Week 2 Assignment

Generative AI

Saarland University – Winter Semester 2024/25

Martínez
7057573
cama00005@stud.uni-saarland.de

1 Exercise Assignment: E1

For this question, we consider a general setting where the input to the model is $N-1$ words denoted by $\mathbf{X}_{t-N+1:t-1} = [\mathbf{x}_{t-N+1}, \ldots, \mathbf{x}_{t-2}, \mathbf{x}_{t-1}]$, where $\mathbf{x}_i \in \mathbb{R}^{|V|} \quad \forall\, i \in [t-N+1,\, t-1]$, i.e., each word is a one-hot encoded vector, which by definition must be of size $|V|$, the vocabulary size. On the other hand, the embedding dimension is d, and the number of neural units in the hidden layer is d_h.

The first step in the model is to embed each vector in our input $\mathbf{X}_{t-N+1:t-1}$ into a d-dimensional space. This is done by multiplying each one-hot encoded input vector with an embedding matrix $E \in \mathbb{R}^{d \times |V|}$, such that $E\mathbf{x}_i = e_i \in \mathbb{R}^d$. *So, the number of parameters introduced by this first step is $|V|d$, the number of elements in the embedding matrix E.*

The second step is to concatenate the $N-1$ embedded vectors into a single vector $e \in \mathbb{R}^{(N-1)d}$. This vector is then passed through a hidden layer with d_h neural units. Each neural unit has a weight vector $\mathbf{w} \in \mathbb{R}^{(N-1)d}$ and a bias term $b \in \mathbb{R}$, such that $\sigma(\mathbf{w}e + b) = h \in \mathbb{R}$, where σ is the activation function (if using any[1]). This means that for each neural unit in the hidden layer, we introduce $(N-1)d + 1$ parameters, that is, the number of elements in the weight vector plus the bias term respectively. *Since there are d_h neural units, the total number of parameters introduced by this step is $((N-1)d + 1)\, d_h$.*

Finally, the output of the hidden layer is composed by terms $h_i \in \mathbb{R}$, with $i = 1, \ldots, d_h$, which are then concatenated into a single vector $\mathbf{h} \in \mathbb{R}^{d_h}$. This vector is passed through a linear layer with weights $U \in \mathbb{R}^{|V| \times d_h}$, such that $U\mathbf{h} = \mathbf{z} \in \mathbb{R}^{|V|}$. This output is then converted to probabilities with the *softmax* operation, such that the output of the model is the probability of the next word being each respective word in the vocabulary. *So, the number of parameters introduced by this last step is $|V|d_h$, the number of elements in the matrix U.*

Thus, summing all previous intermediate results, the final number of parameters is given by:

$$\begin{aligned} \text{\# parameters} &= \underbrace{|V|d}_{\text{Embedding matrix, } E} + \underbrace{((N-1)d+1)\, d_h}_{\text{Layer of neural units}} + \underbrace{|V|d_h}_{\text{Unembedding matrix, } U} \\ &= |V|\,(d + d_h) + ((N-1)d+1)\, d_h \end{aligned}$$

2 Exercise Assignment: E2

For this question, we consider a *Simple Feed-forward Neural Language Model with Single Attention Head*, where the input to the attention head is $N-1$ embedding vectors denoted by $\mathbf{e}_{t-N+1}, \ldots, \mathbf{e}_{t-2}, \mathbf{e}_{t-1}$.

[1] Note that the common activation functions in their base form such as *ReLU*, *sigmoid*, *softmax* or *tanh* do not introduce any parameters.

Preprint. Under review.

1. Compute similarity scores between each pair of $\mathbf{e}$ vectors. That is,

$$\text{score}(\mathbf{e}_{t-1}, \mathbf{e}_i) = \frac{W^q \mathbf{e}_{t-1} \cdot W^k \mathbf{e}_i}{\sqrt{d_k}}, \quad \forall\, i = t-N+1, \ldots, t-2, t-1$$

   where $W^q \in \mathbb{R}^{d_k \times d}$ and $W^k \in \mathbb{R}^{d_k \times d}$, and are the query and key matrices, respectively.

2. Compute attention weights α_i using the *softmax* activation function over the previously computed scores:

$$\alpha_i = \text{softmax}(\text{score}(\mathbf{e}_{t-1}, \mathbf{e}_i))$$

3. Finally, given the previously computed scalar values α_i, the attention vector $\mathbf{a}$ is given by:

$$\mathbf{a} = \sum_{i=t-N+1}^{t-1} \alpha_i \, W^v \mathbf{e}_i$$

   where $W^v \in \mathbb{R}^{d_v \times d}$ is the value matrix.

3 Exercise Assignment: E3

The number of parameters introduced by the single attention head in the *Simple Feed-forward Neural Language Model* introduced in E2 are the number of elements in the query, key and value matrices, i.e., W^q, W^k and W^v, which are respectively $d_k d$, $d_k d$ and $d_v d$. Thus, the total number of parameters introduced by the single attention head is:

$$\text{\# parameters} = d_k d + d_k d + d_v d = 2d_k d + d_v d$$

4 Exercise Assignment: E4

Following a similar reasoning to the one shown in E1, we again have $|V|d$ parameters introduced by the embedding matrix E.

After that, we have the attention mechanism, for which we found out on E3 that it introduces $2d_k d + d_v d$ parameters.

Finally, the output of the attention mechanism is passed through a layer of d_h neural units, where each neural unit has 1 weight vector $w_i \in \mathbb{R}^{d_v}$ and a bias term $b_i \in \mathbb{R}$, such that $\sigma(w_i \mathbf{a} + b_i) = h_i \in \mathbb{R}$, where σ is the activation function (if using any). This means that for each neural unit in the hidden layer, we introduce $d_v + 1$ parameters, that is, the number of elements in the weight vector plus the bias term respectively. Since there are d_h neural units, the total number of parameters introduced by this step is $(d_v + 1)\, d_h$.

Finally, the output of the layer of neural units is composed by terms $h_i \in \mathbb{R}$, with $i = 1, \ldots, d_h$, which are then concatenated into a single vector $\mathbf{h} \in \mathbb{R}^{d_h}$. This vector is passed through a linear layer with weights $U \in \mathbb{R}^{|V| \times d_h}$, such that $U\mathbf{h} = \mathbf{z} \in \mathbb{R}^{|V|}$. This output is then converted to probabilities with the `softmax` operation, such that the output of the model is the probability of the next word being each respective word in the vocabulary. So, the number of parameters introduced by this last step is $|V|d_h$, the number of elements in the matrix U.

Thus, the final number of parameters is given by the sum of all previously introduced parameters, as follows:

$$\begin{aligned}\text{\# parameters} = & \underbrace{|V|\, d}_{\text{Embedding matrix, } E} + \underbrace{2d_k d + d_v d}_{\text{Single Head Attention}} + \underbrace{(d_v + 1) d_h}_{\text{Layer of neural units}} + \underbrace{|V| d_h}_{\text{Unembedding matrix, } U} \\ & = |V|\,(d + d_h) + 2d_k d + d_v d + (d_v + 1) d_h\end{aligned}$$

5 Exercise Assignment: I1

As we saw in this week's slides, the lower the perplexity of a model, the better its quality is. Table 1 shows the perplexity of different configurations for the transformer model as we vary the number

of layers LAYERS and the context size N. We can see that the perplexity decreases as we increase the number of layers, and it also decreases as we increase the context size (but only for the case of LAYERS $= 4$). This is because the transformer model is able to capture more complex patterns in the data as we increase the number of layers, thereby introducing more parameters or, in other words, more tunable weights that can better adapt to the data. On the other hand, increasing the context size allows the model to consider more words in the input sequence, which can help to better predict the next word in the sequence. Nevertheless, if we increase the context size too much without caring for the number of LAYERS, and thus the number of trainable parameters, the model may not be able to take advantage of the larger context size to learn better patterns in the data, which is why the perplexity increases when LAYERS $= 2$ and going from N $= 2$ to N $= 5$. With LAYERS $= 4$, the model does show the desirable behavior, where the perplexity decreases as we increase the context size, because it has enough trainable parameters to take advantage of the larger context size.

Table 1: Perplexity and number of parameters for different configurations of the transformer model.

	LAYERS	
N	2	4
3	8.32 (# 9,721,596)	8.15 (# 14,980,860)
5	10.63 (# 9,721,596)	6.65 (# 14,980,860)

6 Exercise Assignment: I2

Table 1 also shows the number of parameters for different configurations of the transformer model as we vary the number of layers LAYERS and the context size N. As we saw in E4, the number of parameters for a transformer model does not depend on the context size N, thus the number of parameters is the same for all configurations with the same number of layers, regardless of the context size. In this case, $9{,}721{,}596$ and $14{,}980{,}860$ parameters for 2 and 4 layers, respectively.

On the other hand, the number of parameters must depend on the number of layers, since we are adding more stacked transform blocks, in which each block would have, for example, a single-head attention mechanism with their own W^q, W^k and W^v matrices (which we saw in in E3 that they introduce a certain number of parameters). This is why the number of parameters increases as we increase the number of layers, namely, from $9{,}721{,}596$ to $14{,}980{,}860$, a 65% increase with a change from 2 to 4 layers.

Acknowledgements

This week's slides.