Last Day: Project + Introduction to Transformers
A2 due today

- team formation due today
- approval-in-principle due Oct 27

Today: The Core Mechanisms of Transformers & Assignment 3

Recall: When training a Transformer from scratch, we train it to be a language model: given a sequence of \( n \) words, predict the probability of each word in the vocabulary, being the next word.

\[ \text{Input: The smooth blue \underline{lake} became \textit{choppy} in the wind} \]
\[ \text{To: \hspace{1cm} Output sequence of words \underline{label}} \]

E.g. Training Example 3: The smooth blue \underline{lake} became \textit{choppy}

E.g. Training Example 4: The smooth blue \underline{lake} became \textit{choppy}

E.g. Training Example 5: The smooth blue \underline{lake} became \textit{choppy}

- lots of training data! everything ever written

- here is the global structure of a Transformer

\[
\begin{array}{c}
\text{Input Word Vectors} \\
\text{Transformer Block to} \\
\text{Output:} \\
\text{Language} \\
\text{Head Fully Connected MLP} \\
\end{array}
\]

\[
P(W_0), P(W_1), \ldots, P(W_m)
\]

- even though \( n \) words/tokens always goes in to predictor, the ones after the final input length \( n_f \) are ignored = "Causal" language model

- done as part of the set-up
- Now, here is what is in each transformer block $T_i$:
- recall it is the same size in as out.

 skeptical aside: we could ignore the inner structure of and just say \"it is a big network with many parameters\"
- it may be, to my mind, the underlying truth, e maybe a large MLP or CNN could do this same thing. (just bigger is better)
- it may be that the particular structure is better.
- will present the structure \& understanding per Jurafsky.
- however, the explanations only work for block $T_0$, to me.

- Here is the inner structure of one Transformer Block

[Diagram of a Transformer block with labeled components: multi-head self-attention, Layer Normalization, \"feed forward\", and fully connected MLP.]
Attention

Here is the core intuition given of what Transformer Block is doing:

- the input word embeddings are transformed from their initial, very general meanings (across all uses/contexts of the word) to something more specific to the context (i.e., the other words in sequence)

- they call these "contextual embeddings"

- e.g., the embedding for "bank" would become different in these contexts:
  - she saved her money in the bank
  - he emptied his bank account
  - river bank
  - they should not bank on the result...

- From , consider the adapt . Yes, which we say is a transformation of the input word.

  e.g., The bank took his account.

- Self attention asks the question: how similar is to all preceding words and itself? [See Section 9.7]

  e.g., how similar is to ?
  - how similar is to ?
  - how similar is to ?
  - how similar is to ?
  - how similar is to ?
- how have we computed a number that says how similar/related two words are? 
  \[ \Rightarrow \text{use dot product of word embeddings} \]
  Compute \( x_0 \cdot x_3 \)
  Define \( x_1 \cdot x_3 \) \[ \text{score}(x_i, x_j) = x_i \cdot x_j \]
  \( x_2 \cdot x_3 \) \[ \Rightarrow \text{but need to normalize} x_3 \cdot x_3 \]
  \[ \text{if}, \text{so } \]

- [do this for every word]

- when computing the relative similarity of \( x_i \) to everything that comes before, we calculate

\[
C_{ij} = \text{softmax} \left( \text{score}(x_i, x_j) \right) \quad \forall j \leq i
\]

\[
= \frac{\exp \left( \text{score} \left( x_i, x_j \right) \right)}{\sum_{k=0}^{i} \exp \left( \text{score} \left( x_i, x_k \right) \right)}
\quad \forall j \leq i
\]

- only previous \[
\text{correct word}
\]
- This score, $d_{ij}$, gives the relative importance of $x_j$ to $x_i$, and we use it to compute a new embedding, $y_i$, that combines different proportions of the $x_j$ like so:

$$y_i = \sum_{j \neq i} d_{ij} x_j$$

We are adding a fraction of the meaning of those other words into the original embeddings.

This is how "bank," gets more "river" in it.

"Contextual embeddings" makes sense on 1st layer.

Now notice that there are no learned parameters so far.

- Also notice that $x_i$ gets used in 3 ways:
  1. as the focus $x_i$ in score $(x_i, x_j)$
  2. as the "searched" $x_j$ in score $(x_i, x_j)$
  3. to compute $y_i$ in above

- For all 3 cases, we will insert learned parameters to modify these three uses as follows, e.g., for $x_i$:

$$q_i = W^a x_i$$

- This is a ML trick.

- Think of $W^a$ like a Conv Kernel. Something learned across the banking layer.

- Will (to have multiple)

- If you multiply this out, you'll see that $q_i$ is the same vector size as $x_i = d$.

- But it has been projected/transferred by $W^a$.

- The $W_pq_i$ are learned parameters that you need to der.
- Similarly, there are two other learned 2d matrices \( W_r \) and \( W_v \) to get

\[
K_i = W^K X_i \\
V_i = W^V X_i
\]

- So the overall computation becomes

For each input embedding \( X_i \), compute:

\[
\alpha_{ij} = \text{score}(X_i, T_j) = \frac{Z_i \cdot K_j}{\sum_j \alpha_{ij} V_j} \quad \text{for all } j \leq i
\]

\[
Y_i = \sum_{j \leq i} \alpha_{ij} V_j
\]

↑

the output embedding corresponding to \( X_i \)

- the learned matrices \( W^o, W^K, W^V \) learn which values of the \( X_i \) are important in these roles and presumably emphasize them.

Further:

- When the inputs look like this, \( W^o \approx W^K \), then focus the output on that: \( W^V \)

- maybe.

Jurafsky: "Transformers allow us to create a more sophisticated way of representing how individual words can contribute to the representation of longer inputs."
- One set of weights $W^k, W^q, W^v$ can be thought of looking for one set of relationships
  - A single self-attention "head"

- Could be many such relationships, that will help NN succeed.

- So rather than just one set of $W^k, W^q, W^v$, we use several
  - Called "Multi-head Self-Attention"
  - Get $W^k, W^q, W^v$ for each head.

- In Assignment 3, use multiple "heads" * Transformer blocks $= 3$ = 1-layer
  * Heads $h = 3$ = 1-head
  * $d = 48$ = 1-embed

- Every head, though, increases the size of the output — don't want larger output than input

- So also train another weight matrix $W^o$ that is shape $hd \times d$ which "projects" the hidden output back down to $d$.

- The other parts of the transformer block are more common — see 5-3.
  1. Skip connections
  2. Normalization, Dropout, weight decay
Assignmet A3

- the code for a transformer is too complex to write
- instead we use Karpathy's minigo+

- still quite complex! Configurable to large transformers
- nicely written software, but will require careful reading, thinking & looking up by Jack Card.

- we have turned it into a language model
  + given you the code
  ⇒ asked you questions about that code to help you familiarize with it

- train it with:
  1. Simple Corpus.txt from A1
     - easier to see how language model works
     - look at predictions.
  2. Large Corpus.txt
    ⇒ to create pretrained model
    ⇒ fine tune a classifier to do sentiment analysis

- use HuggingFace ⇒ where the need models live, to also do sentiment analysis

- partially released tonight @ 9pm
- fully released later this week.