Notes: "Non-autoregressive neural machine translation"

Sun

September 28, 2020

1 Summary:

1. Translation, but non-autoregressive.
2. By sidestepping autoregressiveness, inference-time speedup is up to 15x above the baseline – at the cost of a lot of BLEU.
3. Works by using a set of transformer-encoders designed to internalize both length prediction and sentence generation to avoid variable-length generation.

1.1 High level:

The most obvious issues with generation non-autoregressive translation (NAT) are:

1. Sequence length is variable;
2. All tokens after timestep t depend explicitly on all previous;
3. There are multiple completely valid translations for a given sentence.

Traditional neural machine translation avoids this issue by generating a single token at a time: generating t_{n+1} given $\{t_0, t_1, ..., t_n\}$. This turns a variable length generation into a chain of fixed length (1) generations. Inbetween each iteration, it also traditionally gets feedback from a "teacher", which continually points it to the correct translation.

How can we do this without being autoregressive? A naive approach is simply to predict sentence length, then, for each word in the (predicted) output sentence, predict what word belongs there independent of the others. As noted in the paper, this is similar to asking a group of translators what the *nth* word of a sentence translation is independently. But this doesn't work very well, and only addresses (1) in the above list.

So instead, let's modify it. During train time, we calculate how many words each word in the source sentence corresponds to in the target sentence. We pass the source sentence into a transformer encoder, then, for each output embedding, predict the number of words the embedding corresponds to in the target sentence.

From there, we then have a second separate transformer encoder (which is used as a decoder, henceforth referred to as such). We input into the decoder both the predicted number of outputs words per input word, and let the decoder attend to encoder representations and itself.

This does address all 3 points of the above list, with the caveat that if there are multiple same-length translations for a single sentence, RIP.

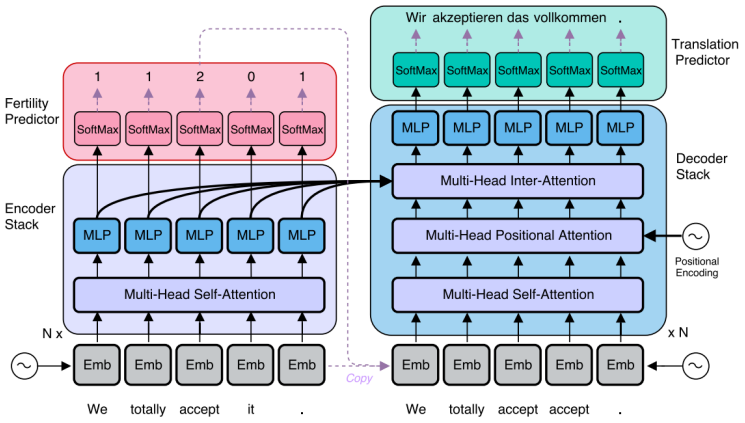

Figure 1: The proposed architecture – notice the embedding word 'smearing' from encoder→decoder.

1.2 Medium level:

The above section does provide a good overview, but I think there are some cool aspects glossed over.

1.2.1 The multimodality problem:

Text is inherintly multimodal. The example the paper gives is translating "Thank you." into German: {Danke, Danke schon, Vielen Dank}. AT avoids this using teacher forcing, but I believe the way the paper presents it is interesting – they see it as inputting a latent variable, z, which tells the decoder how to generate with data about the target sentence. They state that this variable should have some contraints:

1. It should be available, or trivial to generate, at test-time;
2. It should help, but not trivialize, the models generation of the target sentence.

The way the authors model this is by "fertilities".

Fertilities: As seen in figure (1), and mentioned in the high-level overview, for each word in the source sentence, the number of words in the target sentence this corresponds to is predicted (and used at test time). Similar to the amount of crops than can be grown in a single plot of soil, how many words in the target sentence stem from this one? This introduces a weak form of the above wanted z, considering, during train-time, this is explicitly calculated using some fast alignment algorithm.

1.2.2 Knowledge distillation and NPD:

And here is the nasty part of the paper. NAT doesn't work well without knowledge distillation – the paper gets an improvement of over 7 BLEU (18 → 27) via translating the entire corpus using an autoregressive model (and other small tweaks). NPD, however, is a rather smart idea (if inefficient): since regular transformers run fully in parallel at train-time, use a regular transformer to "score" model translations (with sampled fertility), then select the best one. This roughly halves model generation speed, even with increased compute power.

Distillation b=1	Distillation b=4	Decoder Inputs +uniform	Decoder Inputs +fertility	+PosAtt	Fine-tuning +$\mathcal{L}_{\text{KD}}$	Fine-tuning +$\mathcal{L}_{\text{BP}}$	Fine-tuning +$\mathcal{L}_{\text{RL}}$	BLEU	BLEU (T)
				✓				≈ 2	
		✓		✓				16.51	
			✓	✓				18.87	
✓		✓		✓				20.72	
	✓	✓		✓				21.12	
✓			✓					24.02	43.91
✓			✓	✓				25.20	45.41
✓		✓		✓	✓	✓		22.44	
✓			✓	✓			✓	×	×
✓			✓	✓		✓		×	×
✓			✓	✓	✓	✓		25.76	46.11
✓			✓	✓	✓	✓	✓	**26.52**	**47.38**

Figure 2: Performance of NAT model

Models	WMT14 En→De	WMT14 De→En	WMT16 En→Ro	WMT16 Ro→En	IWSLT16 En→De	Latency	Speedup
NAT	17.35	20.62	26.22	27.83	25.20	39 ms	15.6×
NAT (+FT)	17.69	21.47	27.29	29.06	26.52	39 ms	15.6×
NAT (+FT + NPD $s = 10$)	18.66	22.41	29.02	30.76	27.44	79 ms	7.68×
NAT (+FT + NPD $s = 100$)	19.17	23.20	29.79	**31.44**	28.16	257 ms	2.36×
Autoregressive ($b = 1$)	22.71	26.39	31.35	31.03	28.89	408 ms	1.49×
Autoregressive ($b = 4$)	23.45	27.02	31.91	31.76	29.70	607 ms	1.00×

Figure 3: Best NAT models performance

1.2.3 Attention:

This paper does tweak the attention a bit. Notably:

1. There is an attention operation *entirely* over positional encoding;
2. In self-attention in the decoder, the word cannot attend to itself;