Notes: "MASS: Masked Sequence to Sequence Pre-training for Language Generation"

Sun

April 21, 2020

1 Notes:

1.1 Summary:

1. MASS: **MA**sked **S**equence to **S**equence Pre-training for Language Generation
2. Unsupervised. No matching sentence pairs needed.
3. Harnessing the magic of explicit biases, MASS turns a simple masked-language-modeling (MLM) objective into a completely moral generative one.
4. Mask random continuous segments of text, then train a transformer to recreate the masked sections.
5. State of the art in some translation directions, versatile setup, even beat old supervised attention models.

1.2 High level:

History and motivation: At first sight, it's quite easy to mistake MASS for merely an MLM objective. Reiterating from last week, MLM objectives commonly involve masking random words on the output and training the model to recreate them.

The objective of MASS, more formally put, is to randomly replace a *continuous* segment of text with the special `[MASK]`token, then task the decoder to recreate this entire segment.

The magic of this objective is actually the bias introduced by the continuous segments. Naively, one would expect both MLM and MASS to optimize the same thing; to recreate the input. To do so, the model must have some level of understanding of what the sentence *should* be.

However, there are biases here. Let's consider the downstream task. We want to map sequence $\mathcal{A}$ to sequence $\mathcal{B}$. During training, transformer architecture has a subtle trick.

Lets say the objective was simple MLM, and we masked 2 input tokens that are not continuous. Let's say the model incorrectly predicts the first token. Due to the nature of the decoder architecture, the model actually gets to use

the *ground-truth* to predict the second token. Now, this is true for MASS as well. On some level, this is a huge positive. By giving the model a crutch during training, we allow it to learn significantly faster than otherwise.

However, this poses a problem when you look at the downstream task. For most seq2seq tasks, you won't ever want to predict two different tokens do not directly come after one another. MLM introduces a very strong bias here. In training, you will rarely see two masked tokens next to eachother.

At inference, you don't have this crutch. You simply assume the previous prediction from the model is true, and autoregressively generate the next token. If there is an error in your prediction, that error will simply compound and get worse.

Using the MLM objective, your model has never been explicitly trained to recreate continuous gaps of text. As such, it would be more prone to compounding issues.

The entire goal of MASS is to explicitly train the model in a way that better represents a seq2seq downstream task. In the end, you get less compounding errors, a better encoder, and overall, a better generative model.

What exactly is made explicit by this objective? You force the model to recreate continuous masked segments. This is exactly equivalent to the end task, assuming the model makes no errors (since you're still using the ground truth in training).

Why is this better than simply picking a very high masking rate for an MLM objective? Intuitively, it's better because the model can learn phrasing. Once again, by explicitly making the masked segments continuous, the model can learn the true autoregressive nature of sentence generation much better.

Consider the case of how different MLM is from MASS in the sentence "the killer was a vicious beast with no empathy".

With MLM, this sentence might be decomposed into:

(1) • "the killer was a vicious ____ with ____ empathy `[EOS]`

However, with mass, you might get:

(2) • "the killer was a vicious ____ ____ ____ ____ `[EOS]`

Now, look at how easy the first sentence is compared to the second one. "Beast" is an extremely obvious choice to come after "vicious", and a killer described this way *obviously* has no empathy. The 2nd sentence, however, has an insane amount of completely possible completions. "Beast" is likely still the most probable word after "vicious", but after that, there are infinitely many ways this sentence could go.

In the MLM case, the encoders task is extremely easy. In this case, it likely would look for the non-masked words and use them as "anchors", using the contextual information from them to automatically infer what is masked. In the MASS case, the encoder has a much harder problem.

The crutch given to the decoder is even more valuable for MASS. It prevents the model from getting too 'off-track', and once again, speeds up (or even makes it possible at all) training.

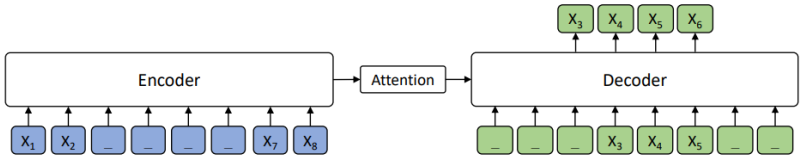


Figure 1: The encoder-decoder framework of MASS. '_' represents `[MASK]`

1.3 Low level:

Once again, the definition of a language model is useful here:

Definition 1 *A **language model** (LM) is a probability distribution over a sequence of tokens. Given a sequence of length m, it assigns a probability $P(w_1, \ldots, w_m)$.*

MASS, like BERT, does *not* result in a language model, unless the entire input is masked.

MASS takes a given sequence, $\mathcal{X}$, and masks k continuous tokens, from u to v, with $0 \leq u < v \leq m$ where m is the length of the sentence. The masked tokens are denoted $\mathcal{X}^{u:v}$. The input sequence, $\mathcal{A}$, is equal to $\mathcal{X}^{\setminus u:v}$, meaning the original sentence with tokens from u to v replaced by the special `[MASK]`token. The target sequence, $\mathcal{B}$, is simply $\mathcal{X}^{u:v}$.

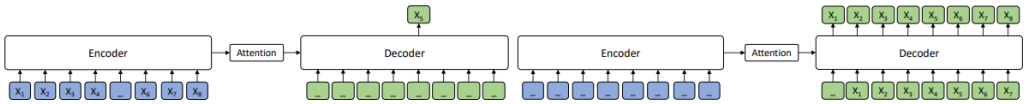
The authors argue that when k is equal to 1, it results in the traditional MLM objective; when k is equal to m, it results a standard language modeling task (such as GPT-2). As such, it is a natural extension to "mix" these two objectives into MASS; your resulting model learns context from an input, yet still learns how to generate continuous spans of text.

1.3.1 MASS as seq2seq:

We can phrase MASS as a traditional encoder-decoder setup, and compare it to MLM well at a low level.

If we consider the encoder as a model, $q_\phi(\mathbf{z}|\mathbf{x})$, that maps a given input into a set of latent variables, followed by a decoder, $p_\theta(\mathbf{x}|\mathbf{z})$, that maps these latent variables back onto some high-dimensional space, the differences between MLM and MASS become more apparent.

For example, the resulting $\mathbf{z}$ from (1) could be extremely similar to the resulting $\mathbf{z}$ from encoding the completely unmasked sentence. This is due to the large amount of context and implicit information received from the words *around* the masked words. This is not true when encoding (2); the information contained within the sentence is extremely different.



(a) Masked language modeling in BERT ($k = 1$) (b) Standard language modeling ($k = m$)

1.3.2 Uses of MASS:

Use 1: MLM: Traditional MLM is a special case of MASS. This includes encoder-only setups, such as BERT.

Use 2: LM: Traditional language modeling is another special case of MASS. This includes decoder-only setups, such as GPT-2.

Use 3: Generative encoder models Most mixes of training an encoder while also training a decoder is doable with MASS. This removes the pain of training encoders and decoders separately. This is used extensively in unsupervised machine translation, both in pretraining for supervised sections, and operating as an end-to-end objective. The paper presented 3 weeks ago by Sphinx used MASS as the pretraining objective to get the model able to bootsrap itself.

1.3.3 Fine-tuning MASS:

Simply train the model end-to-end as if it was randomly initialized; the pretraining will speedup the final training, in addition to getting better generalization in low-resource enviroments.

1.4 Empirical results:

Very good. SOTA BLEU on unsupervised English $\Longleftrightarrow$ French translation, English $\Longleftrightarrow$ German, and English $\Longleftrightarrow$ Romanian. Very good pretraining for other types of translation as well.