Toolformer Summary

Toolformer is aiming to call external APIs in a self-supervised way on LLM.

Given an API call,

$$c = (a_c, i_c)$$

where $a_c :=$ the name of the API call, $i_c :=$ the corresponding input, $r_c :=$ the corresponding result of the API call. Then, we have a linearity, that is

$$e(c) := \langle \text{API} \rangle \ a_c(i_c) \ \langle /\text{API} \rangle$$
$$e(c, r) := \langle \text{API} \rangle \ a_c(i_c) \rightarrow \ r \langle /\text{API} \rangle$$

where $\langle \text{API} \rangle$ stands for the start tokenizer, $\langle /\text{API} \rangle$ stands for the end tokenizer.

Given a dataset $C = \{x^1, ..., x^{|C|}\}$ of sequence plain texts, we would convert the datasets into an augmented dataset C^* by API calls in the upcoming ways.

I. Sampling Step

Denote that P(x) as a prompt encouraging the language model to annotate with API calls. Given an input $x = x_1, ..., x_n$ Define that

$$p_i := p_M(\langle API \rangle | P(x), x_{1:i-1})$$

where $P_M(z_{n+1|z_1,...,z_n})$ be the probability of M assigned to z_{n+1}.

Then, we set a sampling threshold τ_s s.t. all positions $I = \{i \mid p_i > \tau_s\}$.

In other words, if the first k candidate positions for the API calls probability hitting the threshold, we would remain computing; Otherwise, we would throw it out.

II. Executing Step

we execute the API calls generated from M to get corresponding executing results r.

III. Filtering Step

To filter out, we compute the cross entropy loss for M over given tokens

$$L_i(z) := -\sum_{j=1}^{n} w_{j-1} \cdot \log p_M(x_j \mid z, x_{1:j-1})$$
$$L_i^+ := L_i(e(c_i, r_i))$$
$$L_i^- := \min(L_i(\epsilon), L_i(e(c_i, \epsilon)))$$

where ϵ denotes the empty sequence.
And we set another threshold filtering threshold τ_f s.t. $L_i^- L_i^+ \geq \tau_f$.

IV. Finetuning Step

We merge the remaining API calls and insert into original input sequence $x = x_1, \cdots, x_n$, then, we construct a new sequence $x^* := x_{1:i-1}, e(c_i, r_i), x_{i:n}$.
Iteratively finetune augmented dataset C^* interleave them with the new dataset. As API calls inserted in exactly positions, enabling the LM to decide when and how to use which external tool.

V. Inference Step

Generating text after IV finetuning step, decoding until M produce the $\rightarrow$ token, then call API to get the response.

External Tools

Toolformer used question ansering, calculator, Wikipedia Search, Machine Translation and Calendar external APIs. Compared with GPT model, whatever Toolformer API calls disabled or not, Toolformer usually has a better performance, one potential reason that is because the model is finetuned on many API calls and their results, improving its own capabilities.

Conclusion

Labeling a dataset with human input incurs considerable costs. Therefore, we are exploring self-supervised learning methods to reduce computational expenses. However, Toolformer is not capable of independent learning as I had initially anticipated when I read the paper's abstract. Nevertheless, by leveraging external tools, Toolformer's performance can be significantly enhanced in comparison to conventional large models. Considering the vast potential of GPT models in the future, we can not only utilize basic API tools like calculators, calendars, and machine translations, but there are also numerous untapped external API tools that we can explore and harness to our advantage.