Advanced Methods in Text Analytics
Exercise 5: Transformers - Part 1

Daniel Ruffinelli

FSS 2025

1 Transformer Basics

In this task, we review some basic concepts about the transformer architecture, and have a closer look at the original approach for positional embeddings.

(a) Let $x_1, x_2, \ldots, x_n$ be a sequence of input tokens and $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n\}$ where $\boldsymbol{h}_i \in \mathbb{R}^D$ the corresponding sequence of representations given by some neural network's hidden layer (e.g. an RNN or transformer encoder). Give a formal expression for how to compute context vector $\boldsymbol{c}_k$ using dot-product attention over $\boldsymbol{H}$ with given query $\boldsymbol{k} \in \mathbb{R}^K$ (e.g. the context vector corresponding to input token k in self-attention). Make sure you also provide a formal definition for every component in the expression you provide for $\boldsymbol{c}_k$. What are the names of these components? What is the size of $\boldsymbol{c}_k$? And how does D relate to K?

(b) What is the cost of a self-attention layer w.r.t. the input size? Why? Answer this for both the settings were we attend to all tokens in the input sequence and when we attend only to tokens previously seen in the input sequence. Why are the computations in a self-attention layer parallelizable?

(c) In this question, we aim at developing an intuition for how non-learned approaches to encoding positions work, by looking at the approach used in the original transformer architecture. Specifically, the authors used the following encoding to create the d-dimensional positional embedding (PE) for position k:

$$\text{PE}_{(k,2i)} = \sin\left(\frac{1}{n^{2i/d}}k\right),$$
$$\text{PE}_{(k,2i+1)} = \cos\left(\frac{1}{n^{2i/d}}k\right).$$

Here, $1 \leq i \leq d$ and n is a hyperparameter originally set to 10000. Note that the first equation is used to assign values to the *even* elements of the PE, and the second equation to the *uneven* elements. In other words, each element in positional embeddings is determined by either a sine or cosine function, each with different arguments.

(i) Given a maximum input length L, we can use these positional encodings to construct a PE matrix of size $L \times d$. Compute that matrix for the following sequences using $d = 4$ and $n = 1000$:

My cat is fine
My dog is well
My pets are very well

How do the resulting matrices compare to one another? What about the size of the vectors? Are they large in terms of magnitude? Is that what we want? Discuss this w.r.t. the information that we expect that positional encodings provide to our transformer model.

(ii) The multiplying factor to k in the argument of the sin/cos functions above is known as its frequency. Given a fixed frequency, i.e. using a fixed value of d, n, i, what happens to the value given by the sin/cos functions as we increase the values of k? What will the features of such vectors look like as a result?

(iii) The frequency discussed above is defined as the inverse to the period of the function as follows:

$$period = \frac{2\pi}{freq}$$

The period of a function refers to how long it takes to complete a full cycle (after which they start to repeat themselves). In other words, its horizontal stretch. Using the values for d and n given above, compute the periods for different values of i. How do these periods relate to each other? And what happens to the value of a given element of i for different values of k?

2 Thinking about Perplexity

In this task, we take a high-level look at the relation between perplexity, cross-entropy and log loss.

(a) For some random variable X, entropy $H(X)$ is defined as follows:

$$H(x) = -\sum_{x \in X} p(x) \log p(x) \tag{1}$$

Say X is the number shown when you toss a fair die. What is the entropy of X? Use log base 2 throughout.

(b) Following a), perplexity is defined as $2^{H(x)}$. Compute the perplexity of the result you obtained in a). Can you interpret the result? How does this expression relate to how we compute perplexity in code?

(c) In NLP, we are interested in the entropy of sequences. Say X is now a random variable over all possible sequences of length n over some language L. Write the expression to compute this entropy. How would you compute the average entropy per word in such a sequence?

(d) In the context of language models, p is the distribution over some natural language L, and we don't have access to this distribution. Instead, we learn an estimate m of this distribution from data. In such a context, the concept of *cross-entropy* $H(p, m)$ is suitable. It is defined as the sum of $p(x)$ for all $x \in X$ weighted by their corresponding log probabilities according to m. Write the expression for cross-entropy *rate* for a sequence of length n.

(e) It can be shown that the cross-entropy of a language can be approximated by the following expression:

$$H(W) = -\frac{1}{N} \log p(w_1, w_2, \ldots, w_N) \tag{2}$$

where N is a sufficiently large number and w_i are words in that language. As seen in question (b), perplexity is formally defined as $ppl(W) = 2^{H(W)}$, where W is $w_1, w_2, \ldots, w_N$, and we define entropy using log base 2. Use this approximation of cross-entropy given above to compute the expression for perplexity given in the lecture. What does this expression say about how we evaluate a language model by computing perplexity on a held-out corpus?