NLP – assignment 5

Koren Abitbul - 318796448, Pan Eyal - 208722058

שאלה 1

1.1

המרכיבים השונים הם השאילתה (Q), המפתח (K), ערך (V).

מנגנון הצומי העצמי מחשב את הscore בין השאילתה (Q) לכל מפתח (K).

על ידי: $score(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$ כאשר: $q_i = x_i W^Q, k_i = x_i W^K$.

ציון זה מציין עד כמה כל מילה אחרת (Key) רלוונטית למילה הנוכחית (Query).

כך, בהינתן קשת התלות הרלוונטית $w_i \to w_j$, נסתכל פשוט על הscore: $score(w_i, w_j)$.

1.2

ייתכן שנמצא קשרי צומי חזקים יותר בשכבות שונות של הרובוטריק.

תכונה אפשרית ראשונה: מרחק התלות.
בשכבות המוקדמות יותר של הרובוטריק המודל עשוי ללכוד תלויות של מילים שקרובות יותר זו לזו במשפט.
ולכן תלויות המילים הרחוקות זו מזו לא יהיו חזקות בשכבות הראשונות כמו בעמוקות יותר.

תכונה אפשרית שניה: מורכבות התלות.
תלות מורכבת יותר עשויה לחייב את המודל לשלב מידע מחלקים שונים של המשפט. קשרים אלה עשויים להופיע בשכבות מאוחרות יותר, בהן הרובוטריק עיבד מידע דרך שלבי צומי עצמי מרובים ויכול להתמודד עם תלות בטווח ארוך יותר.

1.3

בעזרת האלגוריתם החמדן לניתוח תלויות במעברים, לאחר שנבצע forward pass על הרובוטריק, נשיג את הscore עבור הצומי העצמי בין כל צמדי המילים עבור כל השכבות. נוכל לבדוק את הscore עם שאר המילים לפי שכבות שונות ולאט לאט לבנות את התלויות התחביריות.

שאלה 2

$$logit(p) = \ln\left(\frac{p}{1-p}\right)$$

2.1

נוכיח שהפונקציה הזאת היא ההופכית של פונקציית הסיגמואיד.

עבור פונקציית הסיגמואיד $\sigma(x) = \frac{1}{1+e^{-x}}$ נסתכל על $\sigma(x) = y$ כאל הפלט של הסיגמואיד בהינתן הקלט x ונחפש פונקציה $f(y) = x$ ש x הוא הפלט שלה בהינתן y:

$$\frac{1}{1+e^{-x}} = y$$

$$1 + e^{-x} = \frac{1}{y}$$

$$e^{-x} = \frac{1}{y} - 1$$

$$\ln\left(e^{-x}\right) = \ln\left(\frac{1}{y} - 1\right)$$

$$-x = \ln\left(\frac{1-y}{y}\right)$$

$$x = -\ln\left(\frac{1-y}{y}\right)$$

$$x = \ln\left(\frac{y}{1-y}\right) = f(y) = logit(y)$$

2.2

נעשה את אותו הדבר עבור פונקציית הsoftmax; $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{|V|} e^{z_j}}$,

$$\frac{e^{z_i}}{\sum_{j=1}^{|V|} e^{z_j}} = y_i$$

$$\ln\left(\frac{e^{z_i}}{\sum_{j=1}^{|V|} e^{z_j}}\right) = \ln(y_i)$$

$$\ln(e^{z_i}) - \ln\left(\sum_{j=1}^{|V|} e^{z_j}\right) = \ln(y_i)$$

$$z_i = \ln\left(\sum_{j=1}^{|V|} e^{z_j}\right) + \ln\left(y_i\right)$$

קיבלנו תלות ב-e^{z_j}, לכן אין לפונקציה הופכי. וזה הגיוני כי לדוגמה אם בsoftmax היינו מקבלים התפלגות אחידה, אז הלוגיטים יכולים להיות כל וקטור אשר הערכים שלו הם זהים.
בשביל לחשב את הלוגיטים יש צורך בלדעת את הערך: $\sum_{j=1}^{|V|} e^{z_j}$

2.3

$$\frac{e^{(z_i-\tau)}}{\sum_{j=1}^{|V|} e^{(z_j-\tau)}} = y_i$$

$$\ln\left(\frac{e^{(z_i-\tau)}}{\sum_{j=1}^{|V|} e^{(z_j-\tau)}}\right) = \ln(y_i)$$

$$\ln\left(e^{(z_i-\tau)}\right) - \ln\left(\sum_{j=1}^{|V|} e^{(z_j-\tau)}\right) = \ln(y_i)$$

$$z_i = \ln\left(\sum_{j=1}^{|V|} e^{(z_j-\tau)}\right) + \ln(y_i) + \tau$$

בשביל לחשב את הלוגיטים יש צורך בלדעת את הערך: $\sum_{j=1}^{|V|} e^{(z_j-\tau)}$

2.4

אם ניקח את k האסימונים המובילים ישירות מהלוג'יטים, ולאחר מכן ננרמל בעזרת הsoftmax הדבר יחזיר לנו תוצאה זהה ללקיחת k האסימונים מהsoftmax ולאחר מכן לנרמל אותם.

מכיוון שפעולת הsoftmax היא מונוטונית, בשתי הדרכים נקבל את אותם רכיבים מובילים.
כלומר, עבור לוג'יטים z, אם $z_l \in top_k(z)$ אז גם $\sigma(z)_l \in top_k\big(\sigma(z)\big)$.

כעת, עבור הערך בהתפלגות שלהם נראה:
עבור אוצר מילים V ולקיחת האסימונים מה*softmax*:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{|V|} e^{z_j}}$$

נסמן $l, m \in \{1, \dots, k\} = top_k$ כך ש$\sigma(z)_l$ ו- $\sigma(z)_m$ שייכים לtop_k.
ההסתברויות שנקבל לאחר נרמול הן:

$$\sigma(z)_l = \frac{\frac{e^{z_l}}{\sum_{j=1}^{|V|} e^{z_j}}}{\sum_{m=1}^{k} \sigma(z)_m} = \frac{\frac{e^{z_l}}{\sum_{j=1}^{|V|} e^{z_j}}}{\sum_{m=1}^{k} \frac{e^{z_m}}{\sum_{j=1}^{|V|} e^{z_j}}} = \frac{e^{z_l}}{\sum_{j=1}^{|V|} e^{z_j} \cdot \sum_{m=1}^{k} \frac{e^{z_m}}{\sum_{j=1}^{|V|} e^{z_j}}} = \frac{e^{z_l}}{\sum_{m=1}^{k} \frac{e^{z_m} \cdot \sum_{j=1}^{|V|} e^{z_j}}{\sum_{j=1}^{|V|} e^{z_j}}}$$

$$= \frac{e^{z_l}}{\sum_{m=1}^{k} e^{z_m} \cdot 1} = \sigma(z_{\leq k})_l$$

כאשר $z_{\leq k}$ הוא הווקטור המכיל את k הערכים הגבוהים של ווקטור z.

כלומר, קיבלנו את מה שרצינו להראות שמתקיים שווין.

ישנם טעויות בהם כתוב פעמיים "the" כאן:

word. For example, if the preceding context is *"Thanks for all the"* and we want to know how likely the next word is *"fish"* we would compute:

$$P(fish|Thanks\ for\ all\ the)$$

Language models give us the ability to assign such a conditional probability to every possible next word, giving us a distribution over the entire vocabulary. The n-gram language models of Chapter 3 compute the probability of a word given counts of its occurrence with the $n-1$ prior words. The context is thus of size $n-1$. For transformer language models, the context is the size of the transformer's context window, which can be quite large: up to 2048 or even 4096 tokens for large models.

The job of the language modeling head is to take the the output of the final transformer layer from the last token N and use it to predict the upcoming word at position $N+1$. Fig. 10.13 shows how to accomplish this task, taking the output of the last token at the last layer (the d-dimensional output embedding of shape $[1 \times d]$) and producing a probability distribution over words (from which we will choose one to generate).

וכאן:

5. Randomly sample a word from within these remaining k most-probable words according to its probability.

When $k = 1$, top-k sampling is identical to greedy decoding. Setting k to a larger number than 1 leads us to sometimes select a word which is not necessarily the most probable, but is still probable enough, and whose choice results in generating more diverse but still high-enough-quality text.

10.8.2 Nucleus or top-p sampling

One problem with top-k sampling is that k is fixed, but the shape of the the probability distribution over words differs in different contexts. If we set $k = 10$, sometimes the top 10 words will be very likely and include most of the probability mass, but other times the probability distribution will be flatter and the top 10 words will only include a small part of the probability mass.

עוד קצת כפילויות של מילים:

Fig. 10.8 shows a visualization of this movement.

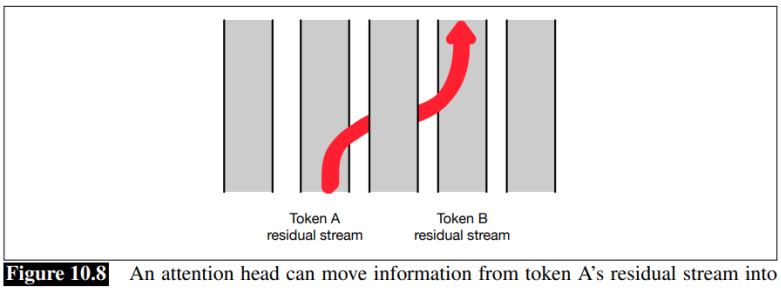

Figure 10.8 An attention head can move information from token A's residual stream into token B's residual stream.

Equation (10.32) and following are just just the equation for a single transformer block, but the residual stream metaphor goes through all the transformer layers, from the first transformer blocks to the 12th, in a 12-layer transformer. At the earlier transformer blocks, the residual stream is representing the current token. At the highest transformer blocks, the residual stream is usual representing the following token, since at the very end it's being trained to predict the next token.

כאן אנחנו חושבים שזה צריך להיות במקום usual, usually (?)

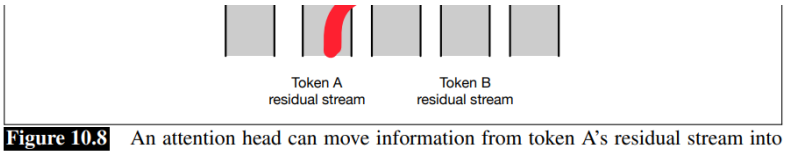

Figure 10.8 An attention head can move information from token A's residual stream into token B's residual stream.

Equation (10.32) and following are just just the equation for a single transformer block, but the residual stream metaphor goes through all the transformer layers, from the first transformer blocks to the 12th, in a 12-layer transformer. At the earlier transformer blocks, the residual stream is representing the current token. At the highest transformer blocks, the residual stream is usual representing the following token, since at the very end it's being trained to predict the next token.

וכאן יש היפר-לינק כושל ביותר:

as defined by the model. Thus we are more likely to generate words that the model thinks have a high probability in the context and less likely to generate words that the model thinks have a low probability.

We saw back in Chapter 3 on page **??** how to generate text from a unigram lan-