Carolyn Kao (chkao831@stanford.edu)
CS224n Assignment 5
February 20, 2021

1. Attention exploration

(a) (2 points) **Copying in attention:** Recall that attention can be viewed as an operation on a query $q \in \mathbb{R}^d$, a set of value vectors $\{v_1, \ldots, v_n\}, v_i \in \mathbb{R}^d$, and a set of key vectors $\{k_1, \ldots, k_n\}, k_i \in \mathbb{R}^d$, specified as follows:

$$c = \sum_{i=1}^{n} v_i \alpha_i \tag{1}$$

$$\alpha_i = \frac{\exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)}. \tag{2}$$

where α_i are frequently called the "attention weights", and the output $c \in \mathbb{R}^d$ is a correspondingly weighted average over the value vectors.

We'll first show that it's particularly simple for attention to "copy" a value vector to the output c. Describe (in one sentence) what properties of the inputs to the attention operation would result in the output c being approximately equal to v_j for some $j \in \{1, \ldots, n\}$. Specifically, what must be true about the query q, the values $\{v_1, \ldots, v_n\}$ and/or the keys $\{k_1, \ldots, k_n\}$?

Precisely, in order to have the output c be approx. equal to v_j for some $j \in \{1, \ldots, n\}$, we need α_j (the attention weights on j) to be approximately equal to 1. Reason could be seen from the definition of α_i: $\frac{\exp(k_j^T q)}{\sum_{i=1}^{n} \exp(k_i^T q)}$ in which the numerator would dominate the whole term when the dot product of k_j and q >> dot products of others.

(b) (4 points) **An average of two:** Consider a set of key vectors $\{k_1, \ldots, k_n\}$ where all key vectors are perpendicular, that is $k_i \perp k_j$ for all $i \neq j$. Let $\|k_i\| = 1$ for all i. Let $\{v_1, \ldots, v_n\}$ be a set of arbitrary value vectors. Let $v_a, v_b \in \{v_1, \ldots, v_n\}$ be two of the value vectors. Give an expression for a query vector q such that the output c is approximately equal to the average of v_a and v_b, that is, $\frac{1}{2}(v_a + v_b)$.[1] Note that you can reference the corresponding key vector of v_a and v_b as k_a and k_b.

$q = \beta(k_a + k_b)$ where β is a very large scale factor

With such expression of q, the softmax (formula for α_a) now becomes

$$\alpha_a = \frac{\exp(\beta)}{\underbrace{\sum_{n-2} \exp(k_j^T q)}_{\text{for } i \neq a,b} + 2\exp\left[k_a^T(\beta(k_a + k_b))\right]}$$

$$= \frac{\exp(\beta)}{\sum_{n-2} \exp(0) + 2\exp\left(\beta \|k_a\|^2 + 1\right)}$$

$$= \frac{\exp(\beta)}{(n-2) + 2\exp(\beta)}$$

where the exponential terms dominate as β is very large

$$= \frac{1}{2}$$

This yields the required result of $c \approx \frac{v_a + v_b}{2}$ by the definition given.

(c) (5 points) **Drawbacks of single-headed attention:** In the previous part, we saw how it was *possible* for a single-headed attention to focus equally on two values. The same concept could easily be extended to any subset of values. In this question we'll see why it's not a *practical* solution. Consider a set of key vectors $\{k_1, \ldots, k_n\}$ that are now randomly sampled, $k_i \sim \mathcal{N}(\mu_i, \Sigma_i)$, where the means μ_i are known to you, but the covariances Σ_i are unknown. Further, assume that the means μ_i are all perpendicular; $\mu_i^\top \mu_j = 0$ if $i \neq j$, and unit norm, $\|\mu_i\| = 1$.

i. (2 points) Assume that the covariance matrices are $\Sigma_i = \alpha I$, for vanishingly small α. Design a query q in terms of the μ_i such that as before, $c \approx \frac{1}{2}(v_a + v_b)$, and provide a brief argument as to why it works.

Since $\Sigma_i = \alpha I$, α small, we could treat $k_i \sim N(\mu_i, \Sigma_i)$ with negligible variance term.

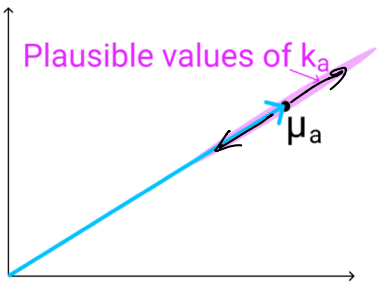
As μ_a and μ_b are given (known to us), we could simply denote $k_a \approx \mu_a$ and $k_b \approx \mu_b$ s.t. $c \approx \frac{1}{2}(v_a + v_b)$ as pointed out in 1(b).

Hence, the design of query q is similar to the solution in 1(b): $\boxed{q = \beta(\mu_a + \mu_b)}$

ii. (3 points) Though single-headed attention is resistant to small perturbations in the keys, some types of larger perturbations may pose a bigger issue. Specifically, in some cases, one key vector k_a may be larger or smaller in norm than the others, while still pointing in the same direction as π_a. As an example, let us consider a covariance for item a as $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$ for vanishingly small α (as shown in figure 1). Further, let $\Sigma_i = \alpha I$ for all $i \neq a$.
When you sample $\{k_1, \ldots, k_n\}$ multiple times, and use the q vector that you defined in part i., what qualitatively do you expect the vector c will look like for different samples?

Previously, with $\Sigma_i = \alpha I \ \forall i \neq a$, $c = \frac{1}{2} v_a + \frac{1}{2} v_b$.
However, now with the additional outer product term $\frac{1}{2} \mu_a \mu_a^T$, the weights imposed on μ_a and μ_b are no longer $\frac{1}{2}, \frac{1}{2}$ respectively.

By definition, $k_a = \exp(k_a^T \mu_a + k_a^T \mu_b)\beta$, and $\|k_a\|^2$ is no longer $= 1$.



This results in K_a being no longer close to μ_a. Its magnitude would differ along the direction as shown on the left. The former Gaussian normal dist. would also become more fat-tailed.

Hence, as there's no guarantee that $\alpha_a = \alpha_b = \frac{1}{2}$, it is obvious that the vector $c = \Sigma v_i \alpha_i$ would fluctuate a lot for different samples.

(d) (3 points) **Benefits of multi-headed attention:** Now we'll see some of the power of multi-headed attention. We'll consider a simple version of multi-headed attention which is identical to single-headed self-attention as we've presented it in this homework, except two query vectors (q_1 and q_2) are defined, which leads to a pair of vectors (c_1 and c_2), each the output of single-headed attention given its respective query vector. The final output of the multi-headed attention is their average, $\frac{1}{2}(c_1+c_2)$. As in question 1(c), consider a set of key vectors $\{k_1, \ldots, k_n\}$ that are randomly sampled, $k_i \sim \mathcal{N}(\mu_i, \Sigma_i)$, where the means μ_i are known to you, but the covariances Σ_i are unknown. Also as before, assume that the means μ_i are mutually orthogonal; $\mu_i^\top \mu_j = 0$ if $i \neq j$, and unit norm, $\|\mu_i\| = 1$.

i. (1 point) Assume that the covariance matrices are $\Sigma_i = \alpha I$, for vanishingly small α. Design q_1 and q_2 such that c is approximately equal to $\frac{1}{2}(v_a + v_b)$.

From (c)(i), when we set $q = \beta(\mu_a + \mu_b)$,
we'd have $c = \frac{1}{2}(v_a + v_b)$

Now, by the def of multi-headed attention $c = \frac{1}{2}(c_1+c_2)$ which takes the average of pairs of vectors c_1, c_2, we could intuitively set c_1 to be v_a and c_2 to be v_b and eventually have $\boxed{q_1 = \beta\mu_a,\ q_2 = \beta\mu_b}$, yielding $q_1 + q_2 = \beta(\mu_a + \mu_b)$ as in (c)(i).

ii. (2 points) Assume that the covariance matrices are $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$ for vanishingly small α, and $\Sigma_i = \alpha I$ for all $i \neq a$. Take the query vectors q_1 and q_2 that you designed in part i. What, qualitatively, do you expect the output c to look like across different samples of the key vectors? Please briefly explain why. You can ignore cases in which $q_i^\top k_a < 0$.

From (d)(i), with the formation, $c_1 = v_a$ and $c_2 = v_b$. This indicates that the attention on a and b would not charge regardless. Recall that

$$\alpha_i \overset{\text{DEF}}{=} \frac{\exp(k_i^T q)}{\sum_j \exp(k_j^T q)} = \frac{\exp(\beta \|k_a\|^2)}{(n-1) + \exp(\beta \|k_a\|^2)}$$

Here, although $\|k_a\|^2 \neq 1$, it doesn't matter. As long as the β term dominates, $\alpha_i = 1$

Compared to part (c) where we don't have the multiheaded attention, now we have no randomness over the attention on a and b.

$c = \frac{1}{2}(c_1 + c_2) = \frac{1}{2}(Va + Vb)$, yielding more control of the result compared to previous parts.

(e) (7 points) **Key-Query-Value self-attention in neural networks:** So far, we've discussed attention as a function on a set of key vectors, a set of value vectors, and a query vector. In Transformers, we perform *self-attention*, which roughly means that we draw the keys, values, and queries from the same data. More precisely, let $\{x_1, \ldots, x_n\}$ be a sequence of vectors in $\mathbb{R}^d$. Think of each x_i as representing word i in a sentence. One form of self-attention defines keys, queries, and values as follows. Let $V, K, Q \in \mathbb{R}^{d \times d}$ be parameter matrices. Then

$$v_i = Vx_i \quad i \in \{1, \ldots, n\} \tag{3}$$
$$k_i = Kx_i \quad i \in \{1, \ldots, n\} \tag{4}$$
$$q_i = Qx_i \quad i \in \{1, \ldots, n\} \tag{5}$$

Then we get a context vector for each input i; we have $c_i = \sum_{j=1}^{n} \alpha_{ij} v_j$, where α_{ij} is defined as $\alpha_{ij} = \frac{\exp(k_j^\top q_i)}{\sum_{\ell=1}^{n} \exp(k_\ell^\top q_i)}$. Note that this is single-headed self-attention.

In this question, we'll show how key-value-query attention like this allows the network to use different aspects of the input vectors x_i in how it defines keys, queries, and values. Intuitively, this allows networks to choose different aspects of x_i to be the "content" (value vector) versus what it uses to determine "where to look" for content (keys and queries.)

i. (3 points) First, consider if we didn't have key-query-value attention. For keys, queries, and values we'll just use x_i; that is, $v_i = q_i = k_i = x_i$. We'll consider a specific set of x_i. In particular, let u_a, u_b, u_c, u_d be mutually orthogonal vectors in $\mathbb{R}^d$, each with equal norm $\|u_a\| = \|u_b\| = \|u_c\| = \|u_d\| = \beta$, where β is very large. Now, let our x_i be:

$$x_1 = u_d + u_b \tag{6}$$
$$x_2 = u_a \tag{7}$$
$$x_3 = u_c + u_b \tag{8}$$

If we perform self-attention with these vectors, what vector does c_2 approximate? Would it be possible for c_2 to approximate u_b by adding either u_d or u_c to x_2? Explain why or why not (either math or English is fine).

By $c_i = \sum_{j=1}^{n} \alpha_{ij} v_j$, $c_2 = \sum \alpha_{2j} v_j = \sum \alpha_{2j} x_j$

$\because u_i \perp u_j \quad \forall i, j \in d$

$\therefore k_1^\top q_2 = (u_d + u_b)^\top u_a = 0$ by (6)

$k_3^\top q_2 = (u_c + u_b)^\top u_a = 0$ by (8)

$k_2^\top q_2 = u_a^\top u_a$ by (7)

$= \|u_a\|^2$

because $\|u_a\| = \ldots \|u_d\| = \beta$

$= \beta^2$ —— (*)

Then, recall that $\alpha_{ij} = \frac{\exp(k_j^\top q_i)}{\sum_{\ell=1}^{n} \exp(k_\ell^\top q_i)}$

By ⊛, $\alpha_{22} \approx \frac{\exp(\beta^2)}{\exp(\beta^2) + \sum_{i=1}^{2} \exp(0)} = \frac{\exp(\beta)^2}{\exp(\beta^2)+2} = 1$

by β domination

$\alpha_{21} \approx 0$

$\alpha_{23} \approx 0$

Finally, $\boxed{\begin{aligned} c_2 &= \alpha_{22} x_2 \\ &= 1 \cdot u_a \end{aligned}}$ by (7) #

It would NOT be possible for c_2 to approximate u_b by adding u_d or u_c to x_2.

For example, say, $x_2 = u_a + u_d$

Then $k_1^T q_2 = (u_d + u_b)^T (u_a + u_d)$

$= \|u_d\|^2 = \beta^2 \neq 0$

$k_2^T q_2 = 2\beta^2$

$k_3^T q_2 = 0$

Hence c_2 is a weighted combination of v_1 and v_2, which contradicts the original setup.

ii. (4 points) Now consider using key-query-value attention as we've defined it originally. Using the same definitions of x_1, x_2 and x_3 as in part i, specify matrices K, Q, V such that $c_2 \approx u_b$, and $c_1 \approx u_b - u_c$. There are many solutions to this problem, so it will be easier for you (and the graders), if you first find V such that $v_1 = u_b$ and $v_3 = u_b - u_c$, then work on Q and K. Some outer product properties may be helpful (as summarized in this footnote)[2].

[2]For orthogonal vectors $u, v, w \in \mathbb{R}^d$, the outer product $uv^\top$ is a matrix in $\mathbb{R}^{d\times d}$, and $(uv^\top)v = u(v^\top v) = u\|v\|_2^2$, and $(uv^\top)w = u(v^\top w) = u * 0$. (The last equality is because v and w are orthogonal.)

By footnote

$(uv^T)v = u(v^Tv) = u\|v_2\|^2$

$(uv^T)w = u(v^Tw) = u \cdot 0 = 0$

$v_i = V x_i$

$k_i = K x_i$

$q_i = Q x_i \quad i \in \{1, \ldots, n\}$

by hint, let $v_1 = u_b$, $v_3 = u_b - u_c$.

$v_1 = V x_1$ — (*)

$v_3 = V x_3$ — (* *)

(*): want $V(u_d + u_b) = u_b$

Let $V = \beta^{-2} u_i u_b^T$,

then $V(u_d + u_b)$

$$= [\beta^{-2}(u_i u_b^T)](u_d + u_b)$$

$$= u_i \|u_b\|^2$$

$$= \beta^{-2} u_i \beta^2$$

$$= u_i$$

From here, we see that

v could be u_b or u_d

and u_i needs to be u_b

Hence, $\boxed{V = \beta^{-2} u_b u_b^T}$

(X.X): Want $VX_3 = V_3$

$\Rightarrow$ $c_1 \approx v_3$, $c_2 \approx v_1$

$\Rightarrow$ $\alpha_{13} \approx 1$, $\alpha_{11} = \alpha_{12} \approx 0$

$\Rightarrow$ $K_3^T q_1 \geq K_1^T q_1$, $K_2^T q_1$

$K_1^T q_2 \geq K_2^T q_2$, $K_3^T q_2$

From here, there are many options, but we could, for example, set

$K_1 = q_2$, $K_2 = 0$, $K_3 = q_1$

$\Rightarrow$ $\boxed{K = u_d u_d^T + u_c u_c^T}$

$\boxed{Q = u_d u_a^T + u_c u_d^T}$

2. Pretrained Transformer models and knowledge access

(d) **Make predictions (without pretraining)**
Report model's accuracy on the dev set as printed by `python src/run.py evaluate vanilla wiki.txt --reading_params_path vanilla.model.params --eval_corpus_path birth_dev.tsv --outputs_path vanilla.nopretrain.dev.predictions`

```
chkao831@CS224nHW4:~/student-new$ python src/run.py evaluate vanilla wiki.txt --reading_params_pa
th vanilla.model.params --eval_corpus_path birth_dev.tsv --outputs_path vanilla.nopretrain.dev.pr
edictions
data has 418352 characters, 256 unique.
number of parameters: 3323392
500it [01:21,  6.12it/s]
Correct: 12.0 out of 500.0: 2.4%
```

```
The accuracy on the dev set is 2.4%.
```

As a reference point, we want to also calculate the accuracy the model would have achieved if it had just predicted "London" as the birth place for everyone in the dev set.

```
chkao831@CS224nHW4:~/student-new$ python3 src/london_baseline.py
500it [00:00, 1204567.49it/s]
Correct: 25.0 out of 500.0: 5.0%
```

```
The accuracy on dev set if the model just had predicted 'London' as the birth place for all is 5.0%.
```

(e) **Define a span corruption function for pretraining**
A sample output for running `python src/dataset.py charcorruption` would be

```
chkao831@CS224nHW4:~/student-new$ python src/dataset.py charcorruption
data has 418352 characters, 256 unique.
x: Khatchig Mouradian. Khatchig Mouradian is a journalist, writer and tra⁇⁇nslator born in Lebanon□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: hatchig Mouradian. Khatchig Mouradian is a journalist, writer and tra⁇⁇nslator born in Lebanon□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
x: Jacob Henry Studer. Jacob H⁇840 Columbus, Ohio - 2 Augu⁇enry Studer (26 February 1□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: acob Henry Studer. Jacob H⁇840 Columbus, Ohio - 2 Augu⁇enry Studer (26 February 1□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
x: John Stephen⁇in G⁇. Born □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: ohn Stephen⁇in G⁇. Born □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
x: Georgina Willis. Georgina Willis is an award wi⁇lm direc⁇nning fi□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: eorgina Willis. Georgina Willis is an award wi⁇lm direc⁇nning fi□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

(f) **Pretrain, finetune, and make predictions**
Report model's accuracy on the dev set as printed by `python src/run.py evaluate vanilla wiki.txt --reading_params_path vanilla.finetune.params --eval_corpus_path birth_dev.tsv --outputs_path vanilla.pretrain.dev.predictions`

```
chkao831@CS224nHW4:~/student-new$ python src/run.py evaluate vanilla wiki.txt --reading_params_path vanilla.finetune.para
ms --eval_corpus_path birth_dev.tsv --outputs_path vanilla.pretrain.dev.predictionsdata has 418352 characters, 256 unique
.
number of parameters: 3323392
500it [01:22,  6.26it/s]
Correct: 146.0 out of 500.0: 29.2%
```

```
Now with pretraining, the accuracy on the dev set becomes 29.2%.
```

(g) **Write and try out the synthesizer variant**
Report the accuracy of your synthesizer attention model on birth-place prediction on `birth_dev.tsv` after pretraining and fine-tuning as printed by `python src/run.py evaluate synthesizer wiki.txt`

```
--reading_params_path synthesizer.finetune.params --eval_corpus_path birth_dev.tsv
--outputs_path synthesizer.pretrain.dev.predictions
chkao831@CS224nHW4:~/student-new$ python src/run.py evaluate synthesizer wiki.txt --reading_params_path synthesizer.finetune.param
s --eval_corpus_path birth_dev.tsv --outputs_path synthesizer.pretrain.dev.predictions
data has 418352 characters, 256 unique.
number of parameters: 3076988
500it [01:17,  6.59it/s]
Correct: 52.0 out of 500.0: 10.4%
```

```
Using synthesizer, the accuracy on the dev set becomes 10.4%.
```

Why might the synthesizer self-attention not be able to do, in a single layer, what the key-query-value self-attention can do?

```
Without the key-query-value structure in self-attention, the synthesizer might not be able to learn
attention weights from token-token (query-key) interactions. Hence, its capability in adapting
itself to different tasks would probably restrict to certain types of source inputs.
```

3. Considerations in pretrained knowledge

(a) **Succinctly explain why the pretrained (vanilla) model was able to achieve an accuracy of above 10%, whereas the non-pretrained model was not.**

```
With pretraining, it is possible to mask out the name of a person given the birth place or mask out
the birth place given the person name. This is because pretraining associates the names with the
birthplaces. Then, at finetuning time, such information can then be accessed as it is encoded in the
parameters at initialization. However, without pretraining, the model wouldn't have any previous
knowledge of the birth places of people that weren't in the finetuning training set, so the accuracy
is hindered.
```

(b) **Take a look at some of the correct predictions of the pretrain+finetuned vanilla model, as well as some of the errors. We think you'll find that it's impossible to tell, just looking at the output, whether the model retrieved the correct birth place, or made up an incorrect birth place. Consider the implications of this for user-facing systems that involve pretrained NLP components. Come up with two reasons why this indeterminacy of model behavior may cause concern for such applications.**

```
1. Lack of reliability and credibility:
Of course we could check the model accuracy while performing experiments; however, from a user's
perspective, if it's hard to tell the correctness of the predicted output (birth place) which might
be wrong or even manipulated, the reliability and credibility of the NLP model is hindered, possibly
affecting one's decision making in such tasks.

2. Introduction of unwanted biases:
```

Although it might not be the case in this assignment, with certain pre-training involved, it is shown that the standard corpora for pre-training contextual word models might exhibit significant bias imbalances. With such underlying social and intersectional biases that might not be introduced in our main task from a narrower scope of finetuning, this increases social and intersectional concern from a user's perspective while solely looking at the unreliable and indeterminate output.

(c) **If your model didn't see a person's name at pretraining time, and that person was not seen at fine-tuning time either, it is not possible for it to have "learned" where they lived. Yet, your model will produce something as a predicted birth place for that person's name if asked. Concisely describe a strategy your model might take for predicting a birth place for that person's name, and one reason why this should cause concern for the use of such applications.**

Without further specification, if my model didn't see a person's name at pretraining or fine-tuning time, it would not 'learn' such information and might simply return some token that corresponds to some similar terms from other input sequences by default. It would potentially cause concern also due to the lack of credibility as well as the increase of difficulty to tell if some prediction is generated by 'learning' or by such way.