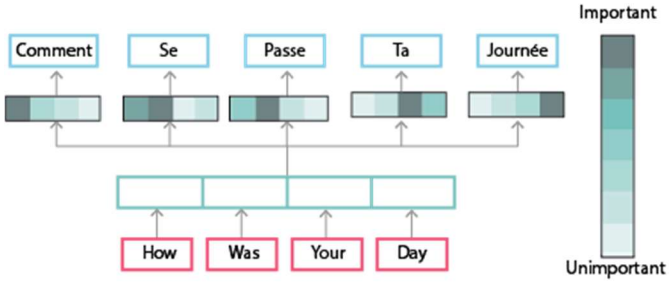
8. Attention Mechanism

8.1 Sequence to Sequence Learning and Attention

8.1.1 Attention in Seq2Seq Models

ניתוח סדרות בהן יש קשר בין האיברים יכול להיעשות בעזרת רשתות עם רכיבי זיכרון, כפי שהוסבר באריכות בפרק 6. ברשתות אלו הסדרה הנכנסת לרשת עוברת דרך encoder היוצר וקטור בגודל ידוע מראש המייצג את הסדרה המקורית, תוך התחשבות בסדר של איברי הסדרה ובקשר ביניהם. לאחר מכן וקטור זה עובר ב-decoder שיכול לפענח את המידע שיש בווקטור ולהציג אותו בצורה אחרת. למשל בתרגום משפה לשפה – מודל של seq2seq מקודד משפט בשפה אחת לווקטור מסוים ולאחר מכן מפענח את הווקטור למשפט בשפה השנייה.

הדרך המקובלת ליצור את הווקטור ולפענח אותו הייתה שימוש בארכיטקטורות שונות של RNN, כמו למשל רשת עמוקה מסוג LSTM או GRU המכילה רכיבי זיכרון. מודלים אלו נתקלו בבעיה בסדרות ארוכות, כיוון שהווקטור מוגבל ביכולת שלו להכיל קשרים בין מספר רב של איברים. כדי להתמודד עם בעיה זו ניתן לנקוט בגישה שונה – במקום ליצור וקטור במוצא ה-encoder, ניתן להשתמש במצבים החבויים של ה-encoder בשילוב המצבים החבויים של ה-decoder, וכך למצוא תלויות בין איברי סדרת הקלט לאיברי סדרת הפלט (general attention) וקשרים בין איברי סדרת הקלט עצמם (self-attention). ניקח לדוגמא תרגום של המשפט "How was your day" מאנגלית לשפה אחרת – במקרה זה מנגנון ה-attention מייצר וקטור חדש עבור כל מילה בסדרת הפלט, כאשר כל רכיב בווקטור מכמת עד כמה המילה הנוכחית במוצא קשורה לכל אחת מהמילים במשפט המקורי. באופן הזה כל איבר בסדרת הפלט ממשקל כל אחד מאיברי סדרת הקלט. מנגנון זה נקרא attention.



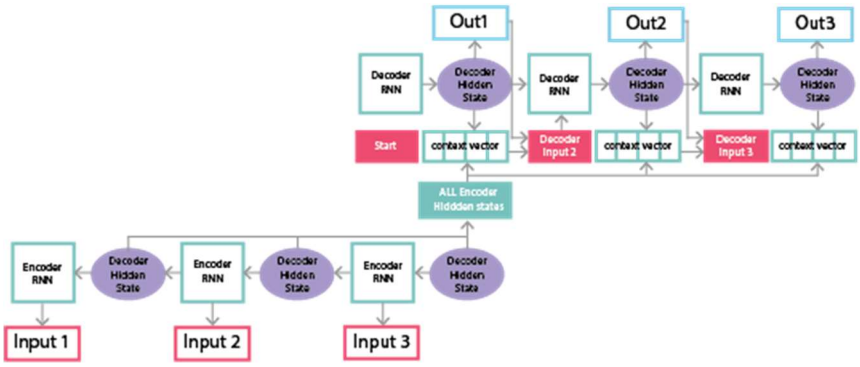
איור 8.1 מנגנון attention – נתינת משקל לכל אחת ממילות הקלט ביחס לכל אחת ממילות הפלט.

במאמר משנת 2017 שנקרא "Attention is All You Need", הוצע להשתמש ב-attention בלבד ללא רשתות מסוג LSTM או GRU, ומאמר זה פרץ דרך לשימושים רבים במנגנון זה תוך קבלת ביצועים מעולים.

בחלק זה יוצגו הגישות המשלבות בין רשתות RNN לבין attention ולאחר מכן יוסבר על ה-transformer שמשתמש ב-self-attention וב-positional encoding בנוסף ל-attention הרגיל, ובכך מייתר את הצורך ברכיבי זיכרון.

8.1.2 Bahdanau Attention and Luong Attention

הגישה הראשונה שהוצעה נקראת Bahdanau Attention על שם הממציא שלה – Dzmitry Bahdanau.



איור 8.2 ארכיטקטורת Bahdanau Attention.

הרעיון של גישה זו היא לבנות ארכיטקטורה בה משתמשים בכל המצבים החבויים של רכיבי הזיכרון ב-encoder ומעבירים אותם ל-decoder. כתוצאה מכך ה-decoder מחשב את המוצא לא רק על סמך מצביו הקודמים, אלא משקלל אותם יחד עם המצבים החבויים של ה-encoder. עבור כל אחד מאיברי סדרת הפלט מחשבים alignment

score בין המצב החבוי של רכיב הזיכרון הקודם בסדרת הפלט לבין כל המצבים החבויים של ה-encoder, וכך יוצרים context vector שבעזרתו מחשבים את הפלט עבור האיבר הנוכחי ב-decoder. ביצוע הפעולה הזו הוא הלב של מנגנון ה-attention, כיוון שהוא קושר בין הקלט לפלט, ובנוסף מחשב עבור כל איבר של סדרת קלט כמה משקל יש לתת לכל אחד מאיברי הקלט האחרים.

ביצוע פעולה זו יוצרת לכל אחד מאיברי הפלט context vector ייחודי משלו הנבנה גם מהמצב הקודם וגם מאיברי ה-encoder, בשונה מהארכיטקטורות הקודמות של seq2seq בהן לא היה ניתן להעביר מידע באופן ישיר מהמצבים החבויים של ה-encoder ל-decoder. את ה-context vector מחברים למוצא של האיבר הקודם ב-decoder, ויחד עם המצב החבוי הקודם יוצרים את המצב החבוי הבא, שבעזרתו מוצאים את הפלט של האיבר הנוכחי.

באופן פורמלי, אם נסמן ב-H_e, H_d את המצבים החבויים של ה-encoder וה-decoder, ה-alignment score יתקבל על ידי:

$$\text{alignment score} = w_{alignment} \times \tanh\left(w_d H_d + w_e H_e\right)$$

כאשר $w_{alignment}, w_d, w_e$ הם המשקלים הנלמדים של ה-encoder, ה-decoder והחיבור ביניהם. את התוצאה מעבירים דרך SoftMax, מכפילים בH_e ומקבלים את ה-context vector:

$$\text{context vector} = H_e \times \text{SoftMax}(\text{alignment score})$$

הווקטור המתקבל מכיל משקל של כל אחד מאיברי הקלט ביחס לאיבר הפלט הנוכחי. את התוצאה כאמור מחברים לפלט של האיבר הקודם, ובעזרת המצב החבוי הקודם מחשבים את המצב החבוי הנוכחי, שממנו מחלצים את הפלט של האיבר הנוכחי.

ישנו שיפור של Bahdanau attention הנקרא Loung attention. שני הבדלים עיקריים יש בין שני המנגנונים: חישוב ה-alignment score מתבצע באופן שונה, ובנוסף בכל שלב לא משתמשים במצב החבוי הקודם של ה-decoder כמו שהוא אלא יוצרים מצב חבוי חדש ובעזרתו מחשבים את ה-alignment score.

8.2 Transformer

לאחר שמנגנון ה-attention התחיל לצבור תאוצה, הומצאה ארכיטקטורה המבוססת על attention בלבד ללא שום רכיבי זיכרון. ארכיטקטורה זו הנקראת transformer מציעה שני אלמנטים חדשים על מנת למצוא קשרים בין איברים בסדרה מסוימת – positional encoding ו-self-attention.

8.2.1 Positional Encoding

ארכיטקטורות מבוססות RNN משתמשות ברכיבי זיכרון בשביל לקחת בחשבון את הסדר של האיברים בסדרה. גישה אחרת לייצוג הסדר בין איברי הסדרה נקראת positional encoding, בה מוסיפים לכל אחד מאיברי הקלט פיסת מידע לגבי המיקום שלה בסדרה, והוספה זו כאמור באה כתחליף לרכיבי הזיכרון ברשתות RNN. באופן פורמלי, עבור סדרת קלט $x \in \mathbb{R}^d$, מחשבים וקטור בממד $d \times 1$ באופן הבא:

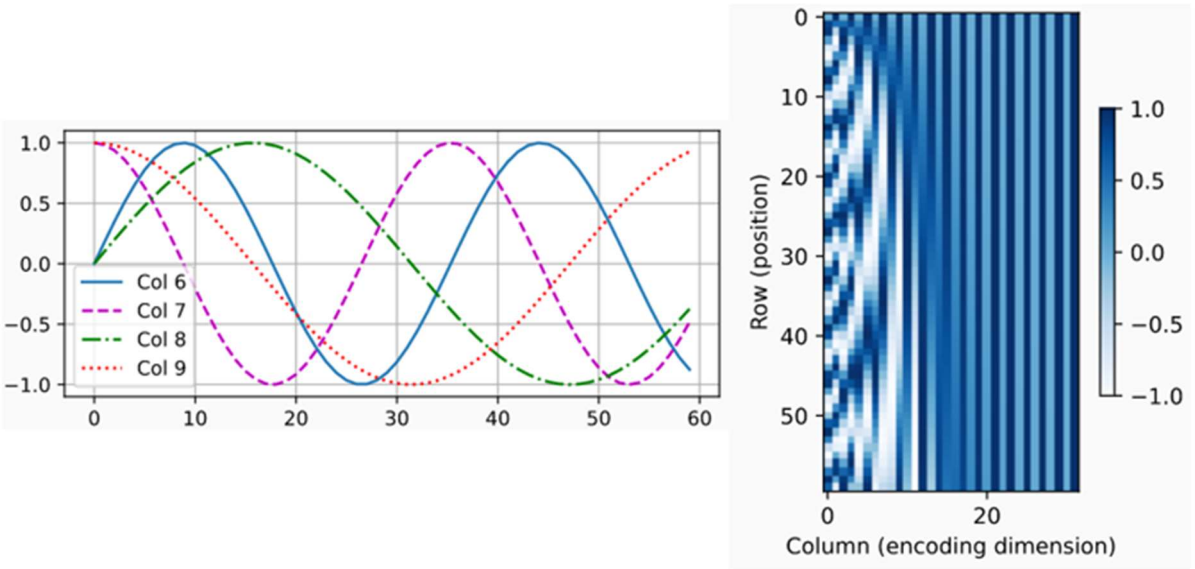
$$p_t(i) = \begin{cases} \sin(\omega_k, t), & i \text{ is even} \\ \cos(\omega_k t), & i \text{ is odd} \end{cases}, \omega_k = \frac{1}{10000^{\frac{2k}{d}}} \rightarrow p_t = \begin{bmatrix} \sin(\omega_1 t) \\ \cos(\omega_1 t) \\ \sin(\omega_2 t) \\ \cos(\omega_2 t) \\ \vdots \\ \sin\left(\omega_{\frac{d}{2}} t\right) \\ \cos\left(\omega_{\frac{d}{2}} t\right) \end{bmatrix}_{d \times 1}$$

בכדי להבין כיצד וקטור זה מכיל משמעות של סדר בין דברים, נציג את הרעיון שהוא מייצג בצורה יותר פשוטה. אם נרצה לקחת רצף של מספרים ולייצג אותם בצורה בינארית, נוכל לראות שככל שלביט יש משקל גדול יותר, כך הוא משתנה בתדירות נמוכה יותר, ולמעשה תדירות שינוי הביט היא אינדיקציה למיקום שלו:

0:	0	0	0	0	8:	1	0	0	0
1:	0	0	0	1	9:	1	0	0	1
2:	0	0	1	0	10:	1	0	1	0
3:	0	0	1	0	11:	1	0	1	1
4:	0	1	0	0	12:	1	1	0	0
5:	0	1	0	1	13:	1	1	0	1
6:	0	1	1	0	14:	1	1	1	0
7:	0	1	1	1	15:	1	1	1	1

איור 8.3 ייצוג בינארי של מספרים. ה-MSB משתנה בתדירות הכי נמוכה, ואילו ה-LSB משתנה בתדירות הכי גבוהה.

כיוון שמתעסקים במספרים שאינם בהכרח שלמים, הייצוג הבינארי של מספרים שלמים הוא יחסית בזבזני, ולכן כדאי לקחת גרסה רציפה של אותו רעיון – פונקציות טריגונומטריות עם תדירות הולכת וגדלה. זהו בעצם הווקטור p – הוא מכיל הרבה פונקציות טריגונומטריות בעלות תדירות הולכת וקטנה, ולפי התדירות שמתווספת לכל איבר בסדרה המקורית ניתן לקבל אינדיקציה על מיקומו.



איור 8.4 Positional encoding. דוגמא למספר פונקציות בעלות תדירות הולכת וקטנה, בהתאם לאיבר אותן הן מייצגות (שמאל). המחשה לקצב השינוי של כל פונקציה בהתאם למיקום של האיבר אותו היא מייצגת – מעין גרסה רציפה לקצב שינוי הביטים בייצוג בינארי של מספרים שלמים (ימין).

ישנו יתרון נוסף שיש לשימוש בפונקציות הטריגונומטריות – עבור כל צמד פונקציות בעלות אותו תדר $\begin{bmatrix}\sin(\omega_k t)\\ \cos(\omega_k t)\end{bmatrix}$, ניתן לבצע טרנספורמציה לינארית ולקבל תדר אחר(Relative Positional Information):

$$M \cdot \begin{bmatrix}\sin(\omega_k t)\\ \cos(\omega_k t)\end{bmatrix} = \begin{bmatrix}\sin(\omega_k t+\phi)\\ \cos(\omega_k t+\phi)\end{bmatrix}, M = \begin{bmatrix}\cos(\omega_t \phi) & \sin(\omega_t \phi)\\ -\sin(\omega_t \phi) & \cos(\omega_t \phi)\end{bmatrix}$$

באופן הזה מקבלים באופן מידי ייחוס בין כל ה-positions, מה שיכול לעזור בניתוח הקשרים שבין איברים שונים.

8.2.2 Self-Attention Layer

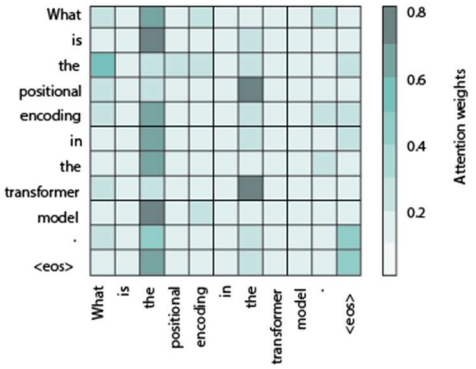
בנוסף ל-positional encoding, עלה הרעיון לבצע attention לא רק בין איברי הקלט לאיברי הפלט, אלא גם בין איברי הקלט בעצמם. הרעיון הוא לייצר ייצוג חדש של סדרת הקלט באותו אורך כמו הסדרה המקורית, כאשר כל איבר בסדרה החדשה ייצג איבר בסדרה המקורית בתוספת מידע על הקשר שלו לשאר האיברים. הרעיון הכללי אומר שיש לקחת כל איבר בסדרה, ולחשב את הדמיון שלו לאר האיברים בסדרה. איברים דומים (קרובים) בסדרה יקבלו ערכי דמיון גבוהים, ואילו איברים שונים (רחוקים) בסדרה יתנו ערכים נמוכים (ב-NLP זה יכול להיות מילים שסביר שיופיעו בסמיכות, ובתמונה זה יכול להיות פיקסלים דומים). דמיון בין איברים נמדד על פי הקשר שיש ביניהם, והוא מחושב באמצעות מכפלה פנימית בין וקטורי ייצוג של האיברים. כל מכפלה פנימית בין שני איברים נותנת מקדם שהוא מספר ממשי, וכך ניתן לסכם את מכפלת כל המקדמים באיברים המקוריים, ולקבל ייצוג חדש לאיבר המקורי המכיל גם קשר בין האיבר הנוכחי לאיברים דומים בסדרה. במילים אחרות, ניתן להסתכל על וקטור המכיל את הקשרים של איבר מסוים בסדרה כייצוגו החדש המשקף את קשריו עם שאר איברי הסדרה.

באופן פורמלי, בשביל לחשב את ה-self-attention יוצרים שלוש מטריצות של מקדמים עבור סדרת הכניסה. המטריצות נקראות Query, Key, Value, כאשר כל אחת מהן נוצרת על ידי הכפלה של מטריצת משקלים באיבר הקלט. בעזרת מטריצות אלו מחשבים את ה-attention score:

$$\text{Attention}(\text{Query}, \text{Key}, \text{Value}) = \text{SoftMax}\left(\frac{\text{Q} \cdot \text{K}}{\sqrt{d_k}}\right) \cdot \text{V}$$

כדי להבין כיצד הנוסחה הזו מסייעת במציאת קשר בין איברים, נבחן כל איבר שלה בנפרד. עבור סדרת קלט x מקבלים שלוש מטריצות, כאשר כל איבר בסדרה המקורית x_i יוצר שורה בכל אחת מהמטריצות. כאשר לוקחים את השורה $q_i = \text{Q} \cdot x_i$, ומכפילים אותה בכל אחת מהשורות במטריצה K, מקבלים וקטור חדש, שכל איבר j בוקטור אומר עד כמה יש קשר בין האיברים i, j בסדרה המקורית. ביצוע ההכפלה הזו עבור כל סדרת הקלט יוצר מטריצה חדשה בה כל שורה מייצגת את הקשר בין איבר מסוים לשאר איברי הסדרה. ההכפלה הזו היא בעצם $\text{Q} \cdot \text{K}$, כאשר כל מכפלה $q_i^T k_j$ מייצגת את הקשר בין האיבר i לאיבר j. את התוצאה מחלקים בשורש של ממד ה-embedding כדי לשמור על יציבות הגרדיאנט, ולאחר מכן מנרמלים על ידי SoftMax. באופן הזה מקבלים מטריצה של מספרים בטווח $[0, 1]$, המייצגים כאמור את הקשר בין כל שני איברים בסדרה המקורית. נסמן כל איבר במטריצה ב-w_{ij}, ונוכל לקבל אותו ישירות על ידי הנוסחה:

$$w_{ij} = \text{SoftMax}\left(\frac{q_i \cdot k_j}{\sqrt{d_k}}\right) = \frac{\exp\left(\frac{q_i^T k_j}{\sqrt{d_k}}\right)}{\sum_{s=1}^{n} \exp\left(\frac{q_i^T k_s}{\sqrt{d_k}}\right)}$$

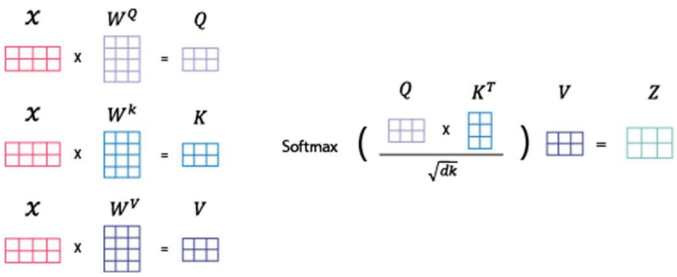


איור 8.5 מטריצת משקלים של המשפט "What is the positional encoding in the transformer model". ככל שקשר בין שתי מילים חזק יותר, כך המשקל ביניהם גבוה יותר. כמובן שיש גם משמעות לסדר – המשקל בין "is" ל-"positional" שונה מהמשקל שבין "model" ל-"is".

כעת בעזרת משקלים אלו בונים ייצוג חדש לסדרה המקורית, על ידי הכפלתם בוקטור V:

$$z_i = \sum_{j=1}^{n} w_{ij} v_j = \frac{\sum_{j=1}^{n} \exp(q_i^T k_j)}{\sum_{s=1}^{n} \exp(q_i^T k_s)} v_j$$

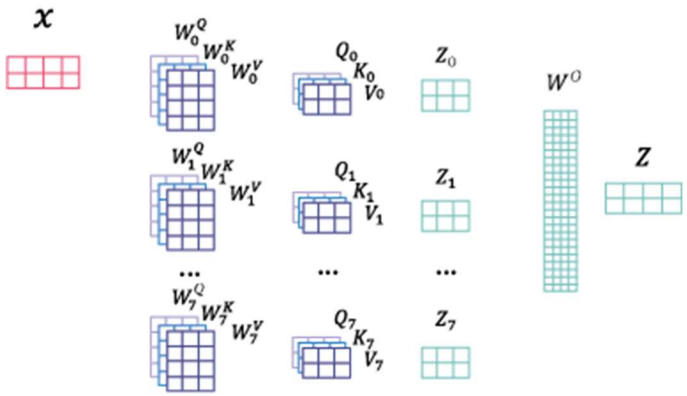
הסדרה המתקבלת z היא למעשה ייצוג חדש של הסדרה המקורית, כאשר כל איבר z_i מייצג איבר בסדרה המקורית יחד עם מידע על הקשרים בינו לבין שאר איברי הסדרה. את הסדרה המתקבלת ניתן להעביר ב-decoder המכיל שכבות נוספות, ובכך לבצע כל מיני משימות, כפי שיוסבר בהמשך.



איור 8.6 ביצוע Self-attention – יצירת מטריצות Query, Key, Value (שמאל) וחישוב ה-attention score (ימין).

8.2.3 Multi Head Attention

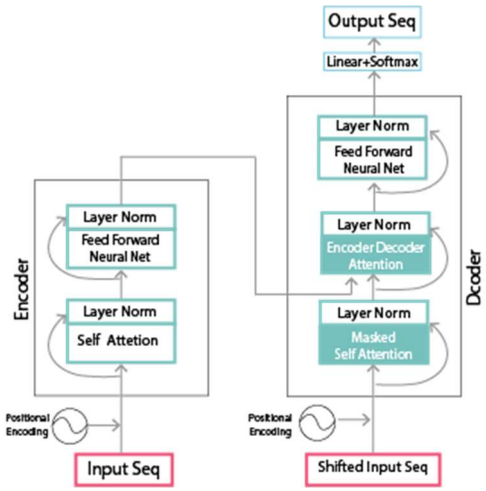
ניתן להשתמש במנגנון self-attention מספר פעמים במקביל. כל פעם מקבלים שלשת מטריצות (Q_r, K_r, V_r), ובעזרתה מחשבים את הייצוגים החדשים של איברי הסדרה (attention score). כל מנגנון כזה נקרא attention head, וחיבור במקביל של כמה attention heads נקרא Multi-head attention. באופן הזה לכל איבר כניסה x_i יש כמה ייצוגים שונים z_{ir}, אותם ניתן להכפיל במטריצת משקלים w_o ולקבל את הייצוג המשוקלל של אותו איבר באמצעות attention heads מרובים.



איור 8.7 Self-attention with 8 heads.

8.2.4 Transformer End to End

בעזרת מנגנוני multi head attention ו-positional encoding ניתן לבנות Transformer – ארכיטקטורה עבור סדרות המבוססת רק על attention ללא רכיבי זיכרון.



איור 8.8 Transformer.

כפי שניתן לראות באיור ה-transformer מורכב משני חלקים – encoder ו-decoder. ה-encoder מקבל סדרה מסוימת x (לרוב אחרי שעברה embedding מסוים) ומבצע עליה positional encoding. לאחר מכן הסדרה עוברת דרך residual block של self-attention, ומתקבלת התוצאה $x + \text{attention}(x)$. על תוצאה זו מבצעים layer normalization (כפי שהוסבר בפרק 5.1.4). לאחר מכן יש residual block נוסף, המכיל שכבת fully connected, ומשם יוצא הפלט לכיוון ה-encoder.

ה-decoder בנוי בצורה מאוד דומה, עם שני הבדלים עיקריים: הקלט שלו הוא איברי הפלט שהיו עד כה, ובנוסף יש בתחילת ה-decoder שכבה של Masked self-attention. שכבה זו מקבלת את כל איברי הפלט שהיו עד כה, ומטרת ה-decoder היא ללמוד בעצמו מה האיבר הבא של הפלט. בשלב הראשון ה-decoder מבצע self-attention על איברי הסדרה שהתקבלו עד כה, וכך לומד ייצוג חדש שלהם, המכיל גם את הקשר בין איברי סדרה זו.

לאחר השכבה הראשונה יש שכבת multi head attention נוספת הנקראת Encoder-Decoder Attention, כיוון שהיא שילוב של ה-encoder וה-decoder: המטריצות Key, Value נלקחות מה-encoder וה-Query מגיע מה-decoder. כעת כשמבצעים את המכפלה $Q \cdot K$, לא מחפשים דמיון בין איברים של אותה סדרה אלא בין האיברים של סדרת הפלט (בייצוג שלהם לאחר ה-encoder) לבין איברי סדרת הקלט (בייצוג שלהם לאחר שכבת ה-masked). שלב זה דומה מאוד ל-attention המקורי, רק שהייצוגים שהתקבלו לא נעזרו ברכיבי זיכרון. כאמור, המכפלה $Q \cdot K$ מייצרת מטריצת משקלים שכל איבר בה אומר מה היחס בין איבר בסדרה המקורית לבין איבר בסדרת הפלט. את המטריצה הזו מכפילים ב-V וכך מקבלים וקטור מסוים שהוא ייצוג חדש של איבר הפלט הבא. וקטור זה עובר בשכבת FC וב-SoftMax, וכך מתקבל איבר הפלט.

ניקח דוגמא ממאמר שנקרא DETR המראה כיצד ניתן להשתמש ב-transformer בשביל זיהוי אובייקטים בתמונה. בשלב הראשון לוקחים כל פיקסל בתמונה ומשווים אותו לשאר הפיקסלים (זוהי בעצם המכפלה $Q \cdot K$). באופן הזה ניתן למצוא אזורים דומים ושונים בתמונה, כאשר דמיון ושוני זה לאו דווקא פיקסלים עם ערכים קרובים, אלא זה יכול להיות למשל שני אזורים שונים בפנים של אדם. לאחר מכן מייצרים ייצוג חדש לתמונה, בעזרת המשקלים והכפלתם ב-V. שלב זה למעשה מאפשר לבצע זיהוי של אובייקטים, בלי לדעת מה הם אותם אובייקטים. בשביל לבצע סיווג לכל אובייקט שזוהה, מעבירים את הייצוג החדש של התמונה ב-decoder, כאשר ה-Query שמכניסים זה כל מיני לייבלים אפשריים, ומחפשים מבין כל ה-Query את הפלט של ה-decoder שמצליח לייצר תמונה שהכי דומה ל-Query.

אם למשל יש תמונה גדולה ויש אזור מסוים בו יש חתול, אז ה-encoder מוצא איפה החתול בתמונה, וה-decoder משווה את האזור הזה לכל מיני חיות אפשריות. כל Query שלא יהיה חתול, המכפלה $Q \cdot K$ תהיה קרובה ל-0, וה-decoder יזהה שה-Query הנוכחי לא תואם לאובייקט שזוהה. אך כאשר ה-Query יהיה חתול, אז כיוון ש-Q, K דומים אחד לשני, המכפלה $Q \cdot K$ תביא לכך שהייצוג החדש $z_i = \sum_{j=1}^{n} w_{ij} v_j$ כן יהיה דומה לחתול. ייצוג זה עובר בשכבת FC, ולאחר מכן ה-SoftMax יסווג את התמונה הזו כחתול.

8.2.5 Transformer Applications

ה-transformer הציג ביצועי state-of-the-art במגוון משימות, והוא היווה השראה להמון יישומים הנשענים על attention בלבד. מלבד הרמה הגבוהה של הביצועים, תהליך האימון של transformer הוא הרבה יותר מהיר מרשתות קונבולוציה או רשתות רקורסיביות. כמו במודלים אחרים, גם עם transformer ניתן לבצע transfer learning, כלומר לקחת transformer שאומן על משימה מסוימת, ולהתאים אותו למשימה חדשה שדומה למשימה המקורית. בפועל לא כל היישומים משתמשים בכל ה-transformer, אלא בהתאם למשימה לוקחים חלקים מסוימים שלו ובונים בעזרתם מודל עבור משימה מסוימת. נביא מספר דוגמאות:

Machine Translation – תרגום משפטים בין שפות שונות הוא יישום טריוויאלי של ה-transformer המלא. המשימה היא לקחת משפט ולהוציא משפט בשפה אחרת, וזה נעשה בעזרת ייצוג המשפט המקורי באופן חדש בעזרת self-attention ולאחר מכן המרתו בעזרת Encoder-Decoder Attention לשפה אחרת.

Bidirectional Encoder Representations from Transformers (BERT) – מודל שפה מבוסס encoder. מודל שפה הוא פונקציה המקבלת כקלט טקסט ומחזירה את ההתפלגות למילה הבאה על פי כל המילים במילון. השימוש הכי מוכר ואינטואיטיבי של מודל שפה הוא השלמה אוטומטית, שמציעה את המילה או המילים הכי סבירות בהינתן מה שהמשתמש הקליד עד כה. כאשר מבצעים self-attention על משפטים, למעשה מקבלים ייצוגים חדשים שלהם יחד עם ההקשרים בין המילים השונות. לכן ה-encoder ב-transformer יכול ליצור מודל שפה, אם מאמנים אותו בצורה מתאימה. המפתחים של BERT בנו encoder המקבל כל מיני משפטים בשני כשני הכיוונים – גם מההתחלה לסוף וגם מהסוף להתחלה, וכך הייצוגים שנלמדו קיבלו קונטקסט שלם יותר. בנוסף, הם אימנו את המודל על משפטים בהם כל פעם באופן רנדומלי עושים masking למילים מסוימות, ומטרת המודל הוא לחזות את המילים החסרות.

Generative Pre-Training (GPT) – מודל לחיזוי המילה הבאה במשפט. ניתן לקחת משפט שקטוע באמצע, ולבחון מהי המילה הבאה באמצעות decoder בלבד. מכניסים משפט קטוע ל-decoder ואז עוברים על המון מילים ובודקים את ההתאמה שלהן למשפט הנתון, והמילה שהכי מתאימה נבחרת להיות המילה הבאה. המשפט הקטוע הוא למעשה ה-Key, וה-Query שנכנס הוא כל פעם מילה אחרת במילון, וכך בעזרת attention בוחנים איזה Query יתאים בצורה הטובה ביותר ל-Key הנתון.

References

https://arxiv.org/abs/1409.0473

https://arxiv.org/abs/1706.03762

https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263

https://towardsdatascience.com/transformer-attention-is-all-you-need-1e455701fdd9

https://arxiv.org/abs/1810.04805