PLUG AND PLAY LANGUAGE MODELS: A SIMPLE APPROACH TO CONTROLLED TEXT GENERATION.
(Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu.)

presented by Alexey Korolev

11 November 2020
1 Generative language model.
2 Plug and Play Language Model.
3 Autoregressive language model ($P(x)$).
4 Attribute model $p(a|x)$.
5 Plug and Play Language Model training.
6 Results.
Nowadays big generative language model archive great performs. But if we want to specify their work some problem arises. We can’t just finetune this model for our task because:

- This model is very big (over a billion parameters).
- Require massive amounts of computing resources.
- On enormous data sets which are often not publicly released.
Authors propose effective, easy to use method Plug and Play Language model. We need the next components:

1. autoregressive language model
2. attribute model.

We steer the enormous language model by gradients.
Autoregressive language model \( P(x) \)

Autoregressive Model is merely a feed-forward model, which predicts the future word from a set of words given a context. Start with your seed \( x_1, x_2, \ldots, x_k \) and predict \( x_{k+1} \). In formula that mean that we compute \( P(x_i | x_{i-1} \ldots x_{i-k}) \) and choose the biggest. Mainly generative language model is Autoregressive (GPT1&2).
In this work, we use the transformer approach. A history matrix $H_t$ to consist of the key-value pairs from the past i.e

$$H_t = [(K^{(1)}_t, V^{(1)}_t), \ldots, (K^{(l)}_t, V^{(l)}_t)],$$

where $(K^{(l)}_t, V^{(l)}_t)$ is key-value pairs from l-th layer generated at all time-steps from 0 to t.

To generate next we use formula: $o_{t+1}, H_{t+1} = LM(x_t, H_t)$

$$x_{t+1}p_{t+1} = \text{Softmax}(W o_{t+1})$$

$W$ is a linear transformation that maps the logit vector $o_{t+1}$ to a vector of vocabulary size.
Attribute model $p(a|x)$, which takes a sentence $x$ and outputs the probability that it possesses the attribute $a$. These models can be tiny and easy to train because, intuitively, classification is easier.

As topic prediction use bag of word \((p(a|x) = \sum_{i}^{k} p_{t+1}[w_i])\)

For sentiment prediction use softmax classifier.
Plug and Play Language Model training.

Main idea base on Bayes rule: \( p(x|a) \sim p(a|x)p(x) \)

There 3 steps:

1. Given a partially generated sentence, compute \( \log(p(x)) \) and \( \log(p(a|x)) \) and the gradients of each with respect to the hidden representation of the underlying language model.

2. Use the gradients to move the hidden representation of the language model a small step in the direction of increasing \( \log(p(a|x)) \) and increasing \( \log(p(x)) \).

3. Sample the next word.
Let $\Delta H_t$ be the update to $H_t$, such that generation with $(H_t + \Delta H_t)$ shifts the distribution of the generated text such that it is more likely to possess the desired attribute.

$$\Delta H_t + \alpha \frac{\nabla_{\Delta H_t} \log p(a|H_t+\Delta H_t)}{||\nabla_{\Delta H_t} \log p(a|H_t+\Delta H_t)||^\gamma} \rightarrow \Delta H_t$$

This update step can be repeated $m$ times.

$o_{t+1}$ as $o'_{t+1}$, $H_{t+1} = LM(x_t, H'_t)$, where $H'_t = H_t + \Delta H_t$. The perturbed $o'_{t+1}$ is then used to generate a new distribution.
Kullback–Leibler (KL) Divergence: \( D_{KL}(P||Q) = \sum \log\left(\frac{P(x)}{Q(x)}\right) \)

We update \( H_t \) to minimise the KL divergence between the output distribution of the modified and unmodified language models in addition to the step above.

Post-norm Geometric Mean Fusion:

Tie the generated text to the unconditional \( p(x) \) LM distribution. We accomplish this by sampling from \( x_{t+1} \sim \frac{1}{\beta} (\tilde{p}_{gmt}^{\gamma_{gmt}} p_{t+1}^{1-\gamma_{gmt}}) \), where \( p_{t+1} \) and \( \tilde{p}_{t+1} \) are the unmodified and modified output distributions, respectively.
Results

B: the baseline, unchanged GPT-2 LM, sampled once;
BR: B but sampled r times, with best sample chosen based on the LL ranking and filtering based on Dist score;
BC: update the latent representations and then sample once; and lastly
BCR: update the latent representations and generate r samples, choose the best sample based on the LL score (after filtering out samples with low Dist scores).

CTRL: a recent language model.
WD: a weighted decoding baseline in which the B LM’s outputs are weighted directly toward maximizing $p(a|x)$.
language detoxification.

Controlled storytelling.
Thank for attentions!

https://github.com/uber-research/PPLM