CIS 700-001 Project
Neuro-Symbolic Dual-System on Task-Oriented Dialogue Generation

Yifei Li, Yukai Yang, Weiqiu You, Liyang Zhou, Haoyu Wang
School of Engineering and Applied Science
University of Pennsylvania
{liyifei,yukaiy,weiqiuy,zhliyang,why16gzl}@seas.upenn.edu

Abstract

Dialog generation based on pretrained language models often factually conflicts with previous generations and is sometimes redundant. To address these issues, we introduce a novel neuro-symbolic dual-system to improve the consistency and coherence in task-oriented dialog generation. By incorporating human knowledge and domain-specific constraints into our proposed model, we observe experimentally that the dual-system greatly improves the generation quality on bAbI story generation. On the task-oriented dialog benchmark MultiWOZ, our model outperforms the single-system baseline in inform and success rate by a large margin, demonstrating the effectiveness of knowledge-infused dialogue generation via neuro-symbolic reasoning.

1 Project Description

Recent pretrained large language models like GPT-3 (Brown et al., 2020) are able to generate fluent sentences. However, when we use them to generate longer text, they often lose track of entity states and conflict with previous generated contents (Nishino et al., 2019). As there are only a few parameters to manipulate (e.g., temperature, top p, frequency and presence penalty in GPT-3 playground[1]) in the generation process of pretrained language models, it is hard to guarantee that some of the attributes are consistent through out the generated passage or conversation, where controllable decoding strategies (Ghazvininejad et al., 2017; Holtzman et al., 2018; Fan et al., 2018; Gu et al., 2017) and prompt tuning (Shin et al., 2020; Jiang et al., 2020) also fall short in enforcing domain-specific constrains. For example, the context might contain the fact that A is B's father, and B is C's daughter, and generate some spurious sentences saying that A's son B is doing something, while B is in fact A's daughter. This issue exists in all generation tasks, including task-oriented dialog generation which is the main focus of this work.

Our goal of the project is to use neuro-symbolic representation to improve consistency and coherence in dialog generation. Contrary to how language generation are hard to control, symbolic structures can explicitly represent facts of the current state, and be used to modify the previous states. Combining the respective strengths of neural and symbolic approaches is beneficial to build more robust models that can more effectively incorporate domain knowledge. We aim to take advantage of both rich representation in neural models and explicit operations in symbolic models. The research question we want to ask in this project is: whether and how we can use symbolic representation to improve consistency of neural dialog generation.

Our contributions are:

1. Adapted neuro-symbolic dual system to task-oriented dialogue generation.
2. Improved inform and success rate comparing to the single system.
3. Showed that end-to-end neural dialog systems have a lot of inconsistency errors, and that both dialog consistency and informativeness of the agent's question towards the goal can be improved using symbolic checks.

2 Related Work

We discuss two lines of relevent research.

Neuro-symbolic systems. Neuro-symbolic paradigm is a rising promising method in artificial intelligence (Garcez and Lamb, 2020). Researchers have proposed to combine statistical learning and symbolic reasoning in the AI community, with pioneer works devoted to different aspects including representation learning and reasoning (Sun, 1994; Garcez et al., 2008; Manhaeve et al., 2018), abductive learning (Dai and Zhou, 2017; Dai et al.,

[1] https://beta.openai.com/playground

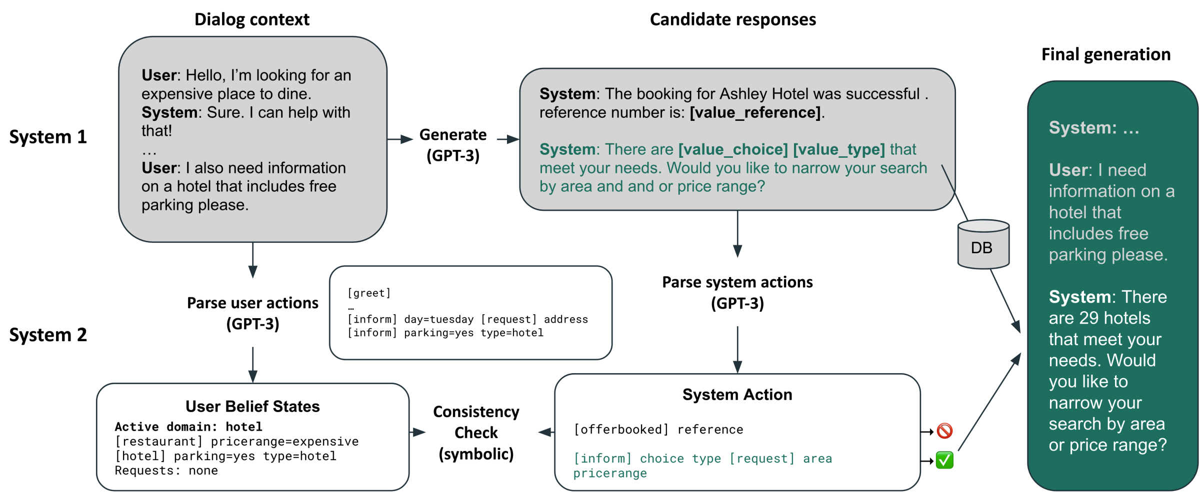

Figure 1: An overview of our approach.

2019; Zhou, 2019), knowledge abstraction (Hinton et al., 2006; Bader et al., 2009), knowledge transfer (Falkenhainer et al., 1989; Yang et al., 2009), *etc*. In this work we aim to leverage this combination to enhance the generation quality in task-oriented dialogue generation.

Dialogue generation. End-to-end task-oriented dialog can be formulated similarly to story generation by prepending the past context before the current user utterance, and generating the next system response. UBAR (Yang et al., 2020) adopts GPT-2 and improves upon this by utilizing slots and system actions from MultiWOZ 2.0 (Budzianowski et al., 2018). For each dialog turn, UBAR takes in the current user utterance, and generates slots and system actions along with a response. Slots are used as the search constraints to query a database for matches, which are fed into UBAR to further control generation. Each turn in the past dialog context also contains the generated slots, system actions, and query results. Conditioning on generated slots and actions helps GPT-2 track states without inferring them from all previous utterances.

Nye et al. (2021) proposes to use a cognitive science-inspired dual-system paradigm in text generation. Along with a system 1 neural language model like GPT-3 that is used to propose candidates for the next sentence, symbolic representations are continuously extracted from the existing sentences and used to update a world model in system 2 and check whether or not each new sentence generated conflict with the existing world model. This work is in the story/text generation domain, while our work will adapt the same concept for the dialog domain as our innovation part.

3 Methods

We first reproduce the text generation results of one of the datasets used by Nye et al. (2021): bAbI. We compare the GPT-3 baseline and the dual-system approach. Then we extend the approach to dialog. We adapt UBAR to GPT-3 babbage as our dialog baseline/system 1.

As shown in Figure 1, we first use GPT-3 to parse user actions from the past dialog context, which converts unstructured dialog text into structured information containing actions and slots of each user turn (e.g., "inform", "greet", "day="). We construct the user belief states as our symbolic representation, which tracks active domain, slots, and requests. The belief states are updated as a result of all the parsed actions, including user and system actions. We then use the dialog context to generate 5 candidate responses for the current turn using the system 1 model before feeding them to a system action parser. Further, system 1 conditions on the slots tracked in the belief states as opposed to the generated ones in the single-system case. We conduct a consistency check where the symbolic constraints come to power. After selecting a candidate response, we query to database to fill entity placeholders and present it to the user.

3.1 Parser

We reformat the state labeling of the dialog data so that they can be parsed from the sentences through GPT-3 (Ada) fine-tuning: for the dialog generation task, involving both the customer/user and the representative/system, we trained two parsers separately to parse system response and user utterances into system act and user act respectively. For the user act parser, we prompted GPT-3 (Ada) with both system response from the last turn as well as the current user utterance to obtain the user act. For the system act parser, we use user's current utterance as well as system's current response to parse the system act. Previously in the diagram, we see that both system act and user act (belief states) are represented as $domain - slot, value$ pairs. Following UBAR's design choices, we also decoupled domain and slot names, e.g.$[hotel][slot\_name]$ instead of $[hotel - slot\_name]$, allowing us to simplify the set of possible slots and perform domain classification separately.

3.2 Consistency checker

To address the problem of narrowing down available choices in the goal-oriented dialogue generation tasks, our consistency checker prioritizes candidate system responses that request users for the most number of unfilled slots required for booking.

Our consistency rule checker **invalidates** the response when the following conflicts occur:

1. Not answering (inform/recommend) when user requested a slot
2. Ask questions about a slot (request/select) that is filled/entailed/user-ignored
3. Change topic: e.g. hotel $\rightarrow$ restaurant when turn domain is hotel
4. Offerbook if there is no database match
5. Nooffer when there are database matches
6. Nooffer and offerbook both present
7. Offerbooked and nobook both present
8. Inconsistent offerbook & nooffer actions across different turns for the same set of constraints
9. Bye/welcome etc. at inappropriate times

In addition, the following are exceptions to the above rules and should be counted as **valid**:

1. Ask about a slot if its value is "unknown"
2. Ask about a slot to relax constraints when there is no database match
3. Change topic to/from general

After each user's utterance, we update the belief state directly. After we choose the system's response, there is one more step to do: remove user's requested slots that are already answered.

4 Data

To recreate the prior work's results on story generation, we use bAbI (Weston et al., 2015), a dataset of synthetic tasks that involves chaining facts, simple induction and deduction.

To evaluate on the task of dialog generation, we use the commonly used benchmark dataset MultiWOZ 2.2 (Zang et al., 2020). MultiWOZ 2.2 is a task-oriented dialog dataset containing over 10,000 dialogs. The dataset spans 8 domains, including restaurant, attraction, hotel, taxi, train, bus, hospital, and police. MultiWOZ 2.2 corrected most annotation errors in MultiWOZ 2.1 (Eric et al., 2020) and MultiWOZ 2.0 (Budzianowski et al., 2018). Notably, MultiWOZ contains state/slot annotations for each dialog turn. For example, in a restaurant booking scenario, states can include information like booked time and food preference that can be updated during the course of the dialog. It comes with dialog states that we can track with our symbolic representation. Following (Yang et al., 2020), we preprocess and delexicalize system responses by replacing specific slot values with their corresponding placeholders using the span annotations in MultiWOZ 2.2. For example, a hotel name "acorn guest house" is replaced with $[value\_name]$. Delexicalization is important for task-oriented systems because these placeholders can be filled according to database query results afterwards. Since MultiWOZ 2.2 has non-categorical slots with values copied from the original utterances, which can be different from the database entities, we implement query using fuzzy matching.

We normalize slot values across slots and action annotations, and convert user/system actions to a serialized format for parser training. However, some action labels in MultiWOZ 2.2 have missing slot values, and many incorrect annotations still persist, especially user actions. Therefore, we define a set of actions that must have parameters such as 'inform' and 'request,' and clean the dataset by removing all these actions without slot values. We also add missing user actions by inferring them from changes in slot values.

5 Evaluation

We use inform rate and success rate to evaluate dialog generation (Nekvinda and Dusek, 2021). A dialog is marked as successful if for each domain in the user's dialog goal: (1) the last offered database entities are a subset of the goal entities, and (2) the system mentioned all slots requested by the user. The inform rate is the proportion of dialogs complying to (1). Success rate is the proportion of successful dialogs that satisfy both conditions. In other words, inform and success measure the rate of goal completion. Further, we use BLEU to measure the fluency of the generated responses. Note that BLEU is not as important as inform and success because groundtruth responses are often not the best ones to satisfy dialog goals.

Due to limited time and budget, we only evaluate on 2 of the 7 domains, restaurant and hotel, from MultiWOZ, which we think are the most important. Developing symbolic rules for all domains would require a substantial amount of effort. Note multi-domain dialogs that interleave restaurant and hotel domains are also considered. To this end, we develop all symbolic rules on the dev set and use 178 dialogs from test set for evaluation.

6 Results

6.1 bAbI

We aim to test the neuro-symbolic dual-system on bAbI story generation following Nye et al. (2021)'s work. We first parse sentences to facts in bAbI dataset. The parsing, partially showed in Table 1, is inclusive and reasonable, given the fact that there are only three actions: go, pickup and drop. All the extracted facts are memorized in a world model.

Sentence	Fact
Mary moved to the bathroom.	go(Mary, bathroom)
Sandra journeyed to the garden.	go(Sandra, garden)
Mary got the football there.	pickup(Mary, football)
John dropped the apple.	drop(John, apple)

Table 1: Parsing sentences to facts in bAbI.

Then we improve the consistency and coherency of the story generation from bAbI neuro-symbolically. In detail, given the story head, the fine-tuned GPT-3 model will generate the next sentence and the symbolic consistency checker will accept or reject this sentence based on whether its parsed fact is consistent with the previous world model. For instance, as demonstrated in Figure 2, we accept sentence "Mary went to the garden" since Mary was not already at the garden, and reject sentence "Mary left the apple there" since the previous story never mentioned that Mary picked up or carried the apple. If the sentence is accepted, the world model would be updated simultaneously. The next sentences are continually generated and checked until the number of accepted sentences reaches the requirement.

John travelled to the office.
Sandra went back to the office.
John journeyed to the hallway.
✓ Mary went to the garden.
✓ Sandra journeyed to the garden.
✓ John travelled to the office.
✗ Mary left the apple there.
✗ Sandra left the football there.
✗ Sandra left the football there.
✓ Daniel went to the garden.
✗ Mary left the milk.
✓ John journeyed to the hallway.
✓ Daniel moved to the office.
✗ Sandra moved to the garden.
✗ Mary went to the garden.
✗ Mary left the apple there.
✗ Mary moved to the garden.
✗ Mary left the milk.
✗ Sandra moved to the garden.
✗ Mary left the milk.
✓ Sandra grabbed the milk there.

Figure 2: bAbI story generation via dual-system. The first three sentences are provided as story head, then the system continually generates and checks the next sentence on consistency, where ✓ and ✗ indicate that the generated sentence is *accepted* or *rejected* respectively.

6.2 MultiWOZ

Table 2 shows that dual system outperforms single system in 5% inform rate, 15.8% success rate and 9.9% combined, showing that consistency and coherency have been improved. Also, dual system maintains almost comparable BLEU to single system.

Model	Inform	Success	BLEU	Combined
Single-system	79.8	61.2	12.2	82.7
Dual-system	84.8	77.0	11.7	92.6

Table 2: Comparison of generation results on restaurant and hotel domains in MultiWOZ 2.2.

6.2.1 Single system

Requesting a slot already filled. Sometimes, although the user already told the system that they want some criteria for the reservation, the system ask again for the same information. For example, in Figure 3, the user has already informed the system that they want to book the restaurant at Tuesday 2 pm, but the system asked again what day the user wanted to book.

Cannot change domain. When we train the generator, we use the domain as part of its input. Since

user: can you recommend some place to eat
***system**: sure, what type of food are you looking for?*
user: chinese
***system**: there are many chinese restaurants, what price range are you looking for?*
user: moderate
***system**: there are 1 chinese restaurants in north. do you have a particular area in mind?*
user: No, what is the name
***system**: shanghai family restaurant is a nice place.*
user: Can you recommend another one
***system**: lan hong house is another option. can i give you their address?*
user: sure book it please for 2 people
***system**: what day and time are you looking to book?*
user: tuesday 2 pm
***system**: what day would you like it for?*
user: I said tuesday
***system**: i was able to book that for you for 2 people on tuesday at 2pm. your reference number is: [value_reference]. is there anything else i can help you with?*
user: exit

Figure 3: Single system error: Requesting day for the reservation which has already been filled

user: hello
***system**: happy to help! what can i help you find?*
user: I want a place to eat
***system**: what kind of food are you looking for?*
user: expensive food
***system**: i am sorry, i have no listings for that. would you like to try a different type of food?*
user: no
***system**: is there anything else you would like to know?*
user: no bye
***system**: goodbye.*
user: exit

Figure 4: Single system error: Domain restriction – We need to set the domain in advance and the system won't be able to switch from general to other domains (and vice versa)

we do not have a parser to parse the most recent user input, we are not able to answer questions about "restaurant" if the domain is set to "general" at first. This can be seen in Figure 4. For single-system, the groundtruth turn domains must be used during testing.

6.2.2 Dual system

There are two types of errors for the dual system: parser error and consistency failure error. When the system parses user/system utterances into user/system actions, the parser can result in wrong parses, or there can be an ambiguity and the parser chooses the one that is wrong according to the context.

Wrong domain gets parsed. For general responses like "not really", it can be an utterance for many different domains like hotel and restaurant.

user: can you recommend some place to eat
user_act: {'restaurant': {}}
✓***system**: there are many options to choose from. are you looking for a specific area of town?*
chosen sys act: {'restaurant': {'request': {'area'}}}
user: not really
user_act: {'hotel': {'inform': {'area': 'dontcare'}}}
✗ ***system**: there are many different options . do you have a specific cuisine in mind ?*
✗ ***system**: what type of food would you like ?*
✗ ***system**: there are no restaurants serving [value_food] food . what type of food would you like to try ?*
✗ ***system**: there is many of options , any food preferences ?*
✗ ***system**: there are many restaurants in town that serve cheap food. what kind of food do you like?*
chosen sys act: {'restaurant': {'inform': {'pricerange', 'choice'}, 'request': {'food'}}}

Figure 5: Dual system error: parser error – parsed user action is in a different domain from previous dialogs

But if we take into the whole context in Figure 5, we can see that the user is looking for a restaurant while the system parses their last utterance as talking about a hotel reservation. This happens because we only feed in the last system response as the context, and that system response "there are many options to choose from. are you looking for a specific area of town?" is also not specific to restaurant. Had we put in the domain "restaurant", it would have been different.

Capturing system switched domain error (combined with parser error). The system should keep talking about one domain instead of switching domains. This rule has been captured in our system 2. However, when the parser wrongly parses the user's intended action, checking whether or not the system's response has switched domain no longer carries out its original intention. Let's look at Figure 5 again. Because the user's response "not really" is parsed to be in the domain of "hotel", all five system responses related to food are rejected. The system ends up choosing the last one only because it is forced to choose from the five candidates. The end result turns out to be correct, but there could be situations where there are hotel related responses accepted and then it's wrong.

Intent action not parsed. When the intent action of user is not parsed out correctly, the system can ask repeated question. For example, in Figure 6, after the system says that there are 4 restaurants and it suggests "jinling noodle bar", the user asks "what else are there", which should be answered with a different restaurant name. However, the parser does not parse from "what else are there"

user: I'm looking for restaurants
user_act: {'restaurant': {}}
✓*system: i have 4 restaurants. what area are you looking for?*
chosen sys act: {'restaurant': {'inform': {'choice'}, 'request': {'area'}}
user: no specific area
✗ user_act: {'restaurant': {'inform': {'area': 'dontcare'}}}
✗ ***system**: i am sorry , i am having trouble finding any restaurants in the centre that meet that criteria . is there a certain price range you would prefer ?* – **Requesting a slot that is already filled**
✗ ***system**: i have 3 restaurants that serve chinese food in the centre and 3 in the centre . is there a part of town you prefer ?* – **Requesting a slot that is already filled**
✗ ***system**: i have 4 chinese restaurants . what is your price range ?* – **Requesting a slot that is already filled**
✗ ***system**: there are 4 options . what is your pricerange so we can narrow it down ?* – **System switched domain**
✓ ***system**: there are 4 restaurants . i suggest the jinling noodle bar . would you like to reserve a table ?*
chosen sys act: {'restaurant': {'inform': {'choice'}, 'recommend': {'name'}, 'offerbook': set()}}
user: what else are there
user_act: {'restaurant': {}}
✓ ***system**: there are 4 options. how about the jinling noodle bar? would you like to book a table?*
chosen sys act: {'restaurant': {'inform': {'choice'}, 'offerbook': {'name'}}}

Figure 6: Dual system error: parser error – "what else are there" should be requesting name of any other restaurants

user: can I reserve a table for 2 at a chinese restaurant
user_act: {'restaurant': {'inform': {'food': 'chinese'}}}
✓***system**: what price range were you looking at?*
chosen sys act: {'restaurant': {'request': {'pricerange'}}}
user: cheap, I have no money
user_act: {'restaurant': {'inform': {'pricerange': 'cheap'}}}
✓ ***system**: there are 4 cheap chinese restaurants in town. which area of town would you like?*
chosen sys act: {'restaurant': {'inform': {'choice', 'food', 'pricerange'}, 'request': {'area'}}}
user: what areas are there
user_act: {}
✓ ***system**: there are 3 restaurants in the south and centre. the rest are cheap. do you have a preference?*
chosen sys act: {'restaurant': {'inform': {'area', 'pricerange'}, 'select': set()}}
user: what is the restaurant in center
user_act: {'restaurant': {'inform': {'area': 'centre'}}}
✓ ***system**: i recommend golden house*
chosen sys act: {'restaurant': {'recommend': {'name'}}}
user: ok can I book it
user_act: {'restaurant': {'inform': {'name': 'golden house'}}}
✓ ***system**: sure, what day and time would you like to book? how many people will be in your party?*
chosen sys act: {'restaurant': {'request': {'people', 'day', 'time'}}}
user: money 1 pm
user_act: {'restaurant': {'inform': {'people': '1'}}}
✓ ***system**: booking was successful. the table will be reserved for 15 minutes. reference number is: cro6ehqd.*
chosen sys act: {'restaurant': {'offerbooked': {'reference'}}}

Figure 7: Dual system error: parser error – Didn't parse out the number of people and requested slots correctly

the intent "recommend {'name': '?'}". As a result, the system keeps asking about the same restaurant.

There is also a limitation of the MultiWOZ dataset that it doesn't support excluding a value. Intuitively, we might want that feature in real life conversations.

Intent slots not parsed correctly. If the parser does not parse out some slots, even if the consistency checker avoids asking for information already provided, it will not be able to avoid that. In Figure 7, the user already asks to reserve a table for 2 in the first sentence, but that is not captured by the parser. The parser only recognizes that the user wants a Chinese restaurant. Therefore at then end the system still asks how many people will be in the user's party. When the user only provides day and time (with day misspelled), the parser does not recognize the time, but instead thinks the number 1 is number of people.

Book the reservation without confirmation. Sometimes the system books a reservation without confirmation with the user. This is a feature we should be adding to our consistency checker but don't have time to add. In Figure 7, we can see that the system booked directly after the user replies with day and time, but the information is not complete at that time. Even if it were complete, the system should double check with the user.

7 Discussion

We used bAbI dataset to reproduce the story generation baseline given its simplicity. Then we can move quickly to work on the dialog dataset.

Most errors of the dual system for dialog come from the parser. There are some consistency checks that we have not yet got time to implement such as confirming booking before actually booking. However, most errors are that the parser does not give the correct domain, slots, etc. We used GPT-3 Ada, the smallest GPT-3 model, for the parsers. Using better models is more expensive but can improve our parsing accuracy. As the whole consistency checker relies on the accuracy of parsing, it may worth using a higher performed parser. Also, as we rely on GPT-3 to generate natural language parses, it is hard to control the results. Adding special tokens or using a hierarchical parser might help in future works.

For future work, a good next step would be to incorporate theory of mind. Right now we only incorporate surface forms of the dialogs. However, we can potentially represent what the user wants as theory of mind for the user. We can have two models, one representing what the user knows, and one representing what the system thinks the user wants.

Author's contribution

- Yifei Li: Found the main paper, formalized the research problem, completed the bAbI story generation on dual system, and participated on MultiWOZ dialog dual system building.
- Yukai Yang: Preprocessed the MultiWOZ dataset, reimplemented UBAR with GPT-3, trained the user parser, and implemented dual-system pipeline. Proposed schema to integrate dual system into task-oriented dialog. Helped write the consistency checker.
- Weiqiu You: Helped formalizing the problem, ran the bAbI parser example. Wrote consistency checker rules and conducted error analysis for MultiWOZ dialog dual system.
- Liyang Zhou: Conducted literature review, trained system act parser and implemented/selected a subset of consistency rules that improves dialogue generation on the MultiWOZ dataset.
- Haoyu Wang: Conducted literature review, wrote consistency rules for dialogue generation and worked on experiments with the MultiWOZ dataset.
- All members collaborate on writing the final report.

Acknowledgement

We thank the course instructors Prof. Callison-Burch and Dr. Martin, teaching assistants Liam Dugan and Artemis Panagopoulou for their support and advice on this project.

References

Sebastian Bader, Artur S d'Avila Garcez, and P Hitzler. 2009. Extracting propositional rules from feed-forward neural networks by means of binary decision diagrams. In *Proceedings of the 5th International Workshop on Neural-Symbolic Learning and Reasoning, NeSy*, volume 9, pages 22–27. Citeseer.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. 2019. Bridging machine learning and logical reasoning by abductive learning. In *Advances in Neural Information Processing Systems*, pages 2811–2822.

Wang-Zhou Dai and Zhi-Hua Zhou. 2017. Combining logical abduction and statistical induction: Discovering written primitives with human knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.

Brian Falkenhainer, Kenneth D Forbus, and Dedre Gentner. 1989. The structure-mapping engine: Algorithm and examples. *Artificial intelligence*, 41(1):1–63.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Artur d'Avila Garcez and Luis C Lamb. 2020. Neurosymbolic ai: the 3rd wave. *arXiv preprint arXiv:2012.05876*.

Artur SD'Avila Garcez, Luis C Lamb, and Dov M Gabbay. 2008. *Neural-symbolic cognitive reasoning*. Springer Science & Business Media.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada. Association for Computational Linguistics.

Jiatao Gu, Kyunghyun Cho, and Victor O.K. Li. 2017. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978, Copenhagen, Denmark. Association for Computational Linguistics.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, Melbourne, Australia. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2018. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, pages 3749–3759.

Tomás Nekvinda and Ondrej Dusek. 2021. Shades of bleu, flavours of success: The case of multiwoz. *CoRR*, abs/2106.05555.

Toru Nishino, Shotaro Misawa, Ryuji Kano, Tomoki Taniguchi, Yasuhide Miura, and Tomoko Ohkuma. 2019. Keeping consistency of sentence generation and document classification with multi-task learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3195–3205, Hong Kong, China. Association for Computational Linguistics.

Maxwell Nye, Michael Tessler, Josh Tenenbaum, and Brenden M Lake. 2021. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. In *Advances in Neural Information Processing Systems*, volume 34, pages 25192–25204. Curran Associates, Inc.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Ron Sun. 1994. *Integrating rules and connectionism for robust commonsense reasoning*. John Wiley & Sons, Inc.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Qiang Yang, Yuqiang Chen, Gui-Rong Xue, Wenyuan Dai, and Yong Yu. 2009. Heterogeneous transfer learning for image clustering via the social web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th international joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 1–9. Association for Computational Linguistics.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2020. UBAR: towards fully end-to-end task-oriented dialog systems with GPT-2. *CoRR*, abs/2012.03539.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, ACL 2020*, pages 109–117.

Zhi-Hua Zhou. 2019. Abductive learning: towards bridging machine learning and logical reasoning. *Science China Information Sciences*, 62:1–3.