進捗報告

1 やったこと

- ファインチューニング後のモデルの出力安定化
- RAG の改善手法論文読み

2 ファインチューニング後のモデルの出力安定化

モデルマージの実験の際には, テキスト生成に用いるパラメータが学習時と異なることが原因で出力が安定していなかった問題があった. ファインチューニング後の出力が安定していた vicuna の tokenizer.json に合わせた設定で Elyza のファインチューニングをしてもうまくいかなかったが, special_token.json 内で Elyza の pad_token が eos_token に設定されたいた. これを見落としていたため, vicuna と同じく pad トークンを未知語を表現する unk トークンに明示的に設定したところ, 安定した出力を得ることができた.

また, 同じく Llama2 派生のモデルで学習がうまくいっていなかった stabilityai/japanese-stablelm-instruct-beta-7b [1] に関しても, vicuna の設定に合わせることでファインチューニング語の出力を安定させることができた. japanese-stablelm-instruct-beta-7b の tokenizer の特徴として special_token の normalized が true となっていた. Github の issue で transformers の開発者の方が 「You should set normalized = False for the token」 と言っていた[2]のでこれが悪さして出力に影響が出ていたと考えられる.

これにより, モデルマージで使えるモデルが vicuna しか無い問題は解消された. この tokenizer の設定を流用することで Llama2 以外のアーキテクチャの LLM でもファインチューニングが安定させることができると考えられる.

3 RAG の改善手法

新規性を出す場所として RAG とモデルマージのアルゴリズムの改良が考えられる.

後者は村田くんとかぶりそうなので, RAG の部分で新規性を出すことを考えている. 現在の RAG の問題点は以下の通り.

- クエリに関係ないドキュメントを引いてくる (今は閾値処理で緩和)
- ドキュメントに関連しない回答をする.
- キャラクターの設定でクエリと似た文しか引いてこれない

  例えば, キャラクター設定の中に 「誰にでも明るく接する」といったどのクエリに対しても応答の参考になるような設定であっても「こんにちは!」のようなクエリに対してその設定を引くことができない.

[1] https://huggingface.co/stabilityai/japanese-stablelm-instruct-beta-7b
[2] https://github.com/huggingface/tokenizers/issues/1408

3.1 本研究で参考にできそうにない RAG の改善手法

少ないパラメータ数のモデルで小さいコストで推論することが本研究の目的となるため, LLM を用いてクエリを Retrive しやすい形に変換する手法 [1] や Retrive したドキュメントを LLM を用いてランキングづけする方法[3] など, 推論時に外部の LLM を用いる手法は本研究では参考にならない.

LLM のテキスト生成の部分クエリの際に参考にならないドキュメントを無視する, 引いてきたドキュメントが必要かどうか考慮して必要であるならその内容を踏まえた回答をするといった LLM の応答部分の改善手法として self-RAG[2] がある. self-RAG では文章の生成の途中で reflection token を混ぜ込めるようにするファインチューニングを推論を実行する LLM に施すことで, 検索した文章が必要かどうか, 回答がドキュメントの内容に即しているか判断するといった処理を可能にしている.

実際にファインチューニングした Llama2 のモデル[4]や, コードなどが公開されていたので触っている.

3.2 Retriver の部分の改良

「こんにちは」のクエリに対して 「誰にでも明るく接する」 といったような設定を引いてくることが望ましいのは Character-LLM のタスクならではの問題で, RAG が用いられる QuestionAnswering のタスクでは見られないため新規性を出すならこの部分であると考えている. 現在考えているのが, 事前に GPT-4o などでデータベース内の設定に対して「設定としての汎用性の高さ」のようなものをスコア化し, 検索の際にベクトル間の内積やコサイン類似度に加えて, 汎用性のスコアなどを考慮した Retrive の仕方を考えている. しかし現在はキャラクターのデータベースがキャラクターの設定を一文ずつ計 60 個というようにかなり簡潔なものになっているため, このデータベースを前提としたスコア化のアプローチは少し卑怯な気がする.

LLM を fix して Retriver 側を学習する REPLUG といったアプローチもあったがまだあまり調査できてないです.

参考文献

[1] Zackary Rackauckas. Rag-fusion: A new take on retrieval augmented generation. *International Journal on Natural Language Computing*, Vol. 13, No. 1, p. 37–47, February 2024.

[2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023.

[3] https://python.langchain.com/v0.2/docs/integrations/document$_transformers/rankllm-rerank$

[4] https://huggingface.co/selfrag/selfrag_llama2$_7b$