

ארQUITטורת הטרנספורמרים (Transformers architecture)

אז מהו הסוד של מבנה תשומת הלב בטרנספורמרים?

כפי שראינו בפרק הקודם, הארכיטקטורות שעשו שימוש במבנה תשומת הלב לבדוק לא שרדו את מבנה הזם, 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 קידוד תליי מיקום, ומדוע הוא משחק תפקיד חשוב בארכיטקטורת הטרנספורמרים?
- מהם תפקידם של המזודד והמפענה בטרנספורמרים, וכייזד הם עובדים יחד?
- כיצד מבנה תשומת הלב בא לידי ביטוי בטרנספורמרים?
- מהו הייחוד של מבנה תשומת הלב בטרנספורמרים ביחס לחישוב תשומת הלב ברשותם איטרטיביות, ולמה תשומת לב היא אכן כל **מה שאנו זוקקים לו?**
- כיצד טרנספורמרים פתרו את המגבלת העיקרית של הארכיטקטורות שקדמו להן, העיבוד הטורי של הדאטה?

אנו נבנה את המאמר בצורה של בובשקה (או מטרישקה ברוסית תקנית). המאמר יבנה סדרה של קופסאות שחרות, 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 ותיר כקופסאות שחרות. אף על פי שהמבנה המודול פשוט באופן יחסית, הוא סובל מתנדית של נושא מורכב ומאתגר להבנה. זאת מכיוון שהוא בנוי ממספר חלקים המבוססים על עקרונות מופשטים כך שהם אינם מובנים לעומק, קשה לחברם יחד לכדי רעיון כולל. מסיבה זו החלטנו לבנות את ההסבר, כך שבכל שלב נוכל להתמקד בReLUוניות ספציפיים האחוריים אבן בניין מסוימת, ואחרות להסביר בתור "קופסא שחרה" שכן 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 שחרה, מלבד חלקו השיבוץ והקידוד תליי מיקום של הטוקנים (token encoding and positional encoding) שזכרים לפרקיהם משליהם. אנו נדבר על הקלט והפלט של המודול. נסביר כיצד הקלט מארגן כך שנitin יהיה לעבדו באופן מקביל, וכייזד הפלט נבנה באופן אוטורגריסבי. בנוסף, נדבר על "שרשת החייל" של הקלט, החל מקלט טקסט בשפה טבעית דרך חבי' והפיכתו בחזרה לטקסט בשפה טבעית בموقع הרשות.

קופסא שנייה: בחלק זה נתעמק במבנה הפנימי של המזודד והמפענה. נסביר מהו התפקיד שלהם, וכייזד הם עובדים יחד. כן נשיר את החלקים הפנימיים של המזודד והמפענה כקופסאות שחרות (מבנה תשומת הלב, שכבת ה-*feed-forward* וכו').

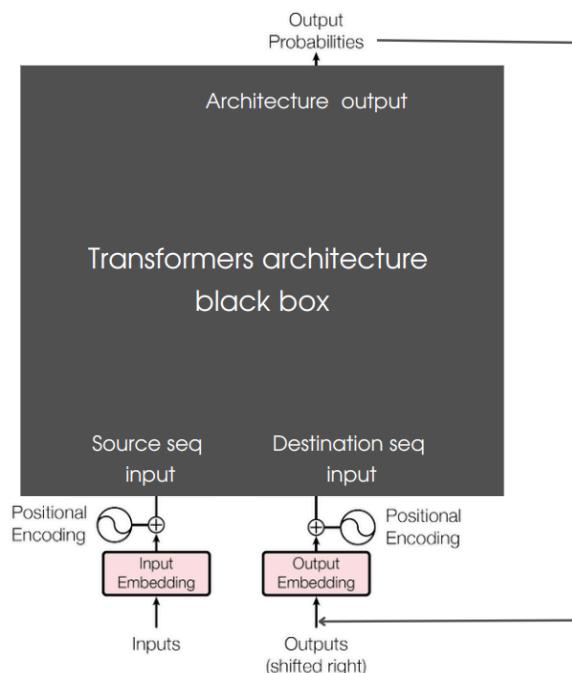
קופסאה שלישית: בחלק זה נסביר מדוע מקייף על הבלוקים המרכיבים את המזודד והמפענה. נרחיב על מבנה הארכיטקטורה של כל אחד מחלקים אלו וונפתח למשה את כל הקופסאות השחרות שנוטרו לנו מהשלב האחרון. נציג לראשונה את הרעיון המרכזי שהרשות בנויה סביבו,

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.
- פולטי היבנים של המקודד והמפענה ובפרט מהו הפלט אחרי שכבה תשומת הלב, והפלט הסופי של המקודד.
- מהו הקלט למנגנון תשומת הלב המוצלבת ומנגנון תשומת הלב העצמית.
- מהו מנגנון תשומת הלב הרב הראשית ? (layer normalization)
- מודיע נדרשת שכבת הנרמול (layer normalization)
- נראה כיצד skip-connections ושבבות ה feed forward מלאות 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 ויכל בתוכו את התהיליך שעובר הקלט מקבלתו בטקסט בשפה טבעית ועד לשלב שניית להזינו לתוך הרשות לצורך האימון. בחלק השני אנו עוסקים בפלט של המודל עד להפיכתו למילה בשפה טבעית שוב.

חלק ראשון: קלט הרשות

כאשר אנו מדברים על מודלים לעיבוד שפה טبيعית, 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 המוכחות על ידי המילון.

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

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

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

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

טוקנייזציה (tokenization):

השלב הראשון בעיבוד הקלט הוא טוקנייזציה. פעולה זו מחלקת את הקלט למילים (בתוך קטגוריות זו אנו כוללים גם סימני פיסוק, וחלקן מילים). ישנים טוקנייזרים שונים שיחלקו את אותה המילה באופן שונה. לדוגמה, המילה 't'don' יכולה להתחולק באופנים הבאים: ['t','don'] או ['t','#','don'] (הסביר מפורט על הטוקנייזרים השונים ניתן למצוא [כאן](#)). לאחר הפעלת טוקנייזציה בסיסית (כל טוקן הינו מילה או סימן פיסוק) על המשפט שננתנו 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 למיניהם) משתמשים בשיטות טוקנייזיה מתקדמות, בהם חלק מהטוקנים הינם תתי-מילים לא מילים שלמות. העיקרונו המוביל בשיטות טוקנייזיה אלו (BytePair Encoding ו-WordPiece) הוא הקניית טוקנים למילים או תתי-מילים השכיחים ביותר בסט הנתונים (dataset) עליו הוא מאומן (בדרכן כל סט זה הינו מנון ועוצם בגודלו). בנוסף, [BERT](#) מגדיר שני טוקנים מיוחדים לתוצר הטוקנייזיה: CLS, SEP. התפקיד של SEP הוא להפריד בין משפטים. טוקן - CLS משמש ליצור ייצוג וקטורי של מקטעי טקסט (הסביר מעמיק יותר ניתן למצוא [בלינק](#)). נציין כי מודלים אחרים מגדרים טוקנים מיוחדים בצורה אחרת.

השלב הבא בפעולות הטוקנייזיה הוא מתן מספר זהה ייחודי לכל טוקן שקיבלנו (כלומר כל טוקן מקבל מספר סידורי). במידה ושנן מילים במשפט שלא נמצאות במילון שלנו, הן יחולקו לתתי טוקנים, עד שכל מילה בקלט תקבל מספר זהה ייחודי (או קבוצת מספרים במידה ופעולות הטוקנייזיה 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 (Learnable Token Embedding) במרחב וקטורי. המינדים השונים בוקטור מיצגים מאפיינים סמנטיים ותחביריים שונים שלו. כותזאה מכך ניתן למדל את מערכת היחסים בין הטוקנים באמצעות אריתמטיות. בארכיטקטורות שקדמו לтренנספורמרים,

נעשו שימוש במודלים שאומנו במיוחד לבנות שיבוץ זה (לדוגמה [word2vec](#)). לעומת זאת, בארכיטקטורת הטרנספורמרים שיבוץ הטוקנים משולב בתהיליך האימון.

כעת נתאר את שני השלבים האחרונים שהקלט עובר עד שנitin להזינו למקודד ולמפענח: תהליך השיבוץ של הטוקנים וקידוד תלוי מיקום (positional encoding). אנו נסביר את החלקים שאינם מוסתרים על ידי קורסא שחורה באירור 1 כלומר Input Embedding - **שיבוץ הטוקנים (Positional Encoding)**:
Embedding

מוצא שכבת השיבוץ נתן על ידי המשוואה הבאה:

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

משוואה 1 - שיבוץ הטוקנים למרחב נסתר

- X - מטריצה הבניה מ [one hot encoded vectors](#) שנודלה 7×7 , כאשר 7 מסמן את מספר הטוקנים במלון. גודל המטריצה X נובע מכך שנודלו של כל וקטור ה-one-hot-one והוא 7 וישנו 7 וקטורים בלבד.
- i - אינדקס הטוקן המשובץ.
- d_{model} - מימד מרחב הקידוד (шибוץ).
- E - מטריצת שיבוץ נלמדת שנודלה $d_{model} \times 7$, מתפרקת כ- C-TU המכילה את השיבוצים של כל הטוקנים, כלומר אנו משבצם 7 טוקנים למרחב שנודל d_{model} .

בפועל, המכפלת $[i]X * E$ היא גישה למיקום i בטבלה E . צורת ייצוג זו היא עיליה מבחינה חשיבותית, כך ניתן לנצל את הטבלה לחישוב שיבוץ הטוקנים של הקלט בו זמןית (במקביל). כתבי המאמר "[Attention is All You Need](#)" לא מצינים מהי הסיבה להכפלת מוצא השכבה הילינארית בגודל $\sqrt{d_{model}}$. אולם, ישנן השערות שהדבר אמרו למונע את דעיכת הנדריאנטים שכן השיבוץ עלול ליצור ערכאים גבוהים מדי שיבולו לרווחה של פונקציית softmax במנגנון תשומת הלב (אשר נרחב עליו בהמשך). השערה אחרת טוענת כי המטריה הינה למנוע מערך קידוד תלוי המיקום (שנסבירותו בהמשך) להיות דומיננטי. אולם, אלו הן רק השערות ויש להתייחס אליהם בהתאם.

כעת נשאלת השאלה מדוע אנו משקיעים בקידוד עצמאי של הקלט ולא משתמשים ברשותות שאומנו מראש (כגון word2vec) במיוחד עבור מטריה זו?

הדבר נובע משלוש סיבות מרכזיות:

1. במשימות הכרוכות ניתוח שפה כל מודל מגדר מהו שיטת השיבוץ והמלון המשמשים לצרכי הטוקנייזציה של הטקסט. מתור כך נובעת ההבנה מדוע לא ניתן להשתמש במודל שיבוץ טוקנים שאומן עם נישה אחרת לטוקנייזציה (כלומר אי אפשר להשתמש ב-[word2vec](#) לטוקנייזציה של טרנספורמרים המשתמשים בשיטת טוקנייזציה כמו [wordpiece](#) או [byte pair encoding](#)).
2. מכיוון שנitin להשתמש באותה המילה בהקשרים שונים או בcplusplus, علينا למצוא ייצוג ממוצע עבור הטוקנים המייצגים את המילה למרחב וקטורי. דוגמא לכך יכולה להיות המילה bat, bat, שיכולה להופיע בהקשר של משחק כדור (bat = מחבט) או בהקשר של זואלוגיה (bat = עטלף). מכיוון שפה טבעית הינה בעיה מורכבת לאפין, נדרשת כמות עצומה של DATA כדי לבנות מודלים שמסוגלים ללמידה את מבנה הקשרים החבויים בתוכה. יציגו טוקנים המופקם באמצעות מודלים כגון [word2vec](#) שאומנו על כמות DATA הרבה יותר מאשר של הדאטסהטים העצומים

שימושים בהם לאימון 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. ولكن, הדבר הגינוי ביותר לעשות הוא למת לרשות ללמידה את השיבוץ שהכי מתאים לה באופן עצמאי.

קידוד תלוי מיקום (Positional Encoding)

החדשנות של ארכיטקטורת הטרנספורמרים נבעה מהניסיונו לענות על השאלה הבאה: **יכיזד ניתן לוותר על הפעלה האיטרטיבית של הרשות בעיבוד סדרות?** הפתרון ההגוני ביותר הוא ניתוח כל הקלט במקביל. אבל כיצד ניתן לבצע זאת? אחת הדרכים לעשות זאת היא ייצוג הקלט כטס (מבנה נתונים שבו הסדר אינו רלוונטי) דבר המאפשר לנו להזין אותו כמקרה אחת. אולם, הקלט למודל הינו סדרתי. אז כיצד ניתן להתגבר על סתירה זו? הפתרון הוא לספק למודל מידע על הסדר של יחידות הדאטה, למשל, מידע על מיקום המילה בטקסט שיאפשר למודל להבין את המשמעות של מרחק בין מילים. במידה ולא נספק למודל מידע זה מנגנון תשומת הלב שארכיטקטורת הטרנספורמרים מבוססת עליו יוכל לנתח באופן זה את שני המשפטים הבאים:

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

המידע שאנו מוסיפים לקלט מאפשר למודל להבין מיקום של כל יחידת דата בטקסט ואת מרחקה היחסית מכל יחידת דטה אחרת. הוספה נוספת זה מאפשרת לרשות ובפרט לפונקציית תשומת הלב להתחשב במיקום 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) או **Positional embedding**.

קידוד תלוי מיקום בטרנספורמרים

כיצד מבצעים קידוד תלוי מיקום? נזכיר שמטרתנו היא להעניק למודל יכולת ללמידה מהמרחב בין יחידות הקלט. אילו נספּ שאננו מעוניינים בו הוא שהקידוד יהיה ייחודי עבור כל מיקום בклט, אחרת לא ניתן יהיה להבחין בין מילים במיקומים שונים. הפתרון הנאיivi הוא להשתמש ב- "one-hot encoding". בקידוד זה אנו יוצרים וקטור שאורך שהוא אורך סדרת הקלט עברו כל טוקן, ומאתחלים את ערכיו לאפסים, במקומות בו מופיעעה המילה אנו מציבים 1.

ניקח לדוגמה את המשפט הבא:

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

דוגמא זו מייצגת את הבעיה שקיים וזה מעמיד בפניםו, והוא ש-*one-hot encoding* הוא קידוד שווה-מרחוק (equidistant), כל וקטור מרוחק מכל וקטור אחר ב- $\sqrt{2}$. בדוגמה שלנו המילה "the" מופיעה 3 פעמים, ולכן הרשות תתקשה לשיר כל "the" למילה המוקורת אליה (dog, fox, river). מתוך כך עולה השאלה: **יכיזד ניתן לקודד מילה כך שמיוכנה במשפט יקונה לה ערך ייחודי ובנוסף שהמרחב בין הקידודים ישקף את מרחק בין מיקומי המילים בклט?** במקרים אחרים אנו רצים קידוד המקרב את ערכם של כל זוג וקטוריים המייצגים טוקנים קרובים (מבחינת מרוחקם בклט) ומרחיק כל שני וקטורים המייצגים מילים רחוקות.

או מהו הפתרון?

קידוד תלוי מיקום באמצעות פונקציות מחזוריות:

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

משוואות 2 - קידוד תלו' מיקום

כאר:

- *sod* - מיקום הטוקן בסדרה המקורית.
 - *i* - האינדקס בטור המרחב השימוש d_{model} כאשר מתקיים: $\{d_{model}\} \in i$.

cut נסbir את המשוואה עבור $d_{model} = 512$ (כפי שהוא מוגדר במאמר המקורי). אנו מחשבים את הוקטור עבור מיקום (pos) של כל טוקן בסדרה, כאשר החישוב הוא לפי פונקציית הסינוס עבור מימדים זוגיים של וקטור הקידוד ופונקציית הקוסינוס למימדים 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 הקידוד שהתקבל עם וקטור השיבוץ.

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

או מודיע השימוש בפונקציות מחזוריות מספק את הדרישות שניסחנו בתחום הפרק הקודם?

זכור את הדרישות שאנו מבקשים עברו וקטור הקידוד ונראה כיצד הfonוקציות המוחזרות מספקות מענה לכל אחת מהן.

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

כפי שתיארנו בפתחה, על המודול ללמידה כיצד לאמוד מרחוק. אולם, אנו לא מעוניינים למדוד את המרחק כיחידה אבסולוטית (כלומר מרחק של טוון לשנה מהטוון הראשון בסדרת הקלט), אלא את **המרחב בין מיקומי הטוונים בסדרה**. המשמעות הינה, שאנו מקובעים טוון מסוים בסדרה, וමבקשים למדוד את המרחק ביןו לכל הטוונים בקלט. לדוגמא, אם המיקום האבסולוטי של הטוון הוא 67, ואנו רצאים למדוד את חישובתו בטוון במיקום ה-47 אוו' המרחק היחסיבי יהיה 20-.

במילים אחרות, אנו בוחנים את המרחקים בין pos1 - pos2 המקיימים:

$$pos2 = pos1 + k$$

בפשטות, אנו מספקים לרשת את ייצוג של 1pos ו- 2pos ומבקשים מהמודול "להפיק" ייצוג של המרחק א' בין'ם.

כיצד פונקציות טריגונומטריות עוזרות לנו במקהה זה? התשובה לכך טמונה בעובדה שפונקציות אלו מקיימות את הזהויות הטריגונומטריות הבאות:

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

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

משווהה 3 - זהויות טריגונומטריות עבור סכום בתור פונקציית סינוס וקוסינוס

ולכן, חישוב המרחק בין שבוצי טוקנים מרוחקים אחד מהשני ב- k -טוקנים באמצעות מנגנון תשומת הלב, יולד ביטויים התלויים במיקום האבסולוטי (sok) והמרחק היחסי (k).

- קידוד ייחודי לכל וקטור (unique embedding)

השאלה הרשונה שנסאלת, היא מודיע לנו משלבים את הפונקציות הטריגונומטריות \cos ו- \sin ? (לומר מודיע המודול לא משתמש רק בפונקציית חוץ או רק בפונקציית \cos). המאמר לא מספק תשובה חד משמעותית עבור שאלת זו, אולם אחת ההשערות הינה ששילוב של שתי פונקציות אלו יוצר יצוג עשיר יותר עבור המידע המיקומי מאשר שימוש בפונקציה יחידה.

השאלה השנייה שנסאלת הינה מודיע המחברים בחורו להשתמש דווקא בביטוי:

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

$$pos^{2i/d_{\text{model}}}$$

על מנת להבין מודיע אנו משתמשים דווקא בביטוי זה עלינו להבין את מרכיביו ואת החזקן שלהם:

- \cos - הדרישה הבסיסית של קידוד תלוי מיקום היא ליצור וקטור ייחודי עבור כל טוקן בקלט. ולכן, הדבר הניגנו ביותר לעשות הוא להשתמש במיקום שלו בקלט (pos), המendir אותו באופן ייחודי.
 - $2i/d_{\text{model}}$ - מכיוון שוקטור הקידוד מתווסף לוקטור השיבוץ, הוא מכיל d ממדים, ולכן יש צורך לקבוע מה יהיה הערך שיקבל כל ממד.
- הפתרון הנאייבי** הוא להשתמש בערכו של $(pos) \sin$ או $(pos) \cos$ עבור כל המרכיבים של וקטורי הקידוד. אולם, הבעיה בכךשה זו הינה קיום מחזריות במרחקים בין וקטורים. לעומת, עבור ערך כלשהו, התלויה במורכבות הפונקציה, המרחקים בין וקטורים המקודדים את מיקומם של טוקנים הנמצאים במרחקים a ו- b למשל עלולים להיות כמעט שווים. זהו מצב לא רצוי מכיוון שהקידוד אמרור לשקר 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 למחרוזות שכזו (ערך a ו- b גדול ככל שהפונקציה מורכבת יותר).

הפתרון המוצע הוא לחת לכל מימד בוקטור ערך שונה. דבר זה מתבצע באמצעות מכפלת של \cos במייד i וחולקוו ב- d_{model} .

כלומר אנו יוצרים את המערך הבא:

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

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

על מנת להוכיח את התזה שהצענו כאן, נבחן מה הייתה יכולה להיות האלטרנטיבה פשוטה ביותר וכי צד היהתה נראית תוצאה.

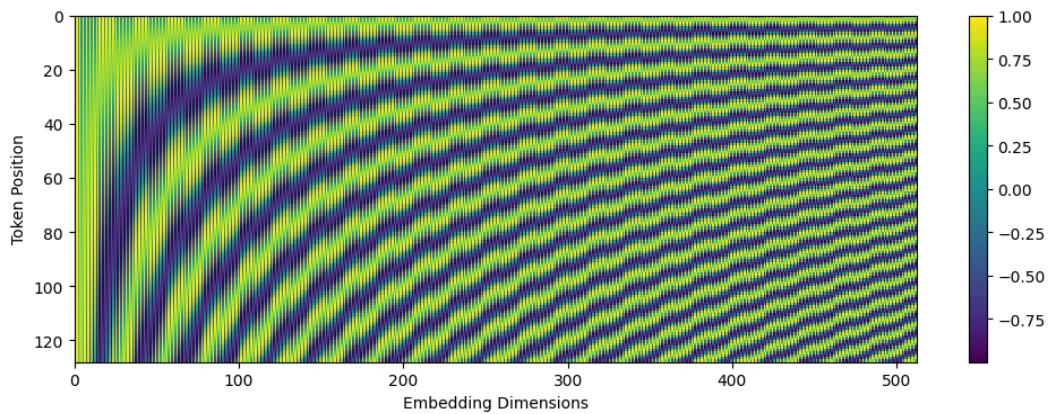
למשל עבור קידוד תלוי מיקום השווה ל:

$$\text{Positional Encoding}(pos, 2i) = \sin(pos * (2i/d_{\text{model}}))$$

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

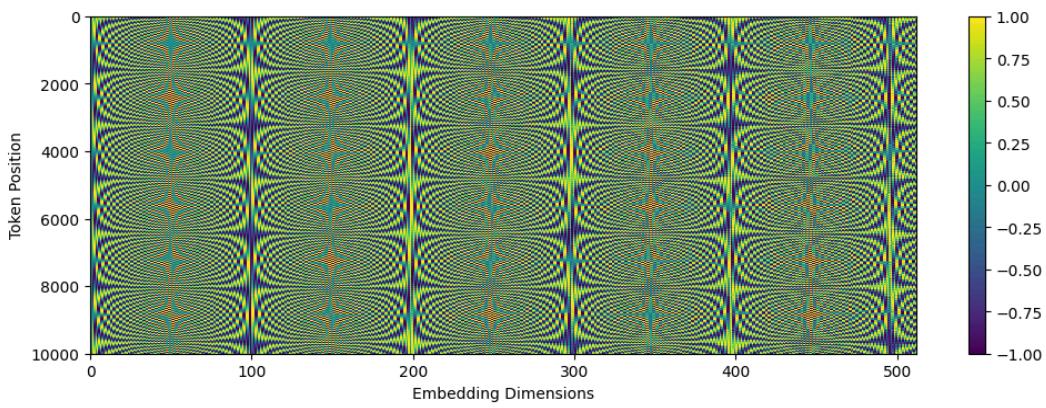
משוואה 5 - קיזוד תלוי מיקום ללא המרכיב המעריצי

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



איור 2 - קיזוד תלוי מיקום אלטרנטיבי, עבור 128 טוקנים

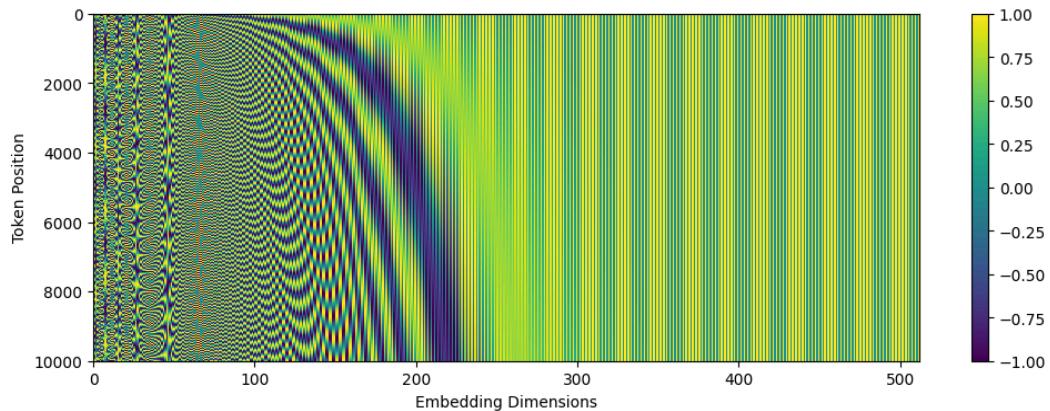
לכארה לא נראהות בעיה גלויה לעין. אולם, אם נגדיל את מספר הטוקנים ל-10,000 כאשר $d_{model} = 512$ (אורך הקשר ריאלי עבור טרנספורמרים בני ימינו) נקבל את הנגרף הבא:



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

כפי שניתן לראות, כאשר מספר הטוקנים בסדרה, גודל d_{model} מתחילה להווצר חזרתיות בדפוסי קיזוד תלוי המיקום. למרות שניתן לפתר את זה באמצעות הגדלת ערכו של d_{model} , זה לא פתרון ריאלי.

כותבי המאמר מציעים את הקידוד המעריצי. הסיבה שהקידוד המעריצי עובד בצורה כל כך טובה, נובעת מהעובדת שהוא מಡק חוזרתות עבור כל גודל סדרת קלט בכניסה, כפי שניתן לראות באירור 4 עבור סדרה בעלת 10,000 טוקנים.



איור 4 - קידוד תלוי מקום מעריצי עבור 10,000 טוקנים

להרחבת הקריאה על הניסויים שביצענו על מנת להגעה למסקנות אלו ניתן להיכנס למחברת זו.

שיטות נוספות לביצוע קידוד תלוי מקום:

קידוד תלוי מקום אינו מחייב לאופן בו הוא מוצג במאמר המקורי, וישנן שיטות נוספות. אנו לא נסקור שיטות אלו במאמר זה (אך ניתן למצוא אותן ב[1] [2] ו-[3] או בסקירה עתידית שנבצע) אולם נתעכט על נקודת אחת: מודיע החוקרים משתמשים דוקא ביצוגים באמצעות פונקציות מחזוריות, ולא לומדים את הקידוד חלק מהתהילה האימוני. 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 בצע קידוד ידוע מראש ונלמד, מכיוון שהדבר עיל יותר מבחינה חישובית.

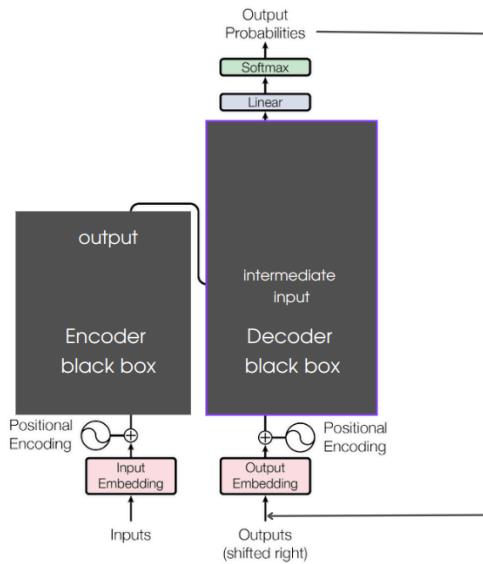
חלק שני : פלט הרשות

פלט המודל הוא תוצר של שכבה softmax על פני כל מיליון הטוקנים שאנו משתמשים בו, כאשר הרשות מייצרת בכל הפעלה את הטוקן החזוי הבא ביחס לקלט. במהלך האימון אנו בונים את הפלט של המודל טוקן אחריו טוקן עד שנadol הסדרה שווה לנודול הסדרה המקורי, ואז מבצעים את פונקציית loss ביחס לפטן המקורי של הרשות.

במהלך השימוש במודל (inference) אנו לא יודעים מה אמרור להיות אורך משפט המוצא. לכן علينا להפוך את הפעלת המודל בנסיבות כלשהי. ישנן שיטות שונות לקביעת נקודת העצירה:

- קביעת סף במספר 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 במידה וערץ זה לא חוצה סף מסוים.

קופסא שנייה: מילודד ומפענה



איור 5 - המילודד והמפענה כקופסאות שחורות

המבנה הכללי:

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

חשוב להבין כי במשימת תרגום שתי הסדרות המתקבלות בכניסת מיילודד ומפענה, מוקשורות אחת לשניה, אך כל אחת מהן מתוארת באמצעות ייצוג שונה (בדמות שפה). לדוגמא, משפטים בעברית ובאנגלית המתארים סיור על יד שמחק בכוור, מכילים את אותו התוכן, אך שונים באופן שבו הם מביעים אותו. **הבעיה הניצבת במשימה זו הינה להבין כיצד ממירם מיצוג אחד לייצוג אחר, במיוחד במקרים בהם מדובר במושגים מורכבים.**

זהו תמצית השימוש בטרנספורמרים כפי שהוא מופיע במאמר המקורי שיועד למשימות תרגום וסיסום טקסט. בסופו של דבר, תכלית המודול **שהוצע במאמר** הינה ללמד 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 הדבר" ולכן פסקה זו תקיפה רק למאמר שאנו מנתחים.

מטרת המודול הינה לחזות את ייצוג הקלט במרחב 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 המודול אנו מוסיפים טוקן נוסף שחזינו להיות חלק מסדרת הקלט של המפענה. על מנת ללמידה כיצד לעשות זאת, אנו מזינים למילודד את סדרת המיקור (לדוגמא, הטקסט שאנו מעוניינים לתרגם) כדי שיבנה ייצוג המכיל את הקשרים בטקסט למרחב וקטוריו המשותף לו ולמפענה (במה שירנסבירות מהו מרחב זה וכיצד בונים אותו). המפענה נעזר במידע בניתו הקלט שלו (שנבנה טוקן אחרי טוקן)

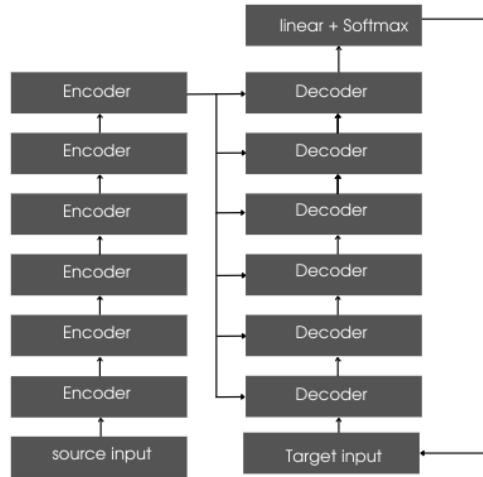
וביצירת הפלט שלו. אם נחזור לדוגמא שפתחנו אותה, המקיים מעביר למפענה את **תוקן הרעיון** שהמשפטים מייצגים, ללא תלות במרקבי הקלט (השפעות השונות).

בדומה לארכיטקטורות קודמות שסקרנו, גם ארכיטקטורות הטרנספורמרים משתמשות במנגנון תשומת לב. אולם בשונה מהן, מנגנון זה הינו לב ליבת הארכיטקטורה ולא מתפרק כמרכיב עזר שתפקידו לבנות ייצוג ביןים למידע המופיע מהקיים או מהמפענה. מכאן מניע גם שם המאמר, כי תשומת לב היא הדבר היחיד שהמודול צריך.

הקיים והמפענה בנויים מ-"לבנים" (blocks) שכיל אחד מהם נקרא **בלוק טרנספורמר** או בקיצור **טראנספורמר**. בלוק טרנספורמר של המקיים 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 בקיים וגם במפענה. אולם **האוף בו המידע מהקיים מזון למנגנון זה שונה מהאוף בו הוא מזון במפענה**. הקלט של המפענה נבנה באופן אוטורגריסיבי, כלומר הפלט שלו הופך להיות חלק מהקלט הבא שלו. עקב לכך 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 והקיים מוערמים (stacked) אחד על השני. איור 6 מציג כיצד חיבור זה מתבצע בפועל.



איור 6 - מערום (stacking) של בלוקי הטרנספורמר

תפקיד המקיים והמפענה

כפי שניתן לראות באיוור 6, כל בלוקי המקיים מחוברים אחד לשני בטוור, כאשר הפלט של כל בלוק מזין את הבלוק אחריו. בינוונוד למקיים, המפענה מזון בפלט המקיים המתתקבל מהבלוק האחרון שלו, בתוספת לפלט הבלוק המפענה הקודם. הסיבה שהארQUITקטורה בנויה כך, טמונה בעובדה שבבלוק המקיים האחרון מפיאק את ייצוג המידע המופשט ביותר ומכיל את האינפורמציה המלאה ביותר עלייו. לכן, הזנתו תסייע למפענה בהבנת ההקשר של הקלט שלו.

קלט ופלט:

cut נגיד מה הפלט והקלט לחלקים השונים של הטרנספורמר. כאשר אנו מדברים על הקלט של המקודד או המפענה אנו תמיד נתיחס לסדרה שהתקבלה בכניסתם.

cut נרחב על הקלט והפלט של המקודד והמפענה:

המקודד:

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

המפענה:

- **קלט המפענה (decoder input)** - סדרת היעד (target) אותה אנו לומדים להזות בצורה אוטורגרסיבית (טוקן לאחר טוקן). חשוב להבין כי חלק מסדרת היעד מוסיף (החלקים שבאים לאחר הטוקן הנוכחי מסוימים על מנת שהרשת לא תשתמש במידע זה). כאמור **הקלט הנוסף** של המפענה, המתתקבל מהמקודד, הינו פלט הבלוק האחרון שלו.
- **פלט המפענה (decoder output)** - קלט לשכבה לנארית שמטרתה לחשב התפלגות של הטוקן הבא בסדרת הפלט הסופית (כגון טקסט). כמו במקודד, הפלט של המפענה נוצר על ידי בלוק הטרנספורמר האחרון שלו.

עקרונות הקלט/פלט של מקודד-מפענה

המקודד מעבד את הקלט בכניסתו באופן מקבילי ומיציר את הפלט בפעולה יחידה (single forward pass) ללא צורך בהפעלה איטרטיבית. בשונה ממנו, המפענה מייציר את הפלט שלו באופן אוטורגרסיבי. המשמעות של אוטורגרסיביות הינה יצירת פלט בהתאם לפלט שכבר נוצרה. כלומר, בכל איטרציה אנו מזינים לתוך הרשות ייחדות מידע נוסף שחושבו באיטרציות קודמות קלט, או אפילו סדרת היעד עצמה עד הטוקן הנוכחי (teacher forcing).

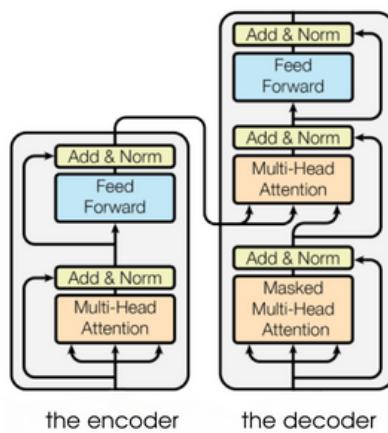
אז מדוע אנו בונים את הקלט של המפענה באופן אוטורגרסיבי? הסיבה לכך נובעת מכך שהמפענה הוא החלק במערכת המפיק את משפט היעד, ומטרתו לחקות את תהליך יצירת המילה האנושי.

אלו ישנה דרך אחרת 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 בודדת (single forward pass) למפענה ונשווה את פלט שהתקבל עם סדרת היעד. אפשרות זו מחייבת למידת פונקציית הזיהוי, אנחנו מקבלים את התוצאה הרצiosa (הרשות מוציאה פלט נכון) אבל הרשות לא לומדת דבר מכיוון שהיא תלמד 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 עובד נט, מאחר ולשפה טבעיות מבנה מורכב, והמפענה יתקשה להסתמך רק על הקלט המקורי, אלא יצטרך להיות מודע לחלקים הקודמים בסדרה שכבר יצר (או קיבלם קלט).

- **האפשרות האחורה ההגונית ביותר** הינה לבנות את הקלט באופן אוטורגרטיבי. כלומר, בכל איטרציה אנו חוזים ארך ורק את הטוקן הבא (שטרם נזהה) ומוסיפים אותו לקלט המפענה המשמש לחיזוי הטוקן הבא. לעיתים לאחר סיום חיזוי מילה (כלומר כאשר נזהה הטוקן "[SEP]" המסמך מרוח בין מילים) המילה שנחוצה מוחלפת במילה מתאימה מסדרת היעד ([teacher forcing](#)). תחילה זה מחקה את אופן יצירתי 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 המקודד והמפענה



איור 7 - המבנה הפנימי של המקודד והמפענה

מבנה הבניין המרכזי של המקודד והמפענה.

המקודד והמפענה בנויים משלושה אבני בניין מרכזיות שבאמצעותן הארכיטקטורה מקבלת את עצמה *הייחודית*.

- **מנגנון תשומת הלב** - מנגן תשומת הלב שהוצע לראשונה ב- [4] הכל חישוב של וקטור דינמי המכיל את הקשרים החשובים ביותר בין המקודד למפענה בעת יצירתו של טוקני הפלט בעת יצירתו של טוקני הפלט (דומה לתשומת הלב המוצלבת בטרנספורמרים). [5] השתמש במנגנון תשומת הלב כדי לחשב את הקשרים בין חלקי הקלט השונים במקודד (דומה לתשומת הלב **העצמית** בטרנספורמרים).
- רעיון דומה יושם גם כמנגנון תשומת הלב של טרנספורמרים שמשקל את עצמת הקשר בין יציגי יחידות דאטה בטור אותה הסדרה (תשומת לב עצמית), או את הקשר בין יציגי יחידות דאטה הנבניהם על ידי המקודד למפענה (תשומת לב מוצלבת). אולם ההבדל העיקרי בין המנגנונים הוא שבטרנספורמרים תשומת הלב (העצמית) מחושבת במקביל עבור כל הטוקנים בתוך הקלט, וכך אין צורך בזכרון. החידוש הנוסף הוא שימוש בשני המנגנונים אלו (עצמית ומוצלבת) יחד.

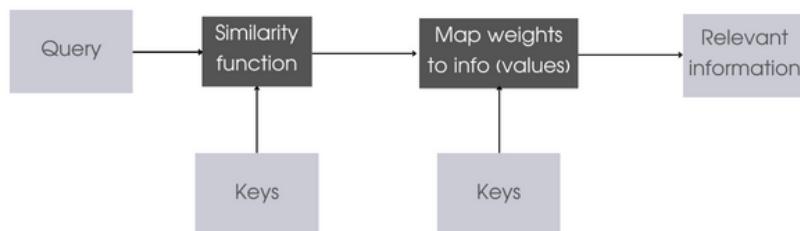
- **רשת feed-forward** - שתי שכבות fully-connected (השכבה השנייה הינה לינארית ולא פונקציית אקטיבציה). לכארה אין טעם לדון בה, אולם יש לרשות זו תפקיד חשוב במבנהו של תוצר מנגנון תשומת הלב. כאשר נדבר עליה נציג מאמרם שמראים כיצד השמתת חלק זה מובילה לירידה משמעותית ביצועי המודל.
- **שכבה Add and Norm** - שכבה שתפקידה לבצע residual connection ונורמל (layer normalization). מוצא פונקציית תשומת הלב היא מטריצה בגודל $d_{model} \times N$ כאשר N מייצג את מספר הトーונים. שכבת זו מנורמלת כל מינד + של הקלט (אנחנו מנורמים את מינד + של כל וקטורי ייצוג של כל הトーונים יחד - קלומר וקטור בגודל N "מנורמל" כל פעם).

מנגנון תשומת הלב

ארQUITקטורת הטרנספורמרים שינתה את האופן בו אנו משתמשים בעיבוד קלט סדרתי. עד לאותו הרגע, מנגנון תשומת הלב השתמשו במידע המתkeletal מהמצבים הפנימיים של רשותות איטרטיביות, שאלצטו חוננת קלט בודד (מילה או חלק מילה) בכל הפעלה. תשומת הלב חושבה באמצעות בניות וקטור זיכרון דינמי, (שדרש הקצתה זיכרון) על מנת לייצג את הקשרים במידע בכל איטרציה. שתי תכונות אלו היו את עקב אכים של מודלים אלו. אולם, עם הופעת הטרנספורמרים, התאפשר בפעם הראשונה לעיבוד מקבילי של הקלט כמקרה אחד. כתוצאה לכך, לא היה צריך יותר בוקטור זיכרון עבור כל איטרציה, מה שאפשר בפעם הראשונה מידול איות של תלויות ארכיטקטורת שקדמו לו, סיבוכיות הירצה וסיבוכיות המיקום (time and space complexity) של מנגנון תשומת הלב עבור הרצה בודדת הימם – ($d_{model}^2 * n^2$), אולם הוא אופן ניסוחו הינו יוכל יתר מבחינה רעיונית כי הוא בניין אינהרטית לעיבוד מקבילי של כל המידע.

- על מנת לבצע זאת הוגדרו 3 אובייקטים הנקרים שאלתיה, מפתח וערך שהרעיון מאחוריהם הושאל מאלגוריתמי אחזור נתונים (information retrieval) (ומנווי חיפוש):
- **שאילתת (query)** היא הבקשה שאנו משתמשים בה על מנת לקבל את הערך/ים.
 - **מפתח (key)** זהה האינפורמציה (יצוג) המקשרת אל הערך, ומשמשת לזיהוי שלו.
 - **ערך (value)** הוא הפריט אותו אנו מחפשים, השמור במסד הנתונים.

דוגמה 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 הסרטוניים, (שם מלא של הסרטון, תיאור שלו, אורך, יצורים וכו') והערך הוא הסרטון בו אנו מעוניינים לצפות. אם נקבע את דוגמת חיפוש סרטוניים להפעלת מנגנון תשומת הלב של טרנספורמרים, הסרטון שיבחר יהיה זה שהמפתח שלו יהיה בעל הקורלציה הגבוהה ביותר לשאילתת שלנו.



אור 8 - המחשת קבלת ערך לפי שאלה וצין

דרך נוספת לחושב על אובייקטים אלו היא באמצעות גראף לא מכון שלם וממושך. ככלומר גראף שבו כל קודקוד מחובר עם כל קודקוד אחר (כולל עצמו) וליקש ביןיהם יש משקל מוגדר. בכל פעע אנו בוחרים קודקוד אחד ומקבאים אותו בתור השאלתה, כל קודקוד בגרף (כולל הקודקוד שקבענו) מוגדר כمفצתה. הערך מוגדר כמשקל הקשת המחברת בין קודקוד השאלתה לקודקוד המפתח. תוצאת פונקציית תשומת הלב תהיה מכפלה פנימית של ייצוג וקטורי של שני הקודקודות 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 בייצוג קשרים בין חלקים שונים של הדטה הסדרתי. בוגנו לאמरנו קודם, שבו מנגנון תשומת הלבעשה שימוש במצבים הפנימיים של המקודד והמפענה, ארכיטקטורת הטרנספורמרם לא מכילה מידע פנימי שמה וולך אנו ווקקים לדרכ בה אנו יכולים לייצג קשרים אלו. מכיוון שמנגנון תשומת הלב של טרנספורמרם מתפקידו מנגנון אוניברסלי גם עבר תשומת הלב העצמית ותשומת הלב המצלבת, עליו להציג כלים מופשטים שיוכולים לייצג את הרעיון עצמו.

הדוגמא שבה הצינו את מנגנון תשומת הלב באמצעות גראף שלם ממושך, מייצגת בדיקות את הקשרים הללו. בדוגמא זו, לא משנה לנו מהין מניעים השאלות או המפתחות, אנחנו עדין יכולים לייצג את הגראף השלם המחבר ביניהם. הייצוג של מנגנון תשומת הלב בצורה שכזו מאפשר לנו להשתמש בו גם בתשומת הלב המצלבת בה המפתחות והערכים מניעים מהמקודד בעוד שהשאלות מניעות מהמפענה.

או כיצד משתמשים במנגנון זה בארכיטקטורה?

כפי שניתן לראות באירור 7, גם המקודד וגם המפענה משתמשים תחילה במנגנון תשומת הלב עצמית, שבו אנו מאטרים את הקשרים בתוך הקלט. 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 מצלבת. בשלב זה אנו מגדירים את השאלהות להיות מוצא פונקציית תשומת הלב של המפענה, ואת המפתחות והערכים אנו מגדירים להיות מוצא בлок המקודד الآخرן.

הקלט למנגנון תשומת הלב של המקודד הינו הפלט של בлок הטרנספורמר ששורשר לפניו (כאשר מדובר בבלוק הטרנספורמר הראשון הקלט הוא סדרת הקלט שעברה טוקנייזציה והוספה קידוד תלוי מיקום). לפניו שקלט כלשהו מוכנס למנגנון תשומת הלב הוא עובר הטלה למרחב חדש באמצעות שלושה שכבות LINאריות (בפשטות, הכפולות במטריצה). במודול המוצע במאמר הטלות אלו הן שיוצרות את המפתח, שאלתה והערכים (אולם אין זה מחייב, ניתן להנדר את המפתח השאלה והערך כרצוננו). זהוי נקודה חשובה, שכן אם **נשתמש בקלט בצורה המקורית ללא הטלה זו, הרשות תאבד את הנגישות (flexibility) שלו לממודד קשרים אלו.**

אבל מדוע זה נכון? אם נחזור לדוגמא של הגראף הממושך, נראה שלא רק משקל הקשת הוא זה שקובע את עצמתה הקשר, אלא גם תכונות (ייצוג) הקודקודות. הדבר להקנות ייצוג וקטורי לקודקודות היא באמצעות הטלה זו. במקרים אחרים, המודול לומד לשני משקל שונה למיללים שונים כתלות במס�מות הסמנטיבית והתחבירית שלהן, ומתוך כך משערך את עצמת הקשר באופן מדויק. נקודה זו מאפשרת להבין מדוע הטרנספורמרם מסוגלים לפתור בעיות שרשות שקדמו להן לא הצליחו.

נקודות מבט נוספת על העניין הנה ההבדל בין ארכיטקטורת הטרנספורמים למודלים שקדמו להן, באופן שבו המודול לומד את התלות בין מרכיבי הקלט. במאמר [4] עצמת הקשר בין מרכיבי הקלט 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 שהוצגה במאמר [4]

כאשר h_j ו- s הם המצבים הפנימיים מהמהפענה והמקודם בהתאם להרחבת הקריאה, ניתן למצוא את הפירוט המלא במאמר המקורי שלנו על מנגנון תשומת הלב.

זהו ההבדל העיקרי בין שני מנגנוני תשומת הלב. במאמר [4] מנגנון תשומת הלב הוא זה שלומד למשקל את חשיבות קלט מסוים באמצעות המכפלה במטריצה W , דבר שגורם לחסור יכולת לייצג קשרים מורכבים בדעתה. לעומת זאת, בארכיטקטורת הטרנספורמים אנו כבר לא זוקרים לסט משקלות בודד (המכיל W ו- v_a) שילד את הקשרים בין מרכיבי המידע השונים בתחום המנגנון, אלא דורשים זאת מהרשת בשלב מוקדם יותר באמצעות הפעלתה עלייה דיברנו. פועלה זו מצמצמת את מנגנון תשומת הלב בטרנספורמים להיות חישוב קורלציה בין משתנים שונים אולם היא הופכת את המודל להיות עצמאי בהרבה.

לפנות הטליה ישן מס' 6 משמעויות חשובות:

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

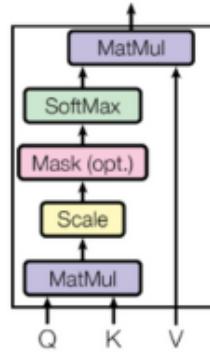
הפלט של מנגנון תשומת הלב מוגדר כמטריצה בוגודל $d \times N$, המהווה את קבוצת הוקטורים המייצנים "יצוגים תלויי" הקשור של כל הtokנים בסדרה.

המודול המתמטי של מנגנון תשומת הלב

- כאמור הקלט שנסמן כת- X הוא מטריצה בוגודל $d \times N$ כאשר N מייצג את מס' הtokנים המתוקבים בכניסה של פונקציית תשומת הלב.
- 3 המטריצות המשמשות להעברת הקלט לשולשות האובייקטים החדשניים מסומנות בתחום Q, W_V, W_K (שאולתה, 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 בוגודל $d \times h$ עבור מנגנון תשומת הלב הרב-ראשית.

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

משוואה 7 - תשומת הלב של טרנספורמים



איור 9 - תרשים הזרימה של מבנה תשומת הלב העצמית

נפרק את המשוואה לגורמים:

- מטריצות W_K, W_Q, W_V הן בעלות מימד $d_k = d_{model}$ כל אחת (המאמר המקורי מגדיר אותן באותו הנודל):

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

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

תשומת לב נוספת

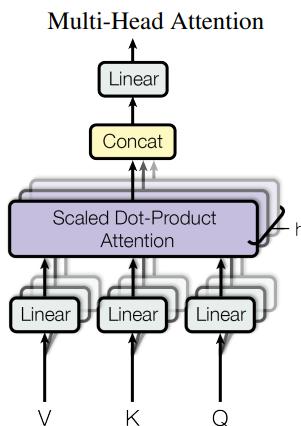
כפי שהזכרנו בפרקים הקודמים, אנו ממסכים חלק מהמידע המתkeletal בכניסת המפענה על מנת שלא יוכל לראות חלקים עתידיים של הטקסט הנבנה ובכך לפוגם בתהילין הלמידה. על מנת לעשות זאת אנו מוסיפים מיסוך במבנה תשומת הלב. לכן, משווהה 6 משתנה לשווהה 7 כאשר M מכילה ∞ - (בפועל מדובר במספר שלילי מאד גדול) במקומות בהם לא נרצה לגישה למפענה, ואפס (= אין השפעה) במקומות שכן נרצה לחשוף. ערך זה (מינוס אינסוף) מתרגם לפונקציית 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 רב-ראשית

ארQUITטורת הטרנספורמרים מצינה הרחבה למנגנון תשומת הלב בדמות מנגנון תשומת לב רב-ראשית (multi head attention). אופן פעולה המנגנון הינה הטלת הטוקנים המשובצים ל- h מרחבים שונים, כולם בגודל $N \times d_k$, במקומם להטיל אותם למרחב אחד בעל מידת d_{model} כאשר $h/d_{model} = d_k$. במלחים פשוטות, ישנו h שלישיות שונות של מטריצות שאליהה מפתח וערך. כל שלישיה שכזו, הנו מודדת באופן עצמאי, מייצגת 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^0 .



איור 10 - מנגנון תשומת הלב רב-ראשית

משוואה 8 מצינה את המידול המתמטי של הפעולה שתיארנו כעת:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) * W^0$$

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

משוואה 9 - מנגנון תשומת הלב רב-ראשית

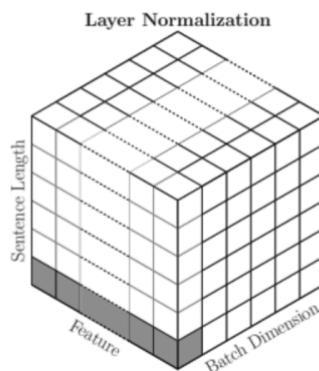
או מדוע אנו זקוקים לתשומת לב רב-ראשית?

החוקרים במאמר המקורי טוענים כי השימוש במנגנון תשומת הלב רב-ראשית מאפשר למודל ללמידה הקשרים שונים הקיימים בטקסט המתבאים Caintraקציות שונות בין המילים. אבל מדוע זה מתאפשר? הסיבה לכך נובעת מהעובדה שאנו לא רק מטילים את המידע הראשים שונים אלא גם מפעילים מנגנון תשומת הלב עליהם. דבר זה מאפשר לכל ראש להתמקד בתכונות שונות של הקלט וכ吐וצאה מכרך ניתן למודל תליות שונות בתוכו.

מאמר [6] שבחן את תפקיך הראשים השונים במנגנון, הגיע למסקנה כי ראשיים שונים מבצעים תפקידיים שונים. לדוגמה, תפקיך אחד הראשים היה לנשת למלילים במקומות קרובים, תפקיך של ראש היה לבצע זיהוי ומידול של מלילים שהופיעו בתדריות מאוד נמוכה, ותקיך ראש אחר הינו למדל מלילים בעלות קשר ספציפי. המאמר ביצע אנליהזה זו על ידי גיזום (pruning) של ראשי תשומת הלב במהלך השימוש ברשת (inference).

שכבה הנרמול (Layer Normalization)

כפי שציינו בפתח, כתבי המאמר המקורי בחרו להשתמש ב- (layer normalization) LN. כתבי המאמר לא מצינים מדויק אם 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 - שכבת הפיצ'רים

מאמר [7] (שמציג בין היתר שיטת נרמול חדשה עבור יישומי שפה בטרנספורמרים) מנטה את המאפיינים הסטטיסטיים של LN (בaż') ביחסות של ניתוח שפה טבעי, ומצביע פרשנות משלו מדויק מאמר מצינים המשמשים דווקא ב-BN במקום שיטת הנרמול (batch normalization) (batch). כתבי המאמר מצינים כי השכבה המרכזית לכך ש-BN לא מתאימה למשימת ניתוח שפה נובעת מכך שסטטיסטיות הביצאים של הטקסט, אינה אחידה. ככלומר ערך הטוקנים הממוצע והשונות שלהם משתנים באופן בלתי צפוי, באז' אחד לאחר, דבר הבא לידי ביטוי בשונות של הנדרידנטים, והניסיון להשתמש בממוצע ושותות נלמדים של באז' פוגע ביצועים.

שכבות ה - skip connection - ו feed forward

הנושא האחרון בו הינו שכבת feed forward ושכבות skip connection. שכבות אלו זכו לפופולריות רבה בלמידה عمוקה והן מככבות במגוון מודלים בדומיננסים שונים עבור משימות רבות. מכיוון שלאו אבני בניין בסיסיות, עלולה להתעורר השאלה מהו מתקדים פרק שלם רק עבורו. הסיבה לכך טמונה בעובדה שנייה שכבות אלו חייניות על מנת למנוע מהמודל להתכנס יציגי פלט מנוגנים במהלך האימון.

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

לא שכבת FF ו- skip connections הפלט של מנגנון תשומת הלב נוטה לשיבוץ טוקנים בעלי תלות לינארית. ככלומר, הם ניתנים לייצוג כ- \sum_i^k עבור וקטור i קבוע וקאלרים $N_{token}, \dots, i = 1, k$. תופעה זו מחייבת כאשר אנו עורמים (stack) את בלוקי הטרנספורמרים. הבעה בתופעה זו הינה שההתלות הלינארית של יציגי הטוקנים אינה מיצגת את הקשרים החבויים בטקסט ואת הסמנטיקה שלו. הסיבות לתופעה זו טרם התבררו, אך ידוע כי הן נובעות מבנה מנגנון תשומת הלב והתפלגות הדאטה.

המחברים של [8] מצינים כי הוספת שכבות skip connections ו- FF מאפשרת להתגבר על סוגיה זו. מראה כי שכבת skip connection הינה הנורם הクリטי למניעת התלות הלינארית בין שיבצ'י הטוקנים. בנוסף שכבת FF מאטה את קצב "היווניפורמייזציה" (uniformization) של "צוני הטוקנים במקומות מנוגן". skip-connections תשומת הלב אך תפקידה אינה בולט כמו שכבת `skip-connections`

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

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

משווהה 10 - שכבת FF