AI539 - Natural Language Processing with Deep Learning

Homework 4 Report
Attention Mechanisms in Sequence-to-Sequence Models

Author: Vy Bui
OSUID: 934370552

Instructor: Professor Stefan Lee

The School of Electrical Engineering and Computer Science
Oregon State University

Task 1.1
Copying [2pts]

Describe (in one or two sentences) what properties of the keys and queries would result in the output $\mathbf{a}$ being equal to one of the input values $\mathbf{v}_j$. Specifically, what must be true about the query $\mathbf{q}$ and the keys $\mathbf{k}_1, ..., \mathbf{k}_m$ such that $\mathbf{a} \approx \mathbf{v}_j$? (We assume all values are unique – $\mathbf{v}_i \neq \mathbf{v}_j,\ \forall ij$.)

Task 1.2
Average of Two [2pts]

Consider a set of key vectors $\{\mathbf{k}_1, ..., \mathbf{k}_m\}$ where all keys are orthogonal unit vectors – that is to say $\mathbf{k}_i \mathbf{k}_j^T = 0,\ \forall ij$ and $\|\mathbf{k}_i\| = 1,\ \forall i$. Let $\mathbf{v}_a, \mathbf{v}_b \in \{\mathbf{v}_1, ..., \mathbf{v}_m\}$ be two value vectors. Give an expression for a query vector $\mathbf{q}$ such that the output $\mathbf{a}$ is approximately equal to the average of $\mathbf{v}_a$ and $\mathbf{v}_b$, that is to say $\mathbf{a} \approx \frac{1}{2}(\mathbf{v}_a + \mathbf{v}_b)$. You can reference the key vectors corresponding to $\mathbf{v}_a$ and $\mathbf{v}_b$ as $\mathbf{k}_a$ and $\mathbf{k_b}$ respectively. Note that due to the softmax in Eq. 1, it won't ever actually reach this value, but you can make it arbitrarily close by adding a scaling constant to your solution.

Task 1.3
Noisy Average [2pts]

Now consider a set of key vectors $\{\mathbf{k}_1, ..., \mathbf{k}_m\}$ where keys are randomly scaled such that $\mathbf{k}_i = \mu_i * \lambda_i$ where $\lambda_i \sim \mathcal{N}(1, \beta)$ is a randomly sampled scalar multiplier. Assume the unscaled vectors $\mu_1, ..., \mu_m$ are orthogonal unit vectors. If you use the same strategy to construct the query q as you did in Task 1.2, what would be the outcome here? Specifically, derive $\mathbf{q}\mathbf{k}_a^T$ and $\mathbf{q}\mathbf{k}_b^T$ in terms of μ's and λ's. Qualitatively describe what how the output a would vary over multiple resamplings of $\lambda_1, ..., \lambda_m$.

Task 1.4
Noisy Average with Multi-head Attention [2pts]

Let's now consider a simple version of multi-head attention that averages the attended features resulting from two different queries. Here, two queries are defined ($\mathbf{q}_1$ and $\mathbf{q}_2$) leading to two different attended features ($\mathbf{a}_1$ and $\mathbf{a}_2$). The output of this computation will be $\mathbf{a} = \frac{1}{2}(\mathbf{a}_1 + \mathbf{a}_2)$. Assume we have keys like those in Task 1.3, design queries $\mathbf{q}_1$ and $\mathbf{q}_2$ such that $\mathbf{a} \approx \frac{1}{2}(\mathbf{v}_a + \mathbf{v}_b)$.

Task 2.1
Scaled-Dot Product Attention [8pts]

Implement Attn(·) in equation (11) as single-query scaled dot-product attention as defined in equations (1) and (2). Here, the query will be the decoder hidden state and the keys and values will be derived from the encoder representations. Implement this attention mechanism by completing the `SingleQueryScaledDotProductAttention` class in `mt_driver.py`.

The forward function takes two inputs – `hidden` is the decoder hidden state $h_j^{(d)}$ and `encoder_outputs` corresponds to encoder word representations $h_t^{(e)}$, $\forall t$. These should be converted to keys, queries, and values:

$$\mathbf{q} = W_q \mathbf{h}_j^{(d)} \tag{1}$$

$$\mathbf{k}_t = W_k \mathbf{h}_t^{(e)} \tag{2}$$

$$\mathbf{v}_t = \mathbf{h}_t^{(e)} \tag{3}$$

And the output – `attended_val` and `alpha` – correspond to the attended value vector ($\mathbf{a}$) and the vector of attention values (α) computed from as in equations (1) and (2). The expected dimensions are asserted above. Note that this is intended to be a batched operation and the equations presented are for a single instance. `torch.bmm` can be very useful here.

Train this model by executing `python mt_driver.py`. Record the perplexity and BLEU score on the test set. These are automatically generated in the script and printed after training.

Task 2.2
Attention Diagrams [1pts]

Search through the attention diagrams produced by your model. Include a few examples in your report and characterize common patterns you observe.Note that German is (mostly) a Subject-Object-Verb language so you may find attention patterns that indicate inversion of word order when translating to Subject-Verb-Object English as in the 2nd example above.

Task 2.3
Comparison [3pts]

Train and evaluate models with the `Dummy` and `MeanPool` 'attention' mechanisms. Report mean and variance over three runs for these baselines and your implementation of scaled dot-product attention. Discuss the observed trends.

Task 2

EC Beam Search and BLEU [2pts]

In the previous homework, we implemented many decoding algorithms; however, in this work we just use greedy top-1 in the `translate_sentence` function. Adapt your implementation of beam search from HW3 to work on this model by augmenting `translate_sentence` (which is used when computing BLEU). Report BLEU scores on the test set for the scaled dot-product attention model with B=5, 10, 20, and 50.