NLP Homework 5

Due 7/20

Watch Video:

<https://www.youtube.com/watch?v=rBCqOTEfxvg>

slides: <https://drive.google.com/file/d/0B8BcJC1Y8XqobGNBYVpteDdFOWc/view>

|  |  |
| --- | --- |
| Previously, we did language models, translation, etc. token by token. But when we speak, we have a larger context (long range dependencies). This is very important to make these networks understand more. So, we want translation model to maybe look back many steps (maybe hundreds…). | Since we want to train long sentences, we can train over Wikipedia articles. It just learns to predict the next word (a language model). This is not translation. |
|  | RNN: Problem is that you cannot use very long sequences since the gradient will start to vanish. |
| People started to solve this with convolutions | If the orange part was just an RNN, to get to the 1000th word, you have to go through 999 words before that. |
|  | **Using CNN’s to solve this:** **Wavenet** (left of slide):  We make a convolution on the lower layer (blue) to jump 2 steps. Then the next (hidden) layer skips one at every point. The next (hidden layer) skips 3 at every point (so you see 8 of the original words). It reduces the complexity logarathmically compared to RNN. Its better than RNN, but still very positional. Example, a hidden neuron always sees word 13 before it. Why 13, what is its significance – nothing. This is referencing by position, whereas referencig by content would be more natural. |
| How do you reference by content? – by using attention mechanisms. **The idea is to make a query with a vector and look at similar things in the past.** | Convolutions are position dependent (hence every position has a different color), but attention looks at everything, but gets weighs the important ones more (the ones connected by the dark grey are similar and the ones connected by light grey are not so similar). |
|  | When you retreive similar things, you can look at a very long context. For example, if the NN referred to a person long back in the sentence, it can look back that far and say, “earlier, I referred to personA, so I should probably refer to personA again now”. You could do this with CNNs, but it was very positional. With Attention, since its context based queries, you can do this quickly and look at everything. |
|  | **3 attention mechanisms:** Previously, we saw only an encoding model. But for translation, we need a decoder as well. So we need an “attention” in the output that can attend to the input (encoder-decoder attention). You need an “attention” in the input which can attend everywhere in the input sequence. You also need “attention” in the output that only attends to things it has decoded before. |
|  | **Input =** English Sentence.  **Decoder:** You could have a RNN there, but since you are decoding the next word from the previous word, for the 2000th word, you will have to process the 1st 1999 words (wait for it). In this model however, we use matrix multiplication with all the words (so its faster), and we just mask the future words. TODO CHECK: But we still have to do it for 2000 words?  The multihead attention that attends each of the 2000 words to each of the 2000 words is a matrix multiply and the GPU hardware is extrememly optimized for this.  This is simpler than a RNN – there is no recurrence, no weight sharing. Everything happenes in a bunch of matrix multiplies. |
|  | **Attention:** There is a **query** which is a vector (think of it as the **current word that I am operating on**). Then you have the **key and value** which is the **memory** (you can think of this as the **past**). Side note: K and V can be the same thing (but they don’t have to be).  **How it works:** Take the the query, find the most similar key (dot product), then get the values corresponding to these similar keys.  To make it efficienct, you multiply Q with transposed K, then take a softmax. This gives you a probability distribution over the keys **(which is peaked at the ones that are most similar to the queries)**. Then you weight each value vector by their respective probabilities. |
| Normalize a little to train well. | RNN: every word needs to make a step so there are ‘n’ steps and in every step, you have a dimension \* dimension operation [(word vector length + num\_rnn) \* weight matrix].  Attention: Everyone is attending to everyone so we have n\*n and in each of these we have a dot product of size d (roughly).  ‘d’ is usually 1000 in the large networks, while n is sequence length (say 70). Hence, for these sizes, attention is more efficient. |
|  | Since attention is a purely similarity based measure, it just retreives the most similar words. It does not know what came before what. So it loses positional information. But the order of words is really important. |
|  |  |
|  |  |
|  |  |
|  | **Winograd Schemas**: You really need to understand the meaning of the word. Example, “He did not put the trophy into the suitcase because it was too big” 🡪 What was big – the trophy or the suitcase? This is hard for machine to do this.  This model does relatvely well compared to others. There were 40 attention heads, the blue one (“it”) is just one of them. In the first case (left), it gives more weightage to animal than to street (as expected) and in the second case (right), it gave more attention to street than to animal (as expected).  It is also interesting that **each head** is doing something useful and **repeatable** (like a filter in CNN). There was one head that was always doing noun resolution like this one. There was another head that was always loking for the head noun, etc. |
|  |  |
| Currently, we need one model for parsing, one model for entity resolution, one model for grammar correction, etc. Can we do other tasks with this attention mechanism? Maybe it should also do images and speech. But how do you have one model that does images (256x256) and text processing (token vectors)?  So they created “modalities” that take the raw input and process it a bit so that it can be processed by the input encoder. | They trained this on 8 different tasks – 4 translation tasks, Imagenet, Image Captioning, Parsing (POS) and Speech Recognition. This did not do so well.  But a “mixture of experts” (MOE) model allows you to scale the size of your network without slowing it down. In a reqular scheme, if your hidden dimension is d, execution slows down by d\*d. But in a MOE model, if we have 1000 matrices, then there is a gate which will only pick 4 of these to operate on (for a word). Then it learn to pick 4 different ones for another word and so on. This helps to reduce complexity. This allows you a lot of capacity to memorize (lot of matrices) wihout paying for execution time. |
| Results with MOE is much better.  You trained a model to translate from English to French and English to German (but not on French to German). **But this can do that** (even though it did not see any data like that in training). **So it helps with tasks where you have little data** (like French to German).  **NOTE: All these tasks are being trained at the same time. Sometime, it sees langugae data, sometimes it sees image data.** | Added all this content to a library 🡪 **tensor2tensor**. It has all the tricks: optimizer, label smoothing, learning rate decay, etc. |
|  |  |
| **Positional Encoding:** For every position ‘i’, you will make a position vector which will have sin(i), sin(i/2), sin(i/4), … sin(i/2^d). Also adds cosine.  This position vector will allows at least one of the attention heads to attend to the positions just before (a few words before) as this is important to know. |  |

Q1: Why is the attention faster than a recurrent when the attention scales as n2d while Recurrent is nd2? (n is sequence length, d is the dimension)

We have seq len \* seq len comparisons hence n2 and we multiply that by d matrix multiplication). But n is much smaller than d

Q2: What is label smoothing

<https://towardsdatascience.com/label-smoothing-making-model-robust-to-incorrect-labels-2fae037ffbd0>

Q3: How was the Transformer extended to other tasks beside translation

Multimodal models – have modules that convert any type of input into the right format for the transformer.

Q4: What 8 tasks was the multimodal trained on.

Image, speech, etc.

Q5: What package does the author talk about that contains the Transformer Code

Tensor2tensor