוד של מנגנון תשומת הלב בטרנספורמרים ביחס לחישוב תשומת הלב ברשות?
- איטרטיביות, ולמה תשומת לב היא אכן כל **מה שאנו זוקקים לו?**
- כיצד טרנספורמרים פתרו את המגבלת העיקרית של הארכיטקטורות שקדמו להן, העיבוד הטורי של הדאטा?

אנו נבנה את המאמר בצורה של בובשקה (או מטוריישקה ברוסית תקנית). המאמר יבנה סדרה של קופסאות שחורות, שכל אחת מהן תזכה לתת-פרק שבו נסביר מה היא תומנת בתוכה. צורת העבודה תהיה "מלמעלה-למטה" (down > top), בכל שלב אנו נחשוף אבן בניין נוספת בארכיטקטורה, וחלקים אחרים שלה נוטיר לקופסאות שחורות. אף על פי שמבנה המודול פשוט באופן יחסית, הוא סובל מתadmית של נושא מורכב ומאתגר להבנה. זאת מכיוון שהוא בנוי ממספר חלקים המבוססים על עקרונות מופשיים כך שגם הם אינם מובנים לעומק, קשה לחברם יחד לכדי רעיון כולל. מסיבה זו החלחנו לבנות את ההסבר, כך שבכל שלב נוכל להתמקד בReLUוניות ספציפיים האחוריים אבן בניין מסוימת, ואחרות להשאיר בתור "קופסא שחורה" שניית יהיה להגדיר את הקלט והפלט שלה, מבלתי לצורך להסביר על אופן הפעולה שלה. אנו מקווים שדבר זה יקל על הקורא.

- **קופסא ראשונה:** בחלק זה כל המודול הוא קופסה שחורה, מלבד חלקו השיבוץ והקיוד תליי מיקום של הטוקנים (token embedding and positional encoding) שזכרים לפרקם משליהם. אנו נדבר על הקלט והפלט של המודול. נסביר כיצד הקלט מארגן כך שנitin יהיה לעבדו באופן מקביל, וכייזה הפלט נבנה באופן אוטורגריסיבי. בנוסף, נדבר על "שרות החייל" של הקלט, החל מקטע טקסט בשפה טבעית דורך קיוד למרחב חבוי והפיכתו בחזרה לטקסט בשפה טבעית במצב הרشت.
- **קופסא שנייה:** בחלק זה נתעמק במבנה הפנימי של המזודד והפענה. נסביר מהו התפקיד שלהם, וכייזה הם עובדים יחד. כן נשאיר את החלקים הפנימיים של המזודד והפענה קופסאות שחורות (מנגנון תשומת הלב, שכבת ה-feed-forward וכו').
- **קופסאה שלישית:** בחלק זה נספק הסבר מקיים על הבלוקים המרכיבים את המזודד והפענה. נರחיב על מבנה הארכיטקטורה של כל אחד מחלקים אלו ונפתח למשה את כל הקופסאות השחורות שנוטרו לנו מהשלב האחרון. נציג לראשונה את הרעיון המרכזי שהרשות בינויו סבירו, והוא ייצוג הקלט בתור מפתח, שאלתה, וערך (key, query, value) שעליהם מבוסס מנגנון תשומת הלב של טרנספורמרים.

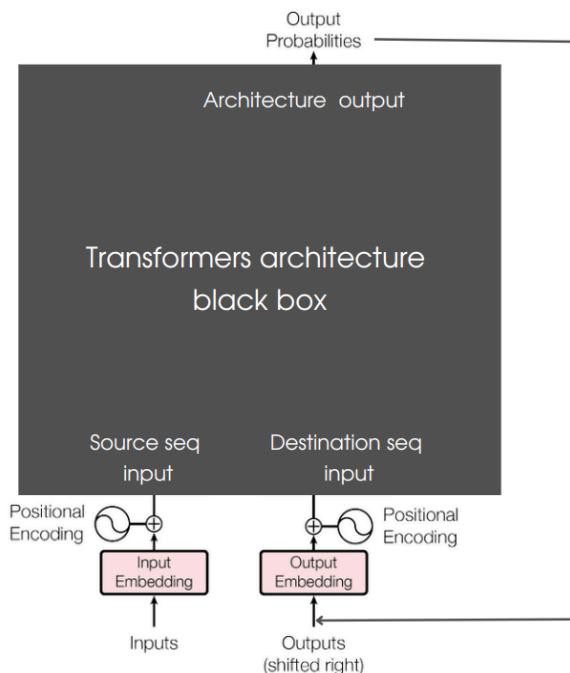
במאמר זה אנו נדון בנושאים הבאים:



- תשומת לב עצמית ותשומת לב מוצלבת.
- פלטי היבנים של המקודד והמפענח ובפרט מהו הפלט אחרי שכבת תשומת הלב, והפלט הסופי של המקודד.
- מהו החלטת מנגנון תשומת הלב המוצלבת ומנגנון תשומת הלב העצמית.
- מהו מנגנון תשומת הלב הרב ראשית.
- מדוע נדרש שכבת הנורמליזציה (layer normalization) ?
- נראה כיצד skip-connections ושכבות feed forward מ滿לאות תפקיד אינטגרלי בארQUITטורה איך הם משפיעים על ביצועי המודל.



קופסה ראשונה: הארכיטקטורה, מבט מלמעלה



איור 1 - ייצוג הארכיטקטורה כקופסה שחורה

אנו נחלק פרק זה לשני חלקים עיקריים. הנושא הראשון עוסק בקלט של המודל ויכיל בתוכו את התהילה שעובר הקלט מקלט טקסט בשפה טבעית ועד לשלב שנייה להזינו לתוכה הרשת לצורכי האימון. בחלק השני אנו נעסק בפלט של המודל עד להפיכתו למילה בשפה טבעית שוב.

חלק ראשון: קלט הרשת

כasher אנו מדברים על מודלים לעיבוד שפה טבעית, אנו מנדרים את המילון המשמש את מודל. מילון זה מכיל את כל המילים שהמודל מכיר, כאשר לא ניתן יהיה להשתמש במילים שלא נמצאות במילון זה או שלא ניתן להרכיבן ממילים אחרות המוכרות על ידי המילון.

לפנינו שנותאר את מבנה הארכיטקטורה, נרחיב על תהיליך העיבוד של הקלט שבסומו הוא מיוצג במרחב וקטורי חדש. תהיליך זה כולל שני שלבים עיקריים:

1. טוקניזציה של הטקסט.
2. שיבוץ הטוקנים שהתקבלו במרחב וקטורי.

נדגין כיצד שני שלבים אלו באים לידי ביטוי באמצעות הקלט הבא:

"The stars danced across the velvet sky, painting the night with their celestial beauty."

טוקניזציה (tokenization):

השלב הראשון בעיבוד הקלט הוא טוקניזציה. פעולה זו מחלקת את הקלט למיללים (בתוך קטגוריות זו אנו כוללים גם סימני פיסוק, וחלקו מילים). ישנים טוקניזירים שונים שיחלקו את אותה המילה באופן שונה. לדוגמה, המילה 't'chop' יכולה להתחלק באופןים הבאים: ['t','chop'] או ['t','#','chop'] (הסביר מפורט על הטוקניזרים השונים ניתן למצוא [כאן](#)). לאחר הפעלת טוקניזציה בסיסית (כל TOKEN הינו מילה או סימן פסוק) על המשפט שנחננו כדוגמה, נקבל את הפלט הבא:

```
tokens = ["The", "stars", "danced", "across", "the", "velvet", "sky", "", "painting", "the", "night", "with", "their", "celestial", "beauty", "."]
```

מודלי שפה מודרניים (למשל אלו מ- "סדרות" BERT או GPTs לMINIIM) משתמשים בשיטות טוקניזציה מתקדמות, בהם חלק מהטוקנים הינם תתי-מילים לא מילים שלמות. העיקרון המוביל בשיטות טוקניזציה אלו (BytePair Encoding ו-WordPiece) הוא הקיונית טוקנים למילים או תתי-מילים השכיחים ביותר בסט הנתונים (dataset) עליו הוא מאומן (בדרכו כלל סט זה הינו מנון ועוצם בגודלו). בנוסף, BERT מגדיר שני טוקנים מיוחדים לתוצר הטוקניזציה: SEP,CLS. התפקיד של SEP הוא להפריד בין משפטים. TOKEN - CLS משמש ליצור ייצוג וקטורי של מקטעי טקסט (הסביר עמוק יותר ניתן למצוא [בלינק](#)). נציג כי מודלים אחרים מגדירים טוקנים מיוחדים בצורה אחרת.

השלב הבא בפעולות הטוקניזציה הוא מתן מספר מזהה ייחודי לכל TOKEN שקיבלנו (כלומר כל TOKEN מקבל מספר סדרי). במידה וישנו מילים במשפט שלא נמצאות במילון שלנו, הן יחולקו לתתי טוקנים, עד שכל מילה בקלט מקבל מספר מזהה ייחודי (או קבוצת מספרים במידה ופעולות הטוקניזציה חילקה אותה במספר טוקנים). בדוגמה שלנו, הפלט של שכבה זו עבר קטע טקסט מסוים יכול להיות:

```
token_ids = [101, 1996, 3340, 5228, 101, 1996, 16441, 3712, 1010, 101, 1996, 2305, 2007, 2037, 12631, 5053, 1012, 102]
```

השלב השני בעיבוד הקלט הוא שיבוץ טוקנים נלמד (Learnable Token Embedding) במרחב וקטורי. המינדים השונים בוקטור מיצגים מאפיינים סמנטיים ותחביריים שונים שלו. כתוצאה לכך ניתן למדל את מערכת היחסים בין הטוקנים באמצעות פעולות אРИתמטיות. בארcitקטורות שקדמו לטרנספורמרים, נעשה שימוש במודלים שאומנו במיוחד לבנות שיבוץ זה (לדוגמה [word2vec](#)). לעומת זאת, בארcitקטורות הטרנספורמרים שיבוץ הטוקנים משולב בתהליכי האימון.

cut נתאר את שני השלבים האחרונים שהקלט עובר עד שנitin להזינו למקודד ולמפענה: תהליך השיבוץ של הטוקנים וקידוד תליי מיקום (positional encoding). אנו נסביר את החלקים שאינם מוסתרים על ידי קויפסא שחורה באיר 1 כollower Input Embedding - (Positional Encoding).



шибוץ הטוקנים (Tokens Embedding):

מוצא שכבת השיבוץ נתן על ידי המשוואה הבאה:

$$\text{Embedded vector} = E[i] * \sqrt{d_{\text{model}}}$$

משוואת 1 - שיבוץ הטוקנים למרחב נסתר

- X - מטריצה הבנויה מ [one hot encoded vectors](#) שנודלה $A \times d$, כאשר d מסמן את מספר הטוקנים במלון. גודל המטריצה X נובע מכך שנודלו של כל וקטור ה-one-hot הוא d והוא A וישנם A וקטורים כאלה.
- i - אינדקס הטוקן המשובץ.
- d_{model} - מימד מרחב הקידוד (шибוץ).
- E - מטריצת שיבוץ נלמדת שנודלה $d \times d$, מתפקדת $C \cdot D \cdot U$ המכילה את השיבוצים של כל הטוקנים, כלומר אנו משבצים A טוקנים למרחב בגודל d .

בפועל, המכפלה $[i] * E$ היא נשאה למיקום i בטבלה E . צורתו ייצוג זו היא עיליה מבחינה חשיבותית, כך ניתן לנצל את הטבלה לחישוב שיבוץ הטוקנים של הקלט בו זמנית (במקביל). כתובבי המאמר "[Attention is All You Need](#)" לא מצינים מהי הסיבה להכפלת מוצא השכבה הלינארית בגודל $\sqrt{d_{\text{model}}}$. אולם, ישנן העשויות שהדבר אמרור למנוע את דעיכת הנרדיאנטים שכן השיבוץ עלול ליצור ערכיים גבויים מדי שיבולו לרווחה של פונקציית softmax במנגנון תשומת הלב (אשר נרחב עלייו בהמשך). העשרה אחרת טעונה כי המטריה הינה למנוע מערך קידוד תלוי המיקום (יחסיבו בהמשך) להיות דומיננטי. אולם, אלו הן רק העשויות ויש להתייחס אליהם בהתאם.

cut נשלת השאלה מדוע אנו משקיעים בקידוד עצמאי של הקלט ולא משתמשים ברשות שאותנו מראש (כגון word2vec) במילוי עבור מטרה זו?

הדבר נובע משלוש סיבות מרכזיות:

1. במשימות הרכוכות ניתוח שפה כל מודל מגדר מהו שיטת השיבוץ והמלון המשמשים לצרכי הטוקניזציה של הטקסט. מתוך כך נובעת ההבנה מדוע לא ניתן להשתמש במודל שיבוץ טוקנים שאומן עם נשאה אחרת לטוקניזציה (כלומר אי אפשר להשתמש ב-word2vec לטוקניזציה של טרנספורמרים המשמשים בשיטת טוקניזציה כמו wordpiece או byte pair encoding).
2. מכיוון שניתן להשתמש במיליה בהקשרים שונים או בכפוף שימוש, علينا למצוא ייצוג ממוצע עבור הטוקנים המייצגים את המיליה במרחב וקטורי. דוגמא לכך יכולה להיות המיליה bat, שיכולה להופיע בהקשר של משחק כדור (bat = מחבט) או בהקשר של זואולוגיה (bat = עטלף). מכיוון שפה טבעית הינה בעיה מורכבת לאפויו, נדרש כמות עצומה של DATA כדי לבנות מודלים שימושיים ללמידה את מנון הקשרים החביבים בתוכה. יציגו טוקנים המופקים באמצעות מודלים כגון word2vec שאומנו על כמות DATA קטנה הרבה יותר מאשר של הדאטאסתים העצומים שימושיים בהם לאמון טרנספורמרים, אינם מסוגלים להכיל את כל הקשרים המורכבים בין הטוקנים.
3. כפי שאמרנו בעבר, מנגנון תשומת הלב לומד את חשיבותה של יחידת קלט אחת אל מול יחידת קלט אחרת כתלות בערכה. וכך, הדבר הגיוני ביותר לעשות הוא לחת לרשף ללמידה את השיבוץ שהכי מתאים לה באופן עצמאי.



קידוד תלוי מיקום (Positional Encoding)

החדשנות של ארכיטקטורת הטרנספורמרים נבעה מהניסיונות לענות על השאלה הבאה: **כיצד ניתן לוותר על הפעלה האיטרטטיבית של הרשות בעיבוד סדרות?** הפתרון הגנוני בויתר הוא ניתוח כל הקלט במקביל. אבל כיצד ניתן לבצע זאת? אחת הדרכים לעשות זאת היא ייצוג הקלט כסט (מבנה נתונים שבו הסדר אינו רלוונטי) דבר המאפשר לנו להזין אותו כמקשה אחת. אולם, הקלט למודל הינו סדרתי. אז כיצד ניתן להתגבר על סתריה זו? הפתרון הוא לספק למודל מידע על הסדר של יחידות הדאטה, ככלומר, מידע על מיקום המילה בטקסט שיאפשר למודל להבין את המשמעות של מרחק בין מילים. במידה ולא נספק למודל מידע זה מנngenנו תשומת הלב שארכיטקטורת הטרנספורמרים מבוססת עליו עלול לנתח באופן זהה את שני המשפטים הבאים:

"Tom bit a dog." | "A dog bit Tom."

המידע שאנו מוסיפים לקלט מאפשר למודל ללמוד את המיקום של כל יחידת DATA בטקסט ואת מרחקה היחסית מכל יחידת DATA אחרת. הוסף מידע זה מאפשר לרשות ובפרט לפונקציית תשומת הלב להתחשב במיקומם של החלקים השונים של הקלט כאשר היא שוקלת את חשיבותה של מילה בклט ביחס למילה אחרת. האופן שבו הרשות מוסיפה מידע זה נקרא **קידוד תלוי מיקום** (Positional encoding).

קידוד תלוי מיקום בטרנספורמרים

כיצד מבצעים קידוד תלוי מיקום? נזכיר שמטרתנו היא להעניק למודל יכולת למדוד מה המרחק בין יחידות הקלט. אילוץ נוסף שאנו מעוניינים בו הוא שהקידוד יהיה ייחודי עבור כל מיקום בклט, אחרת לא ניתן היה להבחין בין מילים במיקומים שונים. הפתרון הנאיבי הוא להשתמש ב- "one-hot encoding". בקידוד זה אנו יוצרים וקטור שאורךו שווה לאורך סדרת הקלט עבור כל טוקן, ומאתחלים את ערכיו לאפסים, במקומות בו מופיעה המילה אנו מציבים 1.

ניקח לדוגמא את המשפט הבא:

"The quick brown fox jumps over the lazy dog near the blue river."

דוגמא זו מייצגת את הבעיה שקידוד זה מעמיד בפנינו, והוא ש-*one-hot encoding* (equidistant), כלומר, כל וקטור מרוחק מכל וקטור אחר ב- $\sqrt{2}$. בדוגמה שלנו המילה "the" מופיעה 3 פעמים, ולכן הרשות תתקשה לשין כל "the" למילה המקוררת אליה ("dog"). מתוך כך עולה השאלה: **כיצד ניתן לקודד מילה כך שמקומה במשפט יקרה לה ערך ייחודי ובנוסף שהמרחק בין הקידודים ישקף את מרחק בין מיקומי המילים בклט?** במקרים אחרים אנו רוצחים קידוד המקרב את ערכם של כל זוג וקטורים המייצגים טוקנים קרובים (mbhinit מרחוקם בклט) ומרחיק כל שני וקטורים המייצגים מילים רחוקות.

או מהו הפתרון?

קידוד תלוי מיקום באמצעות פונקציות מחזוריות:

$$\begin{aligned} \text{Positional Encoding}(pos, 2i) &= \sin(pos/(10000^{2i/d_{\text{model}}})) \\ \text{Positional Encoding}(pos, 2i + 1) &= \cos(pos/(10000^{2i/d_{\text{model}}})) \end{aligned}$$

משווהה 2 - קידוד תלוי מיקום



כאשר:

- pos - מיקום הטוֹקָן בסדרה המקורית.
- i - האינדקס בתוך המרחב השיבוץ d_{model} כאשר מתקיים: $i \in d_{model}$.

כעת נסביר את המשוואה עבור $d_{model} = 512$ (כפי שהוא מוגדר במאמר המקורי). אנו מחשבים את הוקטור עבור מיקום (pos) של כל טוֹקָן בסדרה, כאשר החישוב הוא לפי פונקציית הסינוס עבור מימדים זוגיים של וקטורי הקיזוד ופונקציית הקוסינוס למימיים האי הזוגיים. מכיוון שערכיהם אלו קבועים לאורך כל השימוש במודל אנו מחשבים אותם באתחול ומשתמשים בהם באמצעות LUT (Look Up Table). לבסוף אנו מחברים את הוקטור הקיזוד שהתקבל עם וקטור השיבוץ.

חוקרינו לא מסבירים באופן מפורש מדוע הם מחברים את וקטור הקיזוד עם וקטור השיבוץ, ולא משרשים אותו אילו.

אז מדוע השימוש בפונקציות מחזוריות מספק את הדרישות שניסחנו בתחילת הפרק הקודם?

נזכיר את הדרישות שאנו מבקשים עבור וקטור הקיזוד ונראה כיצד הפונקציות המחזוריות מספקות מענה לכל אחת מהן.

• **למידת מיקום יחסי (relative position):**

כפי שתיארנו בפתיחתו, על המודל ללמידה כיצד לאמוד מרחק. אולם, אנו לא מעוניינים ללמידה את המרחק כיחידה אבסולוטית (כלומר מרחק של טוֹקָן השלישי מהטוֹקָן הראשון בסדרת הקטלט), אלא את **המרחק בין מיקומי הטוֹקָנים בסדרה**. המשמעות הינה, שאנו מקבאים טוֹקָן מסוים בסדרה, וմבקשים לammed את המרחק ביןו לכל הטוֹקָנים בקהלט. לדוגמא, אם המיקום האבסולוטי של הטוֹקָן הוא 67, ואנו רצאים לammed את חישובתו ביחס לטוֹקָן במיקום ה-47 או המרחק היחסי ביןם יהיה 20.

במילים אחרות, אנו בוחנים את המרחקים בין $1 \leq pos < pos + k \leq 2$ מיקומים:

בפשטות, אנו מספקים לרשף את ייצוג של $1 \leq pos < pos + k \leq 2$ מיקומים מהמודל "להפיך" ייצוג של המרחק k בלבד. וכך בינהם. כיצד פונקציות טריגונומטריות עוזרות לנו במקרה זה? התשובה לכך טמונה בעובדה שפונקציות אלו מקיימות את הזהויות הטריגונומטריות הבאות:

$$(1) \sin(pos + k) = \sin(pos) * \cos(k) + \sin(k) * \cos(pos)$$

$$(2) \cos(pos + k) = \cos(pos) * \cos(k) - \sin(pos) * \sin(k)$$

משוויה 3 - זהויות טריגונומטריות עבור סכום בתוך פונקציית סינוס וקוסינוס

ולכן, חישוב המרחק בין שיבוצי טוֹקָנים מרוחקים אחד מהשני ב- k טוֹקָנים באמצעות מנגנון תשומת הלב, יOLID ביטויים התלויים במיקום האבסולוטי (pos) והמרחק היחסי (k).

• **קיזוד ייחודי לכל וקטור embedding unique):**

השאלה הרשונה ששאלתי, היא מדוע אנו משלבים את הפונקציות הטריגונומטריות \cos ו- \sin ? (כלומר נודיע המודל לא משתמש רק בפונקציית חוס או רק בפונקציית \cos). המאמר לא מספק תשובה חד משמעית עבור שאלה זו, אולם אחת ההשערותינה היא שילוב של שתי פונקציות אלו יוצר ייצוג עשיר יותר עבור המידע המיקומי מאשר שימוש בפונקציה יחידה.



השאלה השנייה שנסألת הינה מדוע המחברים בחרו לשימוש דואק'ה בביטוי:

$$\text{משוואה 4 - הביטוי המעריצי בתוך פונקציית הקידוד}$$

$$pos^{2i/d_{\text{model}}} \quad (\sin(pos * (2i/d_{\text{model}})) \quad \cos(pos * (2i/d_{\text{model}})))$$

על מנת להבין מדוע אכן משתמשים דואק'ה בביטוי זה علينا להבין את מרכיביו ואת התפקיד שלהם:

- **זוק** - הדרישה הבסיסית של קידוד תלוי מיקום היא ליצור וקטור ייחודי עבור כל טוקן בקלט. וכך, הדבר הגינוי ביוטר לעשות הוא להשתמש במיקום שלו בקלט (זוק), המגדיר אותו באופן ייחודי.

- **לקבוע** מה יהיה הערך שיקבל כל מינד.
- **הפתרון הנאייבי** הוא לשימוש בערכו של $\sin(pos)$ או $\cos(pos)$ עבור כל המרכיבים של וקטורי הקידוד. אולם, הבעיה בכךה זו הינה קיום מחזריות במרחקים בין וקטורים. כלומר, עבור ערך x כלשהו, התלוי במורכבות הפונקציה, המרחקים בין וקטורים המקיים את מיקומם של טוקנים הנמצאים במרחקים x ו- $x + 2m$ עשויים להיות כמעט שווים. זה מצב לא רצוי מכיוון שהקידוד אמור לשזקף את המרחק בין הטוקנים.

במידה והדבר לא מתקיים המודול יתקשה להפיק מוקטורים אלו את מלא המידע על המרחק בין הטוקנים בסדרת הקלט. ככל שנשתמש בפונקציה מורכבת יותר, כך נקטין את הסיכוי למחרזריות שכך (ערך של x גדל ככל שהפונקציה מורכבת יותר).

הפתרון המוצע הוא לחת לכל מינד בוקטור ערך שונה. דבר זה מתבצע באמצעות מכפלה של זוק בימייד וחלוקתו ב- d_{model} .

כלומר אנו יוצרים את המערך הבא:

$$[pos/d_{\text{models}}, pos * 2/d_{\text{model}}, \dots, pos * i/d_{\text{model}}, \dots, pos] \quad \forall i \in [d_{\text{model}}]$$

- הוספת בסיס החזקה של 10000 יוצרת מורכבות שאינה מאפשרת חזרתיות גם כאשר מספר הטוקנים גדול בסדרי גודל של עשרות אלפיים.

על מנת להוכיח את התזה שהצענו כאן, נבחן מה הייתה יכולה להיות האלטרנטיבה הפешטה ביותר וכייזה הייתה נראית תוצאה.

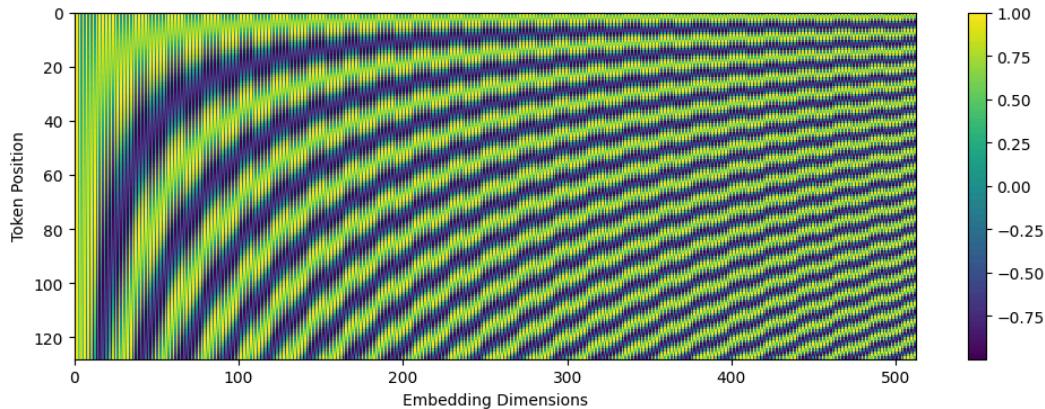
למשל עבור קידוד תלוי מיקום השווה ל:

$$\begin{aligned} \text{Positional Encoding}(pos, 2i) &= \sin(pos * (2i/d_{\text{model}})) \\ \text{Positional Encoding}(pos, 2i + 1) &= \cos(pos * (2i/d_{\text{model}})) \end{aligned}$$

משוואה 5 - קידוד תלוי מיקום ללא המרכיב המעריצי

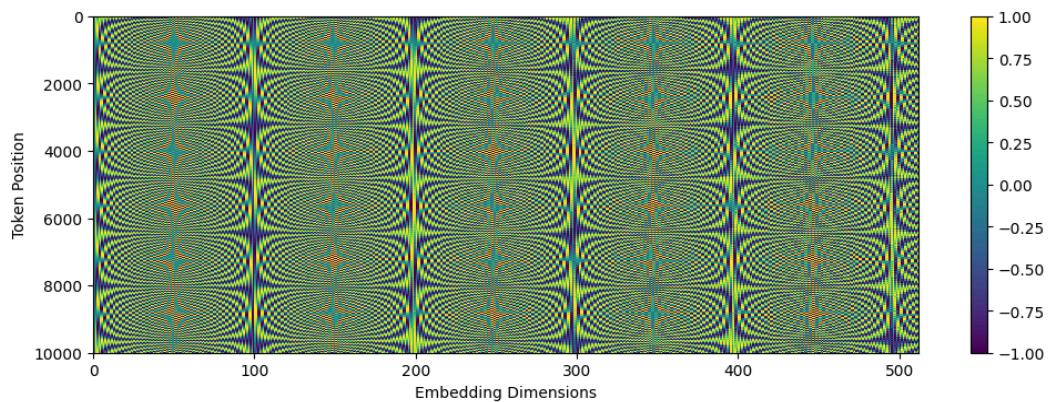
כאשר $512 = d_{\text{model}}$ ו- מספר הטוקנים הוא 128, נקבל את הנגרף הבא:





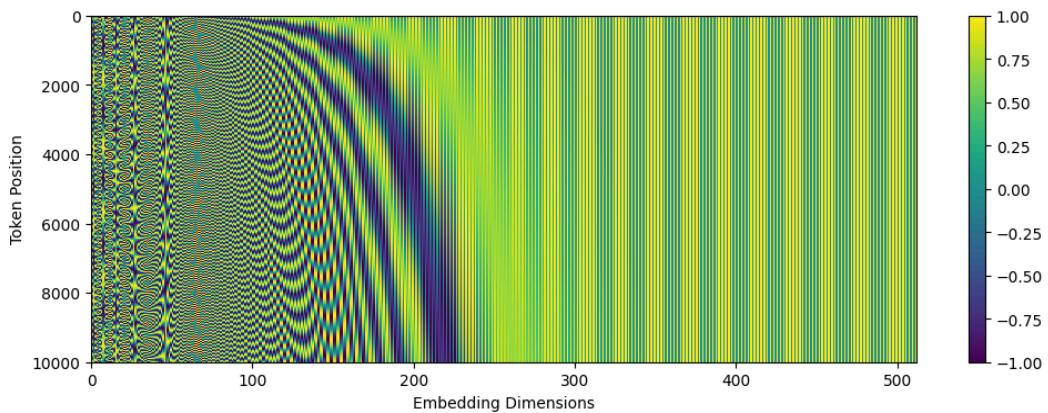
איור 2 - קידוד תלוי מיקום אלטרנטיבי, עבור 128 טוקנים

לכארה לא נראהיה בעיה גלויה לעין. אולם, אם נגדיל את מספר הtokנים ל-10,000 כאשר $d_{model} = 512$ (אורך הקשר ריאלי עבור טרנספורמרים בני ימינו) נקבל את הנגרף הבא:



איור 3 - קידוד תלוי מיקום אלטרנטיבי, עבור 10,000 טוקנים

כפי שניתן לראות, כאשר מספר הtokנים בסדרה, נדול מ- d_{model} נוצרת חוזרתיות בדףו קידוד תלוי המיקום. למחרת שניתן לפתור את זה באמצעות הנדלות ערך של d_{model} , זה לא פתרון ריאלי. כתובבי המאמר מציעים את הקידוד המעריצי. הסיבה שהקידוד המעריצי עובד בצורה כל כך טובה, נובעת מהעובדדה שהוא מדכא חוזרתיות עבור כל גודל סדרת קלט בכניסה, כפי שניתן לראות באירור 4 עבור סדרה בעלת 10,000 טוקנים.



איור 4 - קידוד תלוי מיקום מערכי עבור 10,000 טוקנים

שיטות נוספות לביצוע קידוד תלוי מיקום:

קידוד תלוי מיקום אינו מחייב לאופן בו הוא מוצג במאמר המקורי, וישנן שיטות נוספות. אנו לא נסקור שיטות אלו במאמר זה (אך ניתן למצאו אותן ב[1], [2] ו-[3] או בסקירה עתידית שנבצע) אולם נתעכט על נקודה אחת: מדו"ע החוקרים משתמשים דווקא ביצוגים באמצעות פונקציות מהירות, ולא לומדים את הקידוד כחלק מתהליכי האימון. החוקרים מצינים במאמר המקורי שהם ביצעו ניסויים עם קידוד נלמד, וראו כי התוצאות לא השתנו לעומת קידוד קבוע מראש. מכיוון שאין הבדל בביטויים אנו נעדר לבער קידוד ידוע מראש ולא נלמד, מכיוון שהדבר עיל יותר מבחינה חישובית.

חלק שני : פלט הרשות

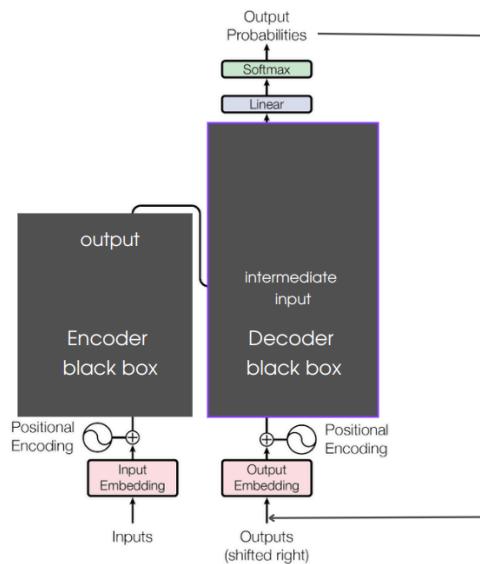
פלט המודל הוא תוצר של שכבת softmax על פני כל מיליון הטוקנים שאנו משתמשים בו, כאשר הרשות מייצרת בכל הפעלה את הטוקן החזוי הבא ביחס לפלט. במהלך האימון אנו בונים את הפלט של המודל טוקן אחריו טוקן עד שנ дол הסדרה שווה לנ Dol הסדרה המקורית, ואז מבצעים את פונקציית loss ביחס לפט הרשות של המודל.

במהלך השימוש במודל (inference) אנו לא יודעים מה אמור להיות אורך משפט המוצא. לכן علينا

להפסיק את הפעלת המודל בנזודה כלשהי. ישנן שיטות שונות לקביעת נזודה העצירה:

- קביעת סף למספר הטוקנים שהמודול יוכל ליצור עבור קלט מסוים.
- שימוש בטוקן מיוחד הנקרא EOS שתפקידו להציג כי המודול סיים יצירת המשפט.
- בחינת וודאות (confidence) של המודל בטוקן אותו הוא יצר, והפסקת הריצה מידה וערץ זה לא חוצה סף מסוים.

קופסא שנייה: מילודד ומפענה



איור 5 - המילודד והמפענה כ קופסאות שחורות

המבנה הכללי:

ארQUITטורת הטרנספורמרים שהוצגה במאמר המקורי (Attention is All You Need) יועדה להמרת סדרת קלט (source) לסדרת פלט אחרת (target). הארכיטקטורה, בדומה לארQUITטורות שקדמו לה, בנוייה כמיילודד ומפענה (ישנם טרנספורמרים, הבנויים מיילודד בלבד או מפענה בלבד וכן נדונם בהם באחד החלקים הבאים של סדרת מאמרים זו).

חשוב להבין כי במשימת תרגום שתי הסדרות המתקבלות בכניסת המיילודד והמפענה, מקשרות אחת לשניה, אך כל אחת מהן מתוארת באמצעות ייצוג שונה (בדמות שפה). לדוגמא, משפטים בעברית ובאנגלית המתארים סיוף על יד משחק בצד, מכילים את אותו התוכן, אך שונים באופן שבו הם מביעים אותו. **הבעיה הניצבת בפנינו במשימה זו היא להבין כיצד ממיררים מיצוג אחד לייצוג אחר, במיוחד במקרים חוקי המיפוי מורכבים.**

זהו תמצית השימוש בטרנספורמרים כפי שהוא מופיע במאמר המקורי שיועד למשימות תרגום וסיקום טקסט. בסופו של דבר, תכילת המודל **שהוצע במאמר** הינה ללימוד כיצד לבצע את הטרנספורמציה (המרה) בין שני מרחבי ייצוג שונים המכילים את אותו המידע. יש לזכור שכוחם משתמשים בטרנספורמרים למשימות מנוגנות שבהן אנו לא לומדים טרנספורמציה בין מרחבי ייצוג של "אותו הדטה" ולכן פסקה זו תקיפה רק למאמר שאנו מנתחים.

מטרת המודל הינה לחזות את ייצוג הקלט למרחב היעד (השפה שמתורגמים אליה). על מנת לעשות זאת, המפענה מרכיב את משפט היעד טוקן אחרי טוקן. אנו מתחילהים את חיזוי המשפט כסדרה ריקה, ובכל שלב של הפעלת המודל אנו מוסיפים טוקן נוסף שחזינו להיות חלק מסדרת הקלט של המפענה. על מנת ללמידה כיצד לעשות זאת, אנו מזינים למילודד את סדרת המוקור (לדוגמא, הטקסט שאנו מעוניינים לתרגם) כדי שיבנה ייצוג המכיל את הקשרים בטקסט למרחב וקטורי המשותף לו ולמפענה (בהמשך נסביר מהו מרחב זה וכיצד בונים אותו). המפענה נעזר במידע בניתו הקלט שלו (שנבנה טוקן אחרי טוקן)

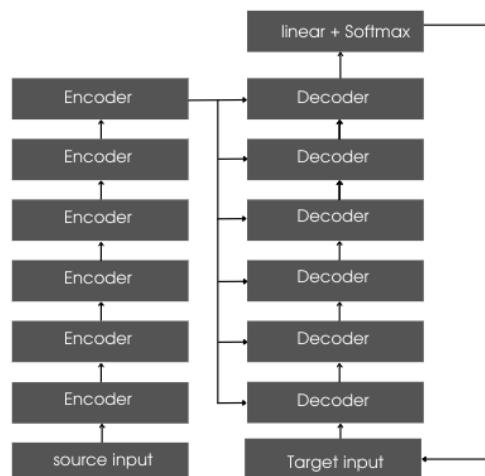
וביצירת הפלט שלו. אם נחזור לדוגמא שפתחנו אותה, המקודד מעביר למפענה את **תוקן הרעיון** שהמשפטים מייצגים, ללא תלות במרקבי הקלט (הสภาพ השונות).

בדומה לארכיטקטורות קודמות שסקרנו, גם ארכיטקטורות הטרנספורמרים משתמשות במנגנון תשומת לב. אולם בשונה מהן, מנגנון זה הינו לב ליבה של הארכיטקטורה ולא מתפרק כמרכיב עוז שתקיים לבנות ייצוג ביןים למידע המופק מהמקודד או מהמפענה. מכאן מניע גם שם המאמר, כי תשומת לב היא הדבר היחיד שהמודול צריך.

המקודד והמפענה בנויים מ-"לבנים" (blocks) שכל אחד מהם נקרא **בלוק טרנספורמר** או בקיצור **טרנספורמר**. בלוק טרנספורמר של המקודד והמפענה שונים אחד מהשני, כאשר המרכיב החשוב ביותר הוא מנגנון תשומת הלב הזהה בשניהם (נרחיב על כך בפרק הבא).

חשוב להבין כי מנגנון תשומת הלב עובד באותה הדרך גם במקודד וגם במפענה. אולם **האוף בו המידע מהמקודד מזון למנגנון זה שהוא מהאוף בו הוא מזון במפענה**. הקלט של המפענה נבנה באופן אוטו-רגרסיבי, כלומר הפלט שלו הופך להיות חלק מהקלט הבא שלו. עקב כך אנו משתמשים בתשומת לב נוספת, ה"מסתירה" מנגנון תשומת הלב את המידע על הטוקנים הבאים בסדרה אחרי הטוקן הנוכחי.

המאמר מצין כי הבלוקים של המפענה והמקודד מוערמים (stacked) אחד על השני. איור 6 מציג כיצד חיבור זה מתבצע בפועל.



איור 6 - מערום (stacking) של בלוקי הטרנספורמר

תפקיד המקודד והמשמעות

כפי שנition לראות באIOR 6, כל בלוקי המקודד מחוברים אחד לשני בטור, כאשר הפלט של כל בלוק מזין את הבלוק אחריו. בינווד למקודד, המשמעות מזון בפלט המ קודד המתקבל מהבלוק האחרון שלו, בתוספת לפט מלוק המשמעה הקודם. הסיבה שהארQUITקטורה בנייה כך, טמונה בעובדה שבлок המ קודד האחרון מפיק את ייצוג המידע המופיע ביוטר ומכל את האינפורמציה המלאה ביוטר עליו. לכן, הזנתו תשיע עלמשמעות בהבנת הקשר של הפלט שלו.

קלט ופלט:

cut נגיד מהו הפלט והקלט לחלקים השונים של הטרנספורמר. כאשר אנו מדברים על הקלט של המקודד או המשמעה אנו תמיד נתייחס לסדרה שהתקבלה בכניסתם.

cut נרחב על הקלט והפלט של המקודד והמשמעות:

המקודד:

- **קלט המקודד (encoder input)** - סדרת הקלט המקורי (source) אותה אנו רצים לעבד.
- **פלט המקודד (encoder output)** - ייצוג וקטור של הפלט המופיע על ידי המקודד, כולם הפלט של בלוק הטרנספורמר האחרון שלו. פלט זה מכיל מידע על הקשרים הקיימים בבלוק.

המשמעות:

- **קלט המשמעה (decoder input)** - סדרת היעד (target) אותה אנו לומדים לחזות בצורה אוטורגרטיבית (טוקן לאחר טוקן). חשוב להבין כי חילך מסדרת היעד מוסף (חילים שבאים לאחר הטוקן הנחוצה מוסתרים על מנת שהרשת לא תשתמש במידע זה). כאמור **הקלט הנוסף** של המשמעה, המתקבל מהמקודד, הינו פלט הבלוק האחרון שלו.
- **פלט המשמעה (decoder output)** - קלט לשכבה לינארית שמרתה לחשב התפלגות של הטוקן הבא בסדרת הפלט הסופית (כגון טקסט). כמו במקודד, הפלט של המשמעה נוצר על ידי בלוק הטרנספורמר האחרון שלו.

עקרונות הקלט/פלט של מקודד-משמעות

המקודד מעבד את הקלט בכניסתו באופן מקבילי ומיציר את הפלט בפעולה יחידה (single forward pass) ללא צורך בהפעלה איטרטיבית. בשונה ממנו, המשמעה מייציר את הפלט שלו באופן אוטורגרטיבי. המשמעות של אוטורגרטיביות הינה יצירת פלט בהסתמך רק על יחידות הפלט שכבר נוצרו. כלומר, בכל איטרציה אנו מזינים לתוך הרשות ייחדות מידע נוספת שחוسبו באיטרציות קודמות קלט, או איברי סדרת היעד עצמה עד הטוקן הנחוצה (teacher forcing).

אוז מודיע לנו בונים את הקלט של המשמעה באופן אוטורגרטיבי? הסיבה לכך נובעת מכך שהמשמעות הוא החלק במערכת המפיק את משפט היעד, ומטרתו לחקות את תהליך יצירת המילה האנושית.

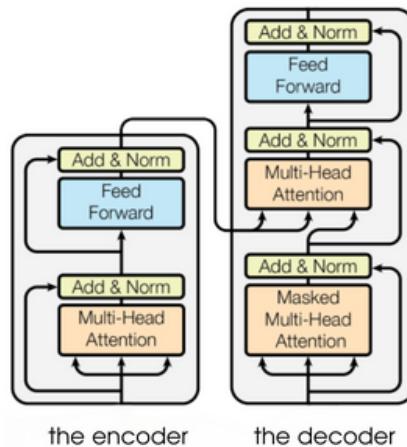
אולי ישנה דרך לבצע זאת. נבחן את האפשרויות העומדות לפניינו:



- **הראשונה** היא לא להשתמש במיסוך כלל. כלומר, נזין את כל סדרת היעד למפענה, ואת כל סדרת המקור למקודד. לאחר מכן נבצע הריצה בזוזת (single forward pass) למפענה ונשווה את פלט שהתקבל עם סדרת היעד. אפשרות זו מקבילה ללמידה פונקציית הזהות, אנחנו מקבלים את התוצאה הרצiosa (הרשות מוציאה פלט נכון) אבל הרשות לא לומדת דבר מכיוון שהוא תלמיד להעתיק את הקלט ללא שינוי.
- **השנייה** היא להזין למקודד את סדרת המקור, ואילו למפענה לא להזין את סדרת היעד כלל, ואת פלט המפענה להשוות מול כל סדרת היעד כמקרה אחד. פתרון זה לא יעבוד גם כן, לאחר לשפה טبيعית מבנה מורכב, והמפענה יתקשה להסתמך רק על הקלט המקורי, אלא יצטרך להיות מודע לחלקים הקודמים בסדרה שכבר יצר (או לקיבלם קודם).
- **האפשרות האחרונה ההגינונית ביותר** הינה לבנות את הקלט באופן אוטורגרסיבי. כלומר, בכל איטרציה אנו חוזים אף ורק את הטוקן הבא (שטרם נחזה) ומוסיפים אותו לקלט המפענה המשמש לחיזוי הטוקן הבא. לעיתים לאחר סיום חיזוי מילה (כלומר כאשר נחזה הטוקן "[SEP]" המסתמן מרוח בין מילים) המילה שנחצתה מוחלפת במילה מתאימה מסדרת היעד ([teacher forcing](#)). תהליך זה מחקה את אופן יצירת מילה חדשה על ידי בני אדם. אנו יכולים לחשב על מילה חדשה רק בהינתן מילים שנאמרו עד כה מכיוון שהמילה החדשה תלויות בהם.



קופסא שלישית: צוללים לתוך המktor וומפענה



איור 7 - המבנה הפנימי של המktor וומפענה

אבני הבניין המרכזיות של המktor וומפענה.

המktor וומפענה בנויים משלושה אבני בניין מרכזיות שבאמצעותן הארכיטקטורה מקבלת את עצמה היחודית.

- מנגנון תשומת הלב** - מנגנון תשומת הלב שהוצע לראשונה ב-[4] הכליל חישוב של וקטור דינמי המכיל את הקשיים החשובים ביותר בין המktor וומפענה בעת יצירתו של טוקני הפלט בעת יצירתו של טוקני הפלט (דומה לתשומת הלב המוצלבת בטרנספורמרים). [5] השתמש במנגנון תשומת הלב כדי לחשב את הקשיים בין חלקו הקלט השונים במרקוויד (דומה לתשומת הלב העצמית בטרנספורמרים).
- רענון דומה יושם גם כמנגנון תשומת הלב של טרנספורמרים שמשקל את עצמתה הקשר בין יצוני יחידות דאטה בתוך אותה הסדרה (תשומת לב עצמית), או את הקשר בין יצוני יחידות דאטה הנבננים על ידי המktor וומפענה (תשומת לב מוצלבת). אולם ההבדל העיקרי בין המנגנונים הוא שבטרנספורמרים תשומת הלב (העצמית) מחושבת במקביל עבור כל הטוקנים בתוך הקלט, ולכן אין צורך ביצירון. החידוש הנוסף הוא שימוש בשני המנגנונים אלו (עצמית ומוצלבת) יחד.
- רשת feed-forward** - שתי שכבות fully-connected (השכבה השנייה הינה לנארית ללא פונקציית אקטיבציה). לכארה אין טעם לדון בה, אולם יש לרשף זו תפקיך חשוב בבנייתו של תוצר מנגנון תשומת הלב. כאשר נדבר עליה נציג מאמרם שמראים כיצד השמטה חלק זה מובילה לירידה משמעותית בביצועי המודול.
- שכבות Add and Norm** - שכבה שתפקידה לבצע residual connection ונורמליזציה (normalization). מוצאת פונקציית תשומת הלב היא מטrice בגודל $d_{model} \times N$ כאשר N מייצג את מספר הטוקנים. שכבת זו מנורמלת כל מידע של הקלט (אנחנו מנורמים את מידע זה של כל וקטור) יציג של כל הטוקנים יחד - קלומר וקטור בגודל N "מנורמל" כל פעם).

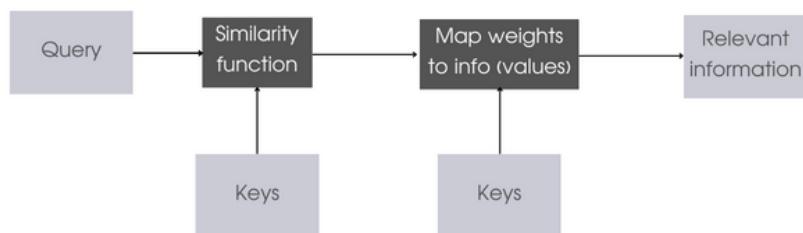
מנגנון תשומת הלב

ארQUITקטורת הטרנספורמרים שינתה את האופן בו אנו משתמשים עיבוד קלט סדרתי. עד לאותו הרגע, מנגנוני תשומת הלב השתמשו במידע המתקבל מהמצבים הפנימיים של רשות איטרטיבית, שאילצו הזנת קלט בודד (מילה או חלק מילה) בכל הפעלה. תשומת הלב חושבה במידע באמצעות בניה וקטור זיכרון דינמי, (שדרש הקצת זיכרון) על מנת ליצג את הקשרים במידע בכל איטרציה. שתי תוכנות אלו היו את עקב אכים של מודלים אלו. אולם, עם הופעת הטרנספורמים, התאפשר בפעם הראשונה עיבוד מקבילי של הקלט כמקרה אחד. כתוצאה לכך, לא היה צורך יותר בוקטור זיכרון עבור כל איטרציה, מה שאפשר בפעם הראשונה מידול איות של תליות ארכוטרktור טוווח בקלט. חשוב לציין שמנגנון תשומת הלב אינו חוסך במשאבי חישוב לעומת ארכיטקטורת שקדמו לו, סיבוכיות הריצה וסיבוכיות המיקום (time and space complexity) של מנגנון תשומת הלב עבור הריצה בזוזת הימנ – $O(d * n^2)$, אולם הוא אופן ניסוחו הינו יעיל יותר מבחינה ריעונית כי הוא בניין אינהרנטי לעיבוד מקבילי של כל המידע.

על מנת לבצע זאת הוגדרו 3 אובייקטים הנקרים שאלתיה, מפתח וערך שהרעיון מאחוריהם הושאל מאלגוריתמי אחזור נתונים (information retrieval) (ומנווּן חיפוש):

- **שאילתת (query)** היא הבקשה שאנו משתמשים בה על מנת לקבל את הערך/ים.
- **מפתח (key)** זהה האינפורמציה (יצוג) המקשרת אל הערך, ומשמשת ליהוי שלו.
- **ערך (value)** הוא הפריט אותו אנו מחפשים, השמור במסד הנתונים.

דוגמא הממחישה את הרעיון של שימוש בשאלתה, מפתח וערך, הינה חיפוש סרטונים ביוטיוב (YouTube). השאלה מייצגת את הטקסט שאנו כותבים בשורת החיפוש, המפתח מייצג את המידע על הסרטון השמור במסד הנתונים המאחסן את הסרטונים, (שם מלא של הסרטון, תיאור שלו, אורך, יוצרים וכו.) והערך הוא הסרטון בו אנו מעוניינים לצפות. אם נקבע את דוגמת חיפוש סרטונים להפעלת מנגנון תשומת הלב של טרנספורמרים, הסרטון שיבחר יהיה זה שהמפתח שלו יהיה בעל הקורלציה הנבואה ביותר לשאלתה שלנו.



איור 8 - המechanism קבלת ערך לפי שאלה וציון

דרך נוספת לחשב על אובייקטים אלו היא באמצעות גרעף לא מכון שלם וממושך. כלומר גרעף שבו כל קודקוד מחובר עם כל קודקוד אחר (כולל עצמו) ולקשת בין יש משקל מוגדר. בכל פעע אנו בוחרים קודקוד אחד ומקבעים אותו בתור השאלה, כל קודקוד בגרף (כולל הקודקוד שקבענו) מוגדר כمفצת. הערך מוגדר כמשקל הקשת המחברת בין קודקוד השאלה לקודקוד המפתח. תוצאת פונקציית תשומת הלב תהיה מכפלה פנימית של יצוג וקטורי של שני הקודקודים במשקל הקשת וביצוע softmax כדי לנормל את המשקל של קודקוד.

מדוע אנחנו משתמשים במפתח, ערך ושאלתה?

הסיבה שאנו משתמשים במפתח, ערך, ושאלתה במנגנון תשומת הלב נובעת מיעילותם ביצוג קשרים בין חלקיים שונים של הדטה הסדרתי. בוגנו למאמרנו הקודם, שב מנגנון תשומת הלב עשה שימוש במצבים הפנימיים של המקודד והמפענה, ארכיטקטורת הטרנספוררים לא מכילה מידע פנימי שמהן ולכן אנו זוקקים בדרך בה אנו יכולים לייצג קשרים אלו. מכיוון שמנגנון תשומת הלב של טרנספוררים מתפרק כמנגנון אוניברסלי גם עבור תשומת הלב העצמית ותשומת הלב המוצלבת, علينا להנידור כלים פשוטים שיוכולים לייצג את הרעיון עצמו.

הדוגמא שבה הצינו את מנגנון תשומת הלב באמצעות גרפ' שלם ממושך, מייצגת בדיק את הקשרים הללו. בדוגמה זו, לא משנה לנו מהican מגיעים השאלות או המפתחות, אנחנו עדין יכולים ליצור את הגרפ' השלם המחבר ביניהם. הייצוג של מנגנון תשומת הלב בצורה שכזו מאפשר לנו להשתמש בו גם בתשומת הלב המוצלבת בה המפתחות והערכים מגיעים מהמקודד בעוד שהשאלות מגיעות מהמפענה.

או כיצד משתמשים במנגנון זה בארכיטקטורה?

כפי שניתן לראות איירור 7, גם המקודד וגם המפענה משתמשים תחיליה במנגנון תשומת הלב עצמית, שבה אנו מאתרים את הקשרים בתוך הקטלט. מטרת מנגנון תשומת הלב העצמית הינה יצירת ייצוג תלוי הקשור עבור כל טוקן. לאחר מכן, המידע שחושב באמצעות מנגנון תשומת הלב העצמית של המקודד והמפענה משולב באמצעות תשומת לב מוצלבת. בשלב זה אנו מגדירים את השאלה שקיימת תשומת הלב של המפענה, ואת המפתחות והערכים אנו מגדירים להיות מוצא בлок המקודד האחרון.

הקלט למנגנון תשומת הלב של המקודד הינו הפלט של בлок הטרנספורמר שעורש לפניו (כאשר מדובר בבלוק הטרנספורמר הראשון) הקלט הוא סדרת הקלט שעברה טוקניותה והוספה קיזוז תלוי מיקום). לפני שקלט כלשהו מוכנס למנגנון תשומת הלב הוא עובר הטלה למרחב חדש באמצעות שלושה שכבות לינאריות (בפשטות, הכפלות במטריצה). במודול המוצע במאמר הטלות אלו הן שייצרות את המפתח, שאלתה והערכים (אולם אין זה מחייב, ניתן להנידור את המפתח השאלתה והערך כרצוננו). זהה נקודה חשובה, שכן אם **נשתמש בקלט בצורתו המקורית ללא הטלה זו, הרשות תאבד את הנגישות**.

(flexibility) שלה ללמידה קשרים אלו.

אבל מדוע זה נכון? אם נחזור לדוגמא של הגרפ' הממושך, נראה שלא רק משקל הקשת הוא זה שקובע את עצמת הקשר, אלא גם תכונות (יצוג) הקודקודים. הדבר להקנות יצוג וקטורי לקודקודים היא באמצעות הטלה זו. במקרים אחרים, המודול לומד לשירות משקל שונה למילימ' שעונה לתלות בעוצמת הסמנטיות והתחבירית שלהן, ומתוך כך משערק את עצמת הקשר באופן מדויק. נקודה זו מאפשרת להבין מדוע הטרנספוררים מסווגים לפטור בעיות שרשות שקדמו להן לא הצליחו.

נקודות מבט נוספת על העניין הינה ההבדל בין ארכיטקטורת הטרנספורמים למודלים שכבר לנו, באופן שבו המודול לומד את התלות בין מרכיבי הקטלט. במאמר [4] עצמת הקשר בין מרכיבי הקטלט למנגנון חושבה באופן הבא:

$$e_{ij} = \text{attention}(s_{i-1}, h_j) = v_a^T * \tanh(W * [s_{i-1}; h_j]), j = 1, \dots, T$$

משוואה 6 - תשומת הלב כפי שהזינה במאמר [4]

כאשר h_j ו- s_{i-1} הם המצביעים הפנימיים מהמפענה והמקודד בהתאם (להרחבת הקריאה, ניתן למצוא את הפירוט המלא במאמר הקודם שלנו על מנגנון תשומת הלב).

זהו ההבדל העיקרי בין שני מנגנון תשומת הלב הוא זה שלומד למשקל את חישובות קלט מסוים באמצעות המכפלה במטריצה W , דבר שגורם לחוסר יכולת לייצג קשרים מורכבים בדעתה. לעומת זאת, בארכיטקטורת הטרנספורמים אנו כבר לא וקווים לסת משקولات בוודח (המכיל W ו- a) שילמד את הקשרים בין מרכיבי המידע השונים בתוך המנגנון, אלא דורשים זאת מהרשת בשלב מוקדם יותר באמצעות הטליה עלייה דיברנו. פעולה זו מצמצמת את המנגנון תשומת הלב בטרנספורמים להיות חישוב קורלצייה בין משתנים שונים אולם היא הופכת את המודל להיות עצמאי בהרבה.

לפועלות הטליה ישן מספר שימושיות חשובות:

- היא מאפשרת למודל ללמידה ייצוגיע של המידע בהתאם לשימושה הנדרשת. כל אחד מהאובייקטים הללו מייצג התבוננות שונה על הקלט, דבר המאפשר למודל ללמידה אספקטים שונים עליו.
- מכיוון שהיצוג מתחשב בקשרים בתוך הדעתה, הוא מאפשר למנגנון תשומת הלב להתאים דרגות חשיבות שונות לחלקים שונים בקלט בהתאם לתוכן שלהם.

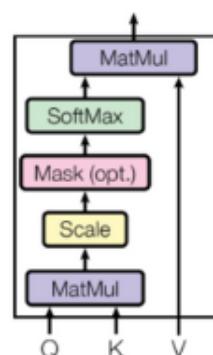
הפלט של המנגנון תשומת הלב מוגדר כמטריצה בגודל $d \times N$, המהווה את קבוצת הוקטורים המייצנים "יצוניים תלויי" הקשור של כל הtokנים בסדרה.

המודול המתמטי של המנגנון תשומת הלב

- כאמור הקלט שנשמענו כתוב ב- X הוא מטריצה בגודל $d \times N$ כאשר N מייצג את מספר הtokנים המתוקבים בכנסה של פונקציית תשומת הלב.
- 3 המטריצות המשמשות להעברת הקלט לשולשות האובייקטים החדשניים מסומנות בתוך Q , W_v , W_k , W_a (שאלותה, מפתח וערך בהתאם) והן כאמור מעבירות את הקלט לייצוג החדש שלו. בפרק הבא נראה כי מטריצות אלו משמשות אותנו להטלת הקלט ל h מרחבים שונים כל אחד בגודל $/h$ עבור מנגנון תשומת הלב הרב-ראשית.

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k})V$$

משוואת 7 - תשומת הלב של טרנספורמים



איור 9 - תרשים הזרימה של מנגנון תשומת הלב העצמית

נפרק את המשוואה לגורמים:

- מטריצות W_K, W_Q, W_V הן בעלות מידת $d_k \times d$ כל אחת (המאמר המקורי מגדיר אותן באותו הנודד):

$$K = X * W_K, Q = X * W_Q, V = X * W_V$$

- X הוא מטריצה בגודל $d \times N$ המייצנת את הטוקנים לאחר שיבוץ וקידוד תלוי מיקום. גודל כל אחת מהמטריצות הינה $d \times d_k$, כאשר בחלק זה נגדר $d_k = d$ כמו במאמר המקורי.
- $K^T * Q$ - תפקידה לחשב את עצמת התאימות (compatibility) שבין כל השאלות לבין כל המפתחות האפשריים. התוצאה הינה מטריצה בגודל $N \times N$.
- מטרת החלוקה היא למנוע פונקציית ה-softmax להגיע לרווחה, דבר שימושי לדעיכת גרדיאנטים ובכך עלולה לעכב את הלמידה. על מנת להבין מדוע הקלט של פונקציית ה-softmax יכול לנזול/לקטן מעבר לרצוי, ניתן להרחיב את הקראיה בקישור [הבא](#).
- $(\sqrt{d_k} / \sqrt{d_k}) softmax(QK^T)$ - כפי שראינו בארכיטקטורות קודמות, פונקציית ה-softmax מאפשרת לקבל פונקציית תשומת לב רציפה, הממשקלת את עצמות התאימות בין שאלתה נתונה למפתח ביחס לכל צמדי שאלתה-פתרונות אחרים.
- $A = (\sqrt{d_k} / \sqrt{d_k}) softmax(QK^T + M)$ - מכיוון שהפעלת פונקציית ה-softmax מחזירה את המשקל של צמדי שאלתה-פתרונות, המכפלה במטריצת הערכים משרתת את אותה המטריה כמו בארכיטקטורות קודמות, ומאפשרת לבנות את ייצוג הקלט באופן רציף.

תשומת לב ממוסכת

כפי שהזכרנו בפרק הקודמים, אנו ממסכים חלק מהמידע המתkeletal בכניסת המפענה על מנת שלא יוכל לראות חלקים עתידיים של הטקסט הנבנה ובכך לפוגם בתהליכי הלמידה. על מנת לעשות זאת אנו מוסיפים מסטור במנגנון תשומת הלב. لكن, משווהה 6 משתנה לשווהה 7 כאשר M מכילה ∞ - (בפועל מדובר במספר שלילי מאד גדול) במקומות בהם לא נרצה לחת נישה למפענה, ואפס (= אין השפעה) במקומות שכן נרצה לחת נישה. ערך זה (מינוס אינסוף) מתרגם בפונקציית ה-softmax לאפס.

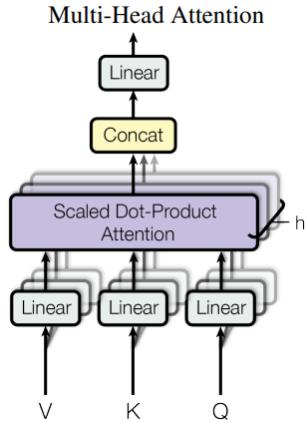
$$\text{Attention}(Q, K, V, M) = softmax((QK^T + M) / \sqrt{d_k})V$$

משווהה 8 - תשומת הלב עם מיסוך

מנגנון תשומת לב הרב-ראשית

ארQUITקטורת הטרנספוררים מצינה הרחבה למנגנון תשומת הלב בדמות מנגנון תשומת לב רב ראשית (multi head attention). באופן פועל המנגנון הינה הטלת הטוקנים המשובצים h -ה מרוחבים שונים, כולם בגודל $N \times d_k$, במקומות להטיל אותם למרוחב אחד בעל מידת d_{model} כאשר $d_k = d$. במקרים פשוטות, ישנו h שלישיות שונות של מטריצות שאלתה מפתח וערך. כל שלישיה שכזו, הנלמדת באופן עצמאי, מייצנת קשרים שונים בקלט, ומעובדת על ידי מנגנון תשומת לב עצמאי כפי שתואר בפסקה הקודמת. בסוף החישוב מוצאי כל הראשים משורשרים אחד לשני, ומוטלים למישור $N \times d_{model}$ באמצעות מטריצה W .





איור 10 - מבנה תשומת הלב הרב-ראשית

משוואה 9 מציגה את המודול המתמטי של הפעולה שתיארנו כעת:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) * W^O$$

$$\text{where } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

משוואה 9 - מבנה תשומת הלב הרב-ראשית

או מדוע אנו זקוקים לשומת לב רב ראשית?

החוקרים במאמר המקורי טוענים כי השימוש במנגנון תשומת הלב הרב-ראשית מאפשר למודל ללמידה

הקשרים שונים הקיימים בטקסט המתבטים כאינטראקציות שונות בין המילים.

אבל מדוע זה מאפשר למעשה? הסיבה לכך נובעת מהעובדה שאנו לא רק מטילים את המידע בראשים

שונים אלא גם מפעילים מבנה תשומת הלב עליהם. דבר זה מאפשר לכל ראש להתמקד בתוכנות שונות של

הקלט וכתוצאה מכך ניתן למודל תלויות שונות בתוכו.

מאמר [6] שבחן את תפקוד הראשים השונים במנגנון, הגיע למסקנה כי ראשיים שונים מבצעים תפקודים

שונים. לדוגמה, תפקוד אחד הראשים היה לנשhaft למיללים במיקומים קרובים, תפקוד של ראש היה

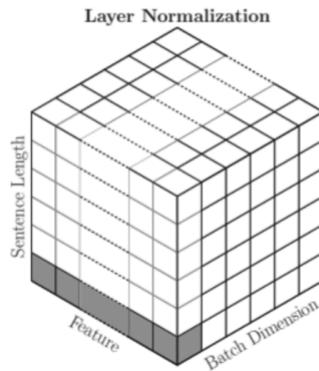
לבצע זהותי ומידול של מיללים שהופיעו בתרומות מאוד נמוכה, ותפקוד ראש אחר הינו למידל מיללים בעלות

קשר ספציפי. המאמר ביצע אנליזה זו על ידי גיזום (pruning) של ראשי תשומת הלב במהלך השימוש

ברשת (inference).

שכבה הנרמול (Layer Normalization)

כפי שציינו בפתח, כותבי המאמר המקורי בחרו להשתמש ב- (layer normalization) LN. כותבי המאמר לא מציינים מדוע הם בחרו להשתמש דווקא בסוג נרמול זה על פני שיטות נרמול אחרות, אולם כפי שנראה בהמשך, נרמול על פני כל השכבה, מונע בעיות היצאות במהלך השימוש בנרמול על פני קבוצה (batch).



איור 11 - שכבה layer normalization של שכבות הפעלים

מאמר [7] (שמציע בין היתר שיטת נרמול חדשה עבור יישומי שפה בתרנספוררים) מנטח את המאפיינים הסטטיסטיים של LN (באצ') בנסיבות של ניתוח שפה טبيعית, ומציג פרשנות משלו מדוע אנו משתמשים דווקא ב-LN במקום שיטת הנרמול (batch normalization) BN. כותבי המאמר מצינים כי הסיבה המרכזית לכך ש-BN לא מתאימה למשימת ניתוח שפה נובעת מכך שסטטיסטיקת הבאים של הטקסט, אינה אחידה. לעומת ערך הトーונים הממוצע והשונות שלהם משתנים באופן בלתי צפוי, באצ' אחד לאחר, דבר הבא לידי ביטוי בשונות של הנרדיאנטים, והnisיון להשתמש בממוצע ושונות נלמדים של באצ' פוגע ביציאותם.

שכבות ה - skip connection - I feed forward

הנושא האחרון שנדון בו הינו שכבת ה-feed forward ושכבות ה-skip connection. שכבות אלו זכו לפופולריות רבה בלמידה عمוקה והן מככבות במגוון מודלים בדומיננסים שונים עבור שימוש רבות. מכיוון שאלהו אבני בניין בסיסיות, עלולה להתעורר השאלה מדוע אנו מזקדים פרך שלם רק עבורו. הסיבה לכך טמונה בעובדה שעובי שכבות אלו חיונית על מנת למנוע מהמודל להתכנס "צוגי" פלט מנוגנים במהלך האימון.

המאמר המקורי אינו מספק את הסיבות לשימוש בשכבות אלו, אולם, המאמרים המאוחרים יותר [8, 9] ביצעו ניתוחים מעמיקים של המודל והגנוו למסקנות הבאות:

לא שכבת FF ו- skip connections של מנגנון תשומת הלב נוטה להתכנס לשיבוץ טוקנים בעלי תלות לינארית. כלומר, אם ניתנים לייצוג \mathbf{z}_i עבור וקטור i קבוע וקאלרים $N_{token}, k_i, i = 1, \dots, N$, אז מלחירה כאשר אנו עורמים (stack) את בלוקי הטרנספורמים. הבעה בתופעה זו הינה שהתלות הלינארית של "צוגי" הトーונים אינה מייצגת את הקשרים החבויים בטקסט ואת הסמנטיקה שלו. הסיבות לתופעה זו טרם התבררו, אך ידוע כי הן נובעות מבנה מנגנון תשומת הלב והתפלגות הדטה.

המחברים של [8] מצינים כי הוספת שכבות skip connections ו- FF מאפשרת להתגבר על סוגיה זו. [8] מראה כי שכבת skip connection הינה הנורם הקיים למניעת התלות הלינארית בין שיבוצי הטוקנים. בנוסף שכבת FF מאטפת את קצב "היווניפורמייזציה" (uniformization) של ייצוג הטוקנים במבנה מנגן skip-connection.

מצין כי FF בטרנספורמרים בניו משכבה לינארית עם אקטיביצית ReLU ושכבה לינארית נוספת (נתונה על ידי המשוואה 9):

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

משוואה 10 - שכבת ה- feed forward

רוצים לדבר איתנו? -> לTAGGOOT, חידודים, הצעות לשיתופי פעולה ושאר ירקוט אטם מוזמנים לבוא לリンקדין
שלמו : [חי, מיכאל](#)

[חי הופמן](#) הוא מסטראנט בשלבי סיום באוניברסיטת אריאל. חוקר בתחום הראייה הממוחשבת והלמידה העמוקה בדגש על אלגוריתמים גנרטיביים. בנוסף כותב תוכן בעברית בתחוםים אלו.

[מיכאל \(מייק\) ארליךסון](#), Michael Erlihson, [PhD](#), Salt Security. עובד בחברת הסיבר Data Scientist. מיכאל חוקר ופועל בתחום הלמידה העמוקה, ולצד זאת מרצה ומנגיש את החומרם המדעיים לקהל הרחב