Decision Transformers for Glut Control in the Active Logic Machine

Justin D. Brody[1], Donald Perlis[2]

[1]Franklin and Marshall College, PO Box 3003, Lancaster PA 17604
[2]University of Maryland Institute for Advanced Computer Studies, College Park MD 20740

1. Introduction

In this paper, we build from the preliminary work in [1] on using deep learning architectures to control *inferential glut* in a logical reasoning agent. Inferential glut is the well-known problem that symbolic reasoners encounter combinatorial explosions in the numbers of formulae they have to process. In [1] we proposed a neural architecture that employed a simple deep-reinforcement-learning network to prioritize potential inferences and control glut. This architecture was able to successfully control *pre-inferential glut* (inferential glut arising from inefficiencies in the reasoner's algorithm) and a limited form of inferential glut (that didn't use state). Work on this architecture continues; in this work we explore the use of Decision Transformers [2] as a replacement for the original deep reinforcement learning model. The decision transformer models reinforcement learning as a sequence modelling problem – specifically it employs transformers to generate the action likely to lead to a given return given the trajectory of (states, actions, returns) seen so far. In [2], the authors showed that using a pre-trained and fine-tuned transformer (such a GPT2) can lead to near state-of-the-art on offline reinforcement learning tasks.

Since glut control as envisioned is largely a matter of selectively attending to the most promising avenues of reasoning, the attention-based mechanisms which underwrite transformers seem a natural fit for our application. In this paper we will explore the effectiveness of using a fine-tuned GPT3 model [3] to predict salient actions as a proof-of-concept.

2. Active Logic

Our architecture is based on Perlis' Active Logic reasoning engine ALMA [4], [5], [6], [7]. Active Logic is a time-sensitive non-monotonic logic which extends first-order logic in a number of ways which make it well-suited for situated agents reasoning "in the wild". Many of the features of active logic will not be relevant for the current pilot work (but will be incorporated in future

In A. Martin, K. Hinkelmann, H.-G. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (Eds.), Proceedings of the AAAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence (AAAI-MAKE 2022), Stanford University, Palo Alto, California, USA, March 21–23, 2022.
justin.brody@fandm.edu (J. D. Brody)

work). For our present purpose we can think of active logic as a first-order reasoner which acts as a step logic. In particular, for our purposes active logic will proceed by enacting a single application of a single deduction rule to the current knowledge base. For example, suppose a knowledge base $\mathcal{K}$ consists of the following sentences at time 0:

1. `location(a).`
2. `location(X) -> location(left(X)).`
3. `location(X) -> location(right(X)).`

Then `location(a)` can be used in conjunction with any one of these to derive `location(left(a))` or `location(right(a))`, So to reach the next timestep, the active logic machine might apply a single deduction and add `location(right(a))` to its knowledge base[1].

For simplicity, in this preliminary study we restrict our attention to reasoning the proceeds completely by way of applications of the *resolution* inference rule. See Section 9.5 of [8] for a detailed discussion of resolution.

While resolution is a complete inference rule it is also particularly prone to producing glut as we shall see. We will focus our attention in this paper on the set of axioms $\mathcal{K}$. Although simple, this set of axioms provides for a meaningful study of glut-control for a few reasons, which we will examine in turn.

0: location(a)
1: location(X0) —> location(left(X0))
2: location(X1) —> location(right(X1))
3: now(0)

5: location(left(a)) (parents: [0, 1])
6: location(right(a)) (parents: [0, 2])
7: **location(X2) —> location(right(left(X2))) (parents: [1, 2])**
8: **location(X3) —> location(left(right(X3))) (parents: [2, 1])**
9: now(1)

11: location(left(left(a))) (parents: [5, 1])
12: location(right(left(a))) (parents: [5, 2], [0, 7])
13: location(left(right(a))) (parents: [6, 1], [0, 8])
14: location(right(right(a))) (parents: [6, 2])
15: **location(X4) —> location(left(right(left(X4)))) (parents: [7, 1], [1, 8])**
16: **location(X5) —> location(right(right(left(X5)))) (parents: [7, 2])**
17: **location(X6) —> location(right(left(left(X6)))) (parents: [1, 7])**
18: **location(X7) —> location(right(left(right(X7)))) (parents: [2, 7], [8, 2])**
19: location(right(left(left(a)))) (parents: [5, 7])
20: location(right(left(right(a)))) (parents: [6, 7])
21: **location(X8) —> location(left(left(right(X8)))) (parents: [8, 1])**
22: **location(X10) —> location(right(left(left(right(X10))))) (parents: [8, 7])**
23: **location(X12) —> location(left(right(right(X12)))) (parents: [2, 8])**
24: location(left(right(left(a)))) (parents: [5, 8])
25: location(left(right(right(a)))) (parents: [6, 8])
26: **location(X13) —> location(left(right(right(left(X13))))) (parents: [7, 8])**
27: now(2)

Figure 1: Growth of Knowledge Base with Classical ALMA Steps

In Figure 1, we have given a short transcript of the first few steps of ALMA's reasoning with $\mathcal{K}$ using resolution. The extremely rapid growth of the knowledge base is immediately apparent.

[1]It is worth noting that this behavior is somewhat different from the traditional behavior of ALMA, where a step would refer to applying all such options a single time; thus at time 1 we would derive `left(a), right(a)` but not, e.g., `left(right(a))`.

We also note that this growth has two sources – a growth in literals that comes from repeated application of rules 2 and 3 of $\mathcal{K}$ joined with rule 1 and a growth in universal formulae that comes from applying resolution to pairs of universal formulae. The latter type of formulae will be produced by resolution but not by forward chaining. These formulae also give a mechanism for growing the lengths of the potential consequents very rapidly. Indeed, in our study we will explore a reward structure that is based on the number of `right` functions applied to an instance of `a`; for example `location(right(right(left(a))))` will have a reward of 2. The continual creation of these universal formulae will allow us to get more highly rewarded literals in the knowledge base rapidly, but some care is needed. By themselves, these rules do not produce any reward at all because the new formula are not instantiated. So an optimal strategy for an agent reasoning for N steps will be to create the longest possible universal rules for $N-1$ steps and then instantiate the longest such rule at the final reasoning step. This requires that an agent know how many steps it will reason for and how many it has already reasoned for – in particular it will need to keep track of a state. This in contrast to a forward-chaining based agent, whose optimal policy will be to repeatedly employ rule 2 from $\mathcal{K}$ to get a reward of $\Theta(n)$ in N steps of reasoning.

It is important to note that the growth of sentences from $\mathcal{K}$ is so rapid that glut control becomes absolutely necessary – our workstation with over 300Gb of memory crashed after a few steps without it. Indeed, we conjecture that the growth in the number of sentences in the knowledge base is doubly exponential with $\Theta(2^{2^n})$ sentences in the knowledge base after n steps of (classical) ALMA reasoning.

3. Architecture

Our glut-control architecture relies crucially on identifying, at each time-step, the set of potential inferences that can be applied and viewing these as actions. By a potential inference we mean a pair of formulae which the system believes it can use in the resolution rule to create new knowledge. A list of these is stored in an internal ALMA data-structure; in [1] this list was passed through a neural heuristic function which would assign a priority to each such potential inference. Our intended use case here is simpler – we will represent the current state of the knowledge base and use a transformer to predict the potential inference which will lead to the maximal reward. The engine will then apply resolution to this potential inference.

In this preliminary study, we are specifically interested in using an autoregressive language generation model (such as one of the GPT models) to determine the best action. Specifically, we are interested in testing the viability of the Decision Transformer [2] as a way of choosing the best potential-inference. The idea of the decision transformer is to train on a sequence of trajectories

$$t_j = \Big((R_{1_j}, s_{1_j}, a_{1_j}), (R_{2_j}, s_{2_j}, a_{2_j}), \ldots, (R_{n_j}, s_{n_j}, a_{n_j})\Big)$$

where s_{i_j} is the ith state of the jth trajectory, a_{i_j} is the ith action of the jth trajectory, and R_{i_j} is the $i$the return-to-go of the $j$the trajectory (the total reward that will be generated in future states on the current trajectory). Actions are predicted by feeding in a desired return-to-go and state and having the model complete the tuple with the optimal action. The success of

this paradigm relies on a remarkable ability of large language models to successfully model out-of-distribution data [9]

In our particular case, our states will correspond to the current content of the knowledge base, actions will correspond to potential inferences and returns will be based on the reward function mentioned above which counts the number of `right` functions in location instantiated with `a`. As an initial test, we represented each triple as text and fine-tuned OpenAI's GPT-3 model using their available tools [10]. We separated each return from the state with the string `====`, separated each state from the corresponding actions with `::::`, enclosed each (return, state, action) triple in angle brackets `<` , `>` and separated each triple in a trajectory with the string `||||||||` (the intuition is to make these syntactic features obvious to the model). An example trajectory representation is given here (with inserted whitespace for readability) :

```
[<3====
  l(a); l(X) --> l(f(X)); l(X) --> l(g(X));  now(0)::::
  l(X) --> l(g(X)); l(X) --> l(f(X))>||||||||
 <3====
  l(a); l(X) --> l(f(X)); l(X) --> l(g(X)); now(1); l(X) --> l(f(g(X)))::::
  l(a); l(X) --> l(f(g(X)))>||||||||
 <2====l(a); l(X) --> l(f(X)); l(X) --> l(g(X)); l(X) --> l(f(g(X)));
  l(f(g(a))); now(2)::::
  l(X) --> l(f(g(X))); l(X) --> l(g(X))>]
```

Here the first line of each triple is the return, the next line is a semi-colon separated list of sentences in the knowledge base. The third line indicates which pair of sentences are combined with resolution to produce new knowledge; we can see that in the second tuple, when the choice increases the reward by one the return in the next triple correspondingly decreases by one.

We represented 4000 such trajectories, each randomly generated with 6 reasoning steps and the same initial knowledge base. These representations were then used as a dataset for fine-tuning GPT3 (using an empty string as prompt and the representation as the completion). This was close to the maximum number of trajectories allowable by the default parameters set by OpenAI; this dataset was then used to fine-tune a "curie" model of GPT3.

4. Preliminary Results

To test our preliminary system, we entered the prompt `[<6====l(a); l(X) -> l(f(X)); l(X) -> l(g(X)); now(0)::::` and asked GPT3 for a completion. Based on our syntax, this is essentially asking for an inference to take that will result in a return of 6 based on the initial knowledge base. We set the temperature to 0 and the response length to 1900 tokens, using `)>]` as a stop sequence. The beginning of the provided completion is `l(X) --> l(f(X)); l(X) --> l(g(X))>`. This indicates that the fine-tuned model has learned the syntax of our representation and selected a valid action – the pair of sentences is in the knowledge base and will resolve to give a new sentence `l(X) --> l(g(f(X)))`. Further, the next part of the completion (with added whitespace) given is

```
<6====
 l(a); l(X) --> l(f(X)); l(X) --> l(g(X)); now(1); l(X) --> l(g(f(X)))::::
 l(X) --> l(g(f(X))); l(X) --> l(f(X))>||||||
```

this further shows that the model is capable of computing the correct returns since the inference selected will not lead to any immediate reward.

Subsequently, the model chooses

- `l(X) --> l(f(g(f(X))));  l(X) --> l(g(X))` leading to `l(X) --> l(g(f(g(f(X)))));`
- `l(a);  l(X) --> l(g(f(g(f(X)))))` leading to `l(g(f(g(f(a)))))` and a reward of 2;
- `l(g(f(g(f(a)))));  l(X) --> l(g(f(g(f(X)))))` leading to `l(g(f(g(f(g(f(g(f(a)))))))))` and a reward of 4

It is noteworthy that each choice the model makes is valid and that the correct returns and logical consequences are modelled. Importantly, this particular sequence does not appear in the training set, so the model has not simply memorized a solution. The overall robustness of this learned model will be a subject of future analysis.

As a final test, we note that the trajectory in the training set with the highest return has a return of 21. Can our model beat this and get a return of 22 in 6 moves? The model makes the following choices:

- `l(a);  l(X) --> l(g(X))` leading to `l(g(a))` and a predicted return of 22
- `l(g(a));  l(X) --> l(f(X))` leading to `l(f(g(a)))` and a predicted return of 21
- `l(f(g(a)));  l(X) --> l(f(X))` leading to `l(f(f(g(a))))` and a predicted return of 19
- `l(f(f(g(a))));  l(X) --> l(f(X))` leading to `l(f(f(f(g(a)))))` and a predicted return of 16
- `l(f(f(f(g(a)))));  l(X) --> l(f(X))>` leading to `l(f(f(f(g(a))))))` and a predicted return of 12
- `l(f(f(f(f(g(a))))))); l(X) --> l(f(X))`

(the final element could not be completed because of the token limit) We note that:

- The model wastes the first move by adding a `g` to get `g(a)`.
- The model employs the same rule every step, which is the same rule that a greedy policy would use.
- The models returns are significantly off in many cases; if we take the decrease in returns as an estimate of the reward for each action.

5. Conclusions and Future Work

Overall, our simple test shows promise but also a great deal of room for improvement. We emphasize that our preliminary investigation only trained on a relatively small model ("curie") and with a small amount of data (4000 trajectories). This already shows a limited ability to correctly model returns and actions in our scenario. Notably, every action picked was viable; the syntactic structure of every completion was correct; and we have found no instances in which the model made a mistake about what new sentence would be entered into the knowledge base as a result of choosing a particular action (that is, it has learned the syntactic structure of resolution).

Further work will expand to larger models, both by using OpenAI's "davinci" model with larger training sets and by fine-tuning open source large language models from HuggingFace. We will also explore training GPT models from scratch to try to measure the degree to which our model is exploiting the general modelling capacities of the pre-trained networks.

References

[1] J. D. Brody, B. Austin, O. Khater, C. Maxey, M. D. Goldberg, T. Clausner, D. Josyula, D. Perlis, Using neural networks to control glut in the active logic machine (2021).

[2] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, I. Mordatch, Decision transformer: Reinforcement learning via sequence modeling, arXiv preprint arXiv:2106.01345 (2021).

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, arXiv preprint arXiv:2005.14165 (2020).

[4] J. J. Elgot-Drapkin, D. Perlis, Reasoning situated in time i: Basic concepts, Journal of Experimental & Theoretical Artificial Intelligence 2 (1990) 75–98.

[5] J. Elgot-Drapkin, S. Kraus, M. Miller, M. Nirkhe, D. Perlis, Active logics: A unified formal approach to episodic reasoning, Technical Report, 1999.

[6] M. L. Anderson, W. Gomaa, J. Grant, D. Perlis, Active logic semantics for a single agent in a static world, Artificial Intelligence 172 (2008) 1045–1063.

[7] K. Purang, Alma/carne: implementation of a time-situated meta-reasoner, in: Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001, IEEE, 2001, pp. 103–110.

[8] S. Russell, P. Norvig, Artificial intelligence: a modern approach, Prentice Hall, 2020.

[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (2019) 9.

[10] Openai api document, 2021 [Online]. URL: https://beta.openai.com/docs/guides/fine-tuning.