

Logical Reasoning for Task Oriented Dialogue Systems

Sajjad Beygi, Maryam Fazel-Zarandi*, Alessandra Cervone,
Prakash Krishnan, Siddhartha Reddy Jonnalagadda

Amazon Alexa AI

{beygi, fazelzar, cervon, prakaskr, sjjonnal}@amazon.com

Abstract

In recent years, large pretrained models have been used in dialogue systems to improve successful task completion rates. However, lack of reasoning capabilities of dialogue platforms make it difficult to provide relevant and fluent responses, unless the designers of a conversational experience spend a considerable amount of time implementing these capabilities in external rule based modules. In this work, we propose a novel method to fine-tune pretrained transformer models such as Roberta and T5, to reason over a set of facts in a given dialogue context. Our method includes a synthetic data generation mechanism which helps the model learn logical relations, such as comparison between list of numerical values, inverse relations (and negation), inclusion and exclusion for categorical attributes, and application of a combination of attributes over both numerical and categorical values, and spoken form for numerical values, without need for additional training data. We show that the transformer based model can perform logical reasoning to answer questions when the dialogue context contains all the required information, otherwise it is able to extract appropriate constraints to pass to downstream components (e.g. a knowledge base) when partial information is available. We observe that transformer based models such as UnifiedQA-T5 can be fine-tuned to perform logical reasoning (such as numerical and categorical attributes' comparison) over attributes seen at training time (e.g., accuracy of 90%+ for comparison of smaller than $k_{\max}=5$ values over heldout test dataset).

1 Introduction

Logical reasoning is an important aspect of human thinking and communication. Humans reason over beliefs, preferences, time, facts, and other contextual information to achieve complex tasks, derive meaning, and analyze emotions. Current

*Work done while at Amazon Alexa AI.

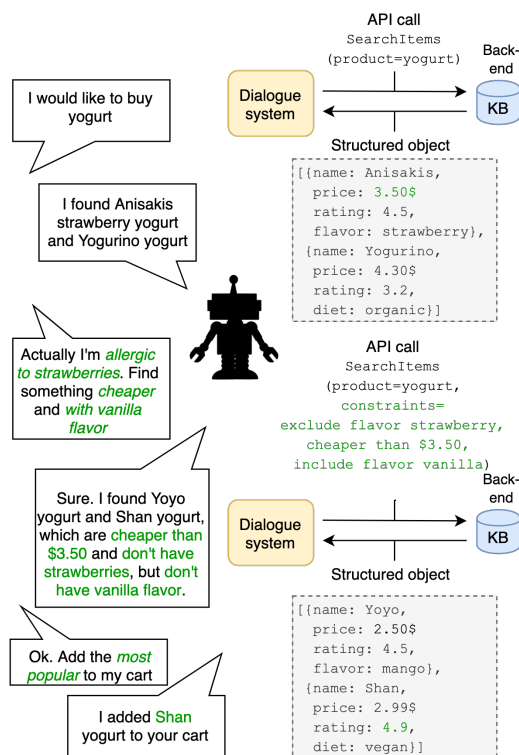


Figure 1: The dialogue system with reasoning ability.

task-oriented dialogue systems, however, only support very limited forms of logical reasoning. More specifically, although reasoning ability has been investigated as part of chatbots (Cui et al., 2020) and question-answering systems (Huang et al., 2019; Chen et al., 2020), in many task-oriented dialogue systems today, the reasoning is mainly focused on determining which slot values are still unknown to the system but are required and elicit them (Guo et al., 2017). However, in realistic task-oriented dialogues, logical reasoning is required to understand the user's request, ask questions that help address the user's task successfully and minimize asking irrelevant questions. The lack of robust, generalizable reasoning capabilities for dialogue systems, requires developers of the system to spend a considerable amount of time implementing these capabilities in external, rule-based and domain spe-

cific components. This leads to a poor user experience requiring users to often correct the system’s understanding, repeat themselves to ask the same question in different ways, restart the conversation when the system fails to recover from a ‘dead-end’, or even change their goal.

In this work, we propose to build on recent advances in research on logical reasoning and deep networks (e.g., [Dong et al. 2019](#); [Wang et al. 2019](#); [Xie et al. 2019](#); [Clark et al. 2020](#); [Arabshahi et al. 2020](#)) to bring reasoning capabilities to task-oriented dialogue systems. Our primary focus in this work is on mechanisms by which logical reasoning can be learned and used in conversational systems. In this direction, we propose a novel deep learning method to fine-tune pretrained models to reason over numerical and categorical attributes in the dialogue context and present an architecture for the integration of this model in task-oriented dialogue systems. Our objective is for the model to do logical reasoning to respond to queries from the dialogue context when it has all the required information available in the dialogue context without additional external logic (e.g., “Add the most popular to my cart” in [Figure 1](#)), extract constraints and inform downstream components when it only has partial context (e.g., “Actually I’m allergic to berries. Find something cheaper and with vanilla flavor” in [Figure 1](#), where cheaper means cheaper than what was shown so far), and not provide an answer when it does not have any relevant information and delegate to the dialogue policy to determine the next action.

We specifically choose to fine-tune transformers since these models operate on language directly, do not impose any structure on the reasoning process ([Clark et al., 2020](#)), and we can leverage the knowledge and diversity of language that the pretrained models have already learned. Furthermore, [Ding et al. \(2020\)](#) recently showed that these approaches can outperform neuro-symbolic methods. Our approach is similar to recent works on using transformers as soft reasoners ([Clark et al., 2020](#); [Talmor et al., 2020](#)). However, compared to these methods, we focus on use cases relevant to conversational systems and our model goes beyond predicting a true/false response to directly predicting the answer when the model has the information or extract constraints when it has partial information. In this direction, we report experimental results that show using our training method transformers

can learn to reason over numerical and categorical attributes in the dialogue context.

Note that although we use transformers for our experiments, our proposed method can be used to generate data and train any other seq2seq model for the same task and be integrated with any dialogue system in a similar manner. Furthermore, our proposed method is different from question-answering or machine reading comprehension in that we are not looking for an answer in a specific passage; rather, we want the model to reason over facts in the dialogue context to draw parallels and conclusions to inform decision making, similar to how humans reason over a multi-turn conversation.

2 Related Work

The approaches for integrating reasoning with deep networks can be categorized into the following.

Reasoning after Semantic Parsing These approaches convert utterances to a semantic representation and feed it to a set of rules or a formal reasoner for reasoning. For example, [Kamath and Das \(2018\)](#) provide examples where given a natural language utterance and context in the form of a relational database, the system first converts the natural language utterance to a SQL query that is then executed using standard SQL grammar to retrieve the answer. This is also similar in approach to how some teams that participated in the WikiSQL task ([Victor et al., 2017](#)) developed natural language interfaces for relational databases. However, writing and maintaining rules is not scalable especially as more complex types of reasoning become needed. The data annotation itself becomes hard to manage efficiently as more functionalities need to be supported. Furthermore, deep semantic parsing and reliably extracting attributes and relations and operating on multi-sentence input remains a challenge.

Satisfiability-based Approaches [Wang et al. \(2019\)](#) propose to integrate a differentiable maximum satisfiability solver into the loop of larger deep learning systems, and use this approach to successfully learn logical structures such as the rules of Sudoku. Previous works have shown that temporal reasoning can be modeled as a propositional satisfiability problem ([Pham et al., 2008](#)); however, generalizability to other types of reasoning needs further investigation. Although covering a rich class of problems, these approaches impose a structure on the reasoning problem ([Clark et al.,](#)

2020), i.e., learning of logical structure specifically as expressed by satisfiability problems.

Neuro-symbolic Approaches Neuro-symbolic systems are hybrid models that leverage neural networks and symbolic reasoning to integrate learning and reasoning. Besold et al. (2017) provide a survey of how symbolic approaches for reasoning are integrated with the machine learning approaches that bring in reasoning. More recently, Dong et al. (2019) propose Neural Logic Machines and apply them to different tasks such as relational reasoning and sorting. Arabshahi et al. (2020) propose an end-to-end differentiable solution that uses a Prolog proof trace to learn rule embeddings from data, and apply their approach to the task of uncovering commonsense presumptions. Similarly, Xie et al. (2019) generate a graph model to embed logic rules into the prediction. However, Ding et al. (2020) show that a fully-learned neural network with the right inductive biases can outperform neuro-symbolic approaches in the context of spatio-temporal interactions between objects.

Transformer Approaches Clark et al. (2020) and Talmor et al. (2020) propose to train transformers to reason over natural language sentences, bypassing a formal representation and show such reasoning over language is learnable. Ding et al. (2020) apply a similar technique to visual question answering and show that their approach outperforms neuro-symbolic approaches. Han et al. (2020) use a similar approach to fine-tune a language model for event temporal reasoning. Our approach builds on top of these works in that we integrate reasoning into task-oriented dialogues and go beyond predicting a true/false response for an input and instead directly predict the answer when the model has the information or extract constraints when it has partial information.

Knowledge Grounding in Dialogue Similar to how Victor et al. (2017) retrieve knowledge from Wikipedia, approaches such as (Ghazvininejad et al., 2018; Neelakantan et al., 2019; Gopalakrishnan et al., 2019) retrieve knowledge from a database to be incorporated into dialogue. These approaches extend the seq2seq approach to condition on the facts present in the knowledge bases. While this is a promising architecture, such approaches are good for applications such as knowledge-grounded open domain chat but not for supporting reasoning in task-oriented dialogues.

Other Approaches There are also other techniques in the literature such as integrating rules defined in first-order logic with knowledge distillation (Hu et al., 2016) that are outside the above categories. There have also been efforts such as CLUTRR (Sinha et al., 2019), bAbI dataset (Weston et al., 2015), Single Rule Test (Richardson et al., 2020), QuaRTz dataset (Tafjord et al., 2019), HotpotQA (Yang et al., 2018), and ROPES (Reasoning over Paragraph Effects in Situations) (Lin et al., 2019), that focus on creating benchmarks for reasoning that measure how well existing systems perform on generalized reasoning.

3 Problem Statement

Task-oriented dialogue systems use a natural language understanding component to extract semantic meaning from the user utterance, and elicit constraints from users to understand their goals in order to provide information, perform a task or provide options and alternatives for users to choose from, retrieved from external knowledge sources (e.g. through API calls). As such, we focus on reasoning over tasks and recommended items in the dialogue which are typically characterized by different attributes, for example, movie names and show-times for a ticket booking scenario. These systems rely on such representations to answer user queries such as “*At what time is Vertigo playing?*” by performing API calls (e.g. `searchTime(movie=Vertigo)`) which return the required information in a structured form (Movie=Vertigo, Times=[12:30-2:30 PM, 3-5 PM], Theater=Cineplex). The required information is then returned to the user in natural language (e.g. *Vertigo is playing today from 12.30 to 2.30 PM and from 3 to 5 PM.*). However, in most currently available task-oriented dialogue systems if the user said next “Book me the earliest one,” although this information is already available to the system from the previous API call, given the lack of reasoning abilities the system would either not support such queries, or it would have to make an additional independent API call (e.g., `searchEarliestTime(movie=Vertigo)` or `searchTime(movie=Vertigo, modifier=earliest)`), creating redundant latency in the response and requiring the developer of the system to add APIs/rules to handle these use cases.

Given the above description, our objective is to train a model to learn how to reason over the

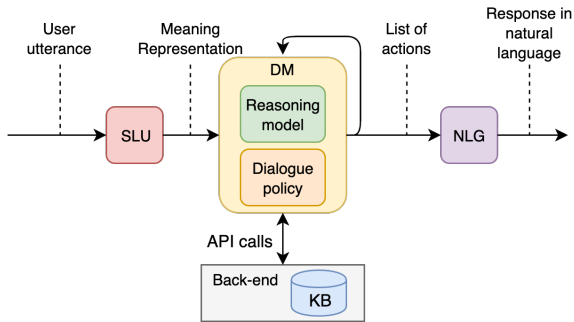


Figure 2: The reasoning model can be easily integrated in task-oriented dialogue architecture, as a component of the Dialogue Manager, i.e., the module in charge of predicting the next system action.

information provided in the context. We assume the following scenarios for each user utterance:

1. Reasoning-required, answer available in the context: The case where the user utterance requires reasoning and it is possible to infer the answer to the user query from the information returned by the previous API calls (e.g., “Give me the earliest one”). Rather than extracting mentions and querying the knowledge base again, in this case the model directly outputs the predicted next system action along with its arguments.

2. Reasoning-required, answer not available in the context: The case where the user utterance requires reasoning, but it is not possible to infer the answer to the user query from the information returned by the previous API calls (e.g., “Show me cheaper options”). In this case the model extracts constraints from the user utterance to be passed to the back-end API.

3. Reasoning-not-required: The case where the user utterance does not require reasoning (e.g., “Please repeat”).

In order to support these scenarios, the model needs to learn to 1) compare between different items based on numerical and categorical attributes, 2) compare across a list of numerical values to identify the minimum/maximum value among alternatives, 3) be able to formulate constraints when it is not possible to infer the answer to the user query given the dialogue context but partial inference can be made, and 4) respond no answer when no reasoning is required for answering the user’s request.

Figure 2 shows the overall architecture of a dialogue system with the reasoning model. The new model is part of the dialogue manager which predicts the next system action, along side a domain specific dialogue policy. The dialogue policy can predict API calls for retrieving information from a

back-end Knowledge Base (KB) or can predict a list of natural language generation (NLG) actions for communicating information to the user (requesting constraints, informing available options, etc.). The reasoning model is added as a modular component that runs along-side the dialogue policy model. Although it would be possible to combine the two models, e.g, by extending the reasoning model to also predict domain specific APIs and actions, we believe that this modular architecture allows the reuse of a trained reasoning model across different domains and tasks.

4 Method

In this work we propose to fine-tune transformers to learn logical reasoning over dialogue context in the form of natural language sentences, bypassing a formal representation and showing such reasoning over language is learnable.

4.1 Data Generation

We describe a general methodology for automatically creating a dataset for logical reasoning in task-oriented dialogue systems. Each example in the dataset is a triple (user-query, context, answer), where the user-query refers to the last user utterance, the context refers to the dialogue context and information returned by API calls to the back-end system (see an example in Figure 1), and the answer refers to the next action to be taken by the dialogue system. The user-query and the context constitute the information given as input to the model, while the answer represents the output.

In order to simulate the context, the objects returned by API calls to the back-end system, we assume an available knowledge base (KB). We further assume that the KB will have different items, identified by an item-name (e.g., *Yogurt Anisakis*), an item-type (e.g., *yogurt*), and a series of attributes, each with an attribute key and value (e.g., *price*: \$3.40). For generalizability, we do not assume that all item types have the same attributes, nor that all items of the same type have the same attributes.

The data generation procedure consists of four main steps:

1. Items sampling: In order to construct input-output pairs for training, we first randomly select k items, where $0 \leq k \leq k_{max}$, with the same item-type to create the input context c . While in this work we compare items of the same item-type, this is not a strict requirement of data generation.

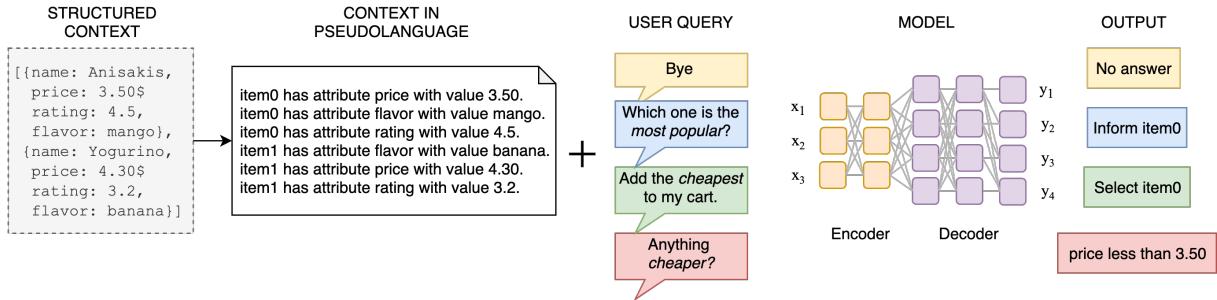


Figure 3: Task structure for the generative model.

The motivation behind this choice is given by a typical scenario of a task-oriented dialogue system where a user might search for a specific object (movie times of Vertigo) and the system would subsequently present different options for that object (“Vertigo is playing today from 12:30 to 2:30 PM and from 3 to 5 PM.”).

2. Context conversion to pseudo-language: Once a set of items has been sampled, we transform the structured information (list of triplets) associated to each item into pseudo-language by using a template-based approach, as in Figure 3. Our templates are constructed in a domain-agnostic way, so that they would be directly applicable to other scenarios. We define two main types of statements in pseudo-language, each one associated to a specific template (see first two rows in Table 1). The `IsA` template is used to define the type of an item, while the `HasAttribute` relation is used for triplets expressing the value of a given attribute for the specified item. We note that other templates for the context statements could easily be created to accommodate different scenarios. Finally, we concatenate all the generated statements, after randomizing their order for improving robustness, to form the final input context.

3. Query generation: In this step we generate a set of user queries q suitable for the given context using templates, thus generating several number of different input pairs (c, q_i) where i is an index over possible queries related to the context c . Note that templates for the queries are manually created for each attribute, but they are all agnostic from

the domain of the task-oriented dialogue system. Examples of user queries are shown in Table 1. As it can be seen, each template for the user query was associated to the expected output action predicted by the system and the particular reasoning ability involved (e.g., `Inform`). We also consider more complex cases such as *negation*, e.g., “I don’t want anything vegan,” and *conjunction*, e.g., “Which is the cheapest one and doesn’t have strawberry?”. Additionally, each template is associated with several different surface form variations to add robustness to the model. Each generated user query is then prepended to the context c . An additional optional post-processing step consists of converting all the numerical values in the user queries from written to spoken format (e.g. “\$3.50” is converted to “three dollars fifty”). This step might be required in the context of a spoken dialogue system scenario, which takes directly as input the output of the Automatic Speech Recognition model.

4. Output creation: In the final step, for each generated input, we automatically create the output by combining the information from each template in regards to the action type to take and calculating the correct answer from the context, e.g., *Yogurt Anisakis is the cheapest*. The output space consists of four main outcomes, as shown in Table 2, depending on whether reasoning is required to respond to the user utterance, and whether the answer is retrievable from the available context. We use the special token `NoAnswer` for user queries that do not require reasoning. When the answer is retrievable from the context and reasoning is re-

Type	Name	Template	Example
Context Statement	IsA	[subject] is [object].	Yogurt Anisakis is a yogurt.
Context Statement	HasAttribute	[subject] has attribute [attribute] with value [value].	Yogurt Anisakis has attribute price with value 3.55.
User Query	Inform	I want something [predicate] [value].	I want something cheaper than \$5.
User Query	Inform_TrueFalse	Which one is [predicate]?	Which one is the cheapest?
User Query	Select	Is [subject] [predicate]?	Is Yogurt Anisakis the cheapest?
User Query	Select	Select [predicate].	Select the cheapest.

Table 1: Examples of templates for context statements (in pseudo-language) and user queries (in natural language)

Reasoning Required	Answer in Context	Action Type	Example	Output
Yes	Yes	Inform	Is the first one cheaper than the second one?	inform <true/false>
Yes	Yes	Inform	Which one is the cheapest?	inform <item_name>
Yes	Yes	Select	Add the cheapest to my cart.	select <item_name>
Yes	No	Constraint	Give me something cheaper	<relation> <attribute> <value>
No	—	No Answer	Find yogurt.	NoAnswer

Table 2: Output space. In cases where there are multiple answers/constraints, they are concatenated with *and*.

User Utterance	Constraint
Give me something vegan.	include diet vegan
I don't want mango.	exclude flavor mango
It should cost \$1.50.	equal price 1.50
I want it cheaper than \$2	less-than price 2
Anything more popular?	more-than rating 4.5

Table 3: Examples of constraints representation, given as context the one in Figure 2.

quired, we further distinguish between two main cases: *inform*, when the user is simply seeking information (e.g., “Which one is the cheapest?”), thus performing an Information-Transfer type of Dialogue Act (see Bunt et al. (2010)), and *select*, when the user is requesting the system to perform a specific action (e.g., “Add the cheapest to my cart.”), an Action-Discussion Dialogue Act. For the *inform* action, we also distinguish in the output space between True/False questions and open-answer questions.

In the case of constraint extraction answers, i.e., when the user utterance requires reasoning but the context has partial information, the output consists of the list of constraints extracted from the user query and concatenated with *and*, as shown in Table 3. The constraints extracted from the user query depend on the context, not only in terms of action to take (whether to provide an answer directly or to extract constraints), but also in terms of constraints generation. In the last row of Table 3, for user query (“..more popular?”) the reasoning model relies on the context by looking at the ratings of the available products to extract the appropriate rating constraint (e.g., *more-than rating 4.5*).

4.2 Training Procedure

In order to teach the model rules such as inverse relations and transitivity by example, we investigate the use of appending to the context clues that describe the relations of one or more items. These clues are appended to the final input context during training, but not at inference time. We consider two types of clues: 1) *Comparative clue* describes a comparison of two items in the context along

a specific attribute. The template for this clue is: [subject] is [predicate] [object], where *predicate* refers to the quality regarding which the items are being judged (e.g., “cheaper than”, “pricier than”, “less than”, “equal to”). 2) *Superlative clue* describes an object at the upper/lowest range of a specific attribute. The template for this clue is: [subject] is [predicate] with value [value]. Using the base data generation and clue generation, we are able to construct three types of training scenarios, as follows:

Case I - Clueless context: This scenario uses the base context encompassing the information about the items’ different attributes. This is also the scenario we expect at inference time.

Case II - Comparative clues: In this scenario, we sort the items in the base context according to the values of their attributes and append to the base context the comparative relation between pairs of items that are neighbors. The direction of comparison selected is random (e.g. “A is larger than B” or “B is smaller than A”) and independent from the user query. This scenario is designed to assess the ability of the model to learn *inverse* relations, since in some queries users will ask for a relation in the opposite direction in regards to the comparative clue in the context (e.g., user asks “Is the second one cheaper than the first one?” while in the context we have “A is pricier than B”), so that the model could learn that these two statements are equivalent. When we have more than two items in context, we can also assess the ability of the model to learn *transitivity*, as we might have cases where the user asks “Is the first one pricier than the third one?” and in the context we have “A is pricier than B” and “B is pricier than C”.

Case III - Superlative clues: In this scenario, besides comparative clues, we also add superlative clues to the context to give hints to the model about which item in the context has the extreme value of the attributes (e.g. “A is the cheapest”).

We pick the number of items in each context randomly from 0 to k_{max} , so that the model can be robust in its prediction for different number of

Rating	Price	Diet	Flavor
Bounded	Unbounded	10	10K
Numeric	Numeric		

Table 4: Attributes and their catalogs size.

items in the context. We also consider an additional training procedure, which we refer to as Case IV, where we randomly select one of Case I, Case II, or Case III as our context. The random selection of context helps the model to experience all three different cases and by cross learning between different cases, it learns to apply the inverse and transitivity rules for examples with Case I context to draw the right conclusion.

5 Experiments

We showcase our proposed methodology in the context of a dialogue system for a shopping assistant (see Appendix A for an example interaction). We use an ontology for data generation which consists of `item-type` (e.g. yogurt) and `item-name` (“Greek yogurt Anisakis”) and each item is characterized by two numerical attributes `price` and `rating`, and two categorical attributes `diet` and `flavor`. This choice of attributes can help us explore and assess the model’s performance based on attribute’s characteristics. Table 4 summarizes the size of the catalog or range of values for each attribute.

We consider two settings for assessing the logical reasoning capability of transformer models. In the first setting, we fine-tune RoBERTa-base (Liu et al., 2019) with a training dataset generated for reasoning using only numerical attributes. In this setting, we only focus on True/False prediction for each query q given the facts provided in the context c . The objective of this experiment is to understand whether transformer models can learn to reason over numerical attributes. In the second setting, we use a T5 model (Raffel et al., 2019) fine-tuned for the UnifiedQA data (Khashabi et al., 2020), to predict a sequence similar to one given in Table 2. In both cases, we use disjoint catalogs to generate examples for train/dev/test datasets to avoid over-fitting to attribute values.

5.1 True/False Queries

We consider True/False reasoning over attributes such as assessing a conclusion about the comparison of two values of an attribute, or finding minimum or maximum value among list of values of

Train/Test	I/I	II/II	III/III
2 items	90%	97%	97%
3 items	88%	95%	95%
5 items	77%	91%	93%

Table 5: Roberta-Base model performance for T/F Reasoning over Price and Rating.

Train → Test ↓	Case II (5 items)	Case III (5 items)
Case I, (2 items)	75%	76%
Case I, (3 items)	70%	71%
Case I, (5 items)	67%	69%

Table 6: Train on Case II or Case III with 5 items in all the contexts and test on Case I with 2, 3, or 5 items.

an attribute for several items. Example queries include “is the second item the cheapest one” and “is the first one cheaper than the fourth one”. We fine-tune RoBERTa to predict True/False for each (q, c) by adding a classification layer on top of the RoBERTa encoder model to perform binary classification. The training hyper-parameters for fine-tuning this model are provided in Appendix B. For these experiments, we generate 120K samples for train, 5K for dev, and 25K for test set.

Clueless Training: In this case, we only add `IsA` and `HasAttribute` relations and don’t include any clue in the context c in the training data (i.e., Case I). For each generated context, the data generation process attaches all possible forms of queries and the potential true/false label and adds them to training samples. For evaluation, we generate the test samples in a similar fashion. Table 5 summarizes the model performance for predicting the right label for each query given the context with $k \in 2, 3, 5$ number of items in the context. We can see that by increasing the context size (or number of returning items from back-end) the model performance decreases. To understand how well a model with larger k with comparative or superlative clues can generalize to fewer number of items in context, Table 6 shows the performance of a model trained with context size of 5 items using Case II or Case III samples and tested on samples generated by Case I and with $k \in 2, 3, 5$ items. We observe that the model does not generalize to different context sizes if we fix the number of items in the context during model training.

Clue-Aware Training: To resolve the issues in clueless training, we add comparative and superlative clues randomly to each context during the

Train/Test	IV/I	IV/II	IV/III
up-to 5 items	98.70%	99.70%	99.70%

Table 7: Training with CaseIV: Roberta model performance for T/F reasoning over numerical attributes.

training such that the model can learn the inverse and transitivity rules; and also we add random number of items to each individual context (up to k_{max}). Note that we do not add clues to the context during evaluation/inference. Results in Table 7 show the accuracy performance of models trained using samples generated by Case IV and tested on Case I (clue-less), Case II (only comparative clues), and Case III (both comparative and superlative clues) samples. From the results, we observed that adding clues during model training helps the model to achieve better performance.

5.2 Beyond True/False Queries

For this set of experiments, we pick the T5 transformer model which can enable us to perform text-to-text prediction. Similar to (Khashabi et al., 2020), we remove the task prefix that has been used in the original T5 models, since we will use this model only for a single reasoning task within our defined framework. To take advantage of transfer learning from other publicly available question-answering datasets, we start our fine-tuning from the pretrained Unified-QA-T5 small model. We generate 100K samples for training dataset, 5K for dev, and 20K examples for each test set. In our test set we make sure that for each element in Table 8, we have at least 5K examples. Samples are generated as described in Section 4.1. The training hyper-parameters for fine-tuning this model are provided in Appendix B.

In Table 8, we summarize the performance of the fine-tuned model for different scenarios, reporting the results separately for pair of (q, c) such that q can have one (e.g., ‘‘Give me something organic’’) or two attributes (e.g., ‘‘Something cheaper than \$100 but not vegan’’) about user-preferences. We use the exact-match (EM) accuracy metric to evaluate model performance. We can observe that the model can achieve an EM accuracy of over 90% across all the scenarios. Furthermore, we see that when increasing the number of items in the reasoning context, predicting the correct Inform/Select or Extract output form becomes harder with more attributes in the user query. Evaluating the model performance on all examples (about 8K samples)

# of Attr.s	k_m	Inform/Select	Extract
1	0	–	99.5±0.02%
	1	98.6±0.05%	99.2±0.03%
	2	97.3±0.05%	98.5±0.05%
	3	97.0±0.05%	98.0±0.03%
	4	96.0±0.10%	98.0±0.05%
2	5	95.5±0.09%	96.0±0.06%
	0	–	98.6±0.03%
	1	98.5±0.05%	97.8±0.02%
	2	95.0±0.08%	96.7±0.01%
	3	94.5±0.05%	96.3±0.03%
	4	91.5±0.09%	95.0±0.03%
	5	90.0±0.11%	93.5±0.06%

Table 8: EM accuracy for test sets with different number of attributes, context size, and reasoning task.

from our test set that include spoken form of numerical values in q (e.g., ‘‘Give me something cheaper than five dollars’’), we observe 95% EM accuracy, showing the ability of the model to compare written form and spoken form versions of numbers. We should note that the accuracy of the model for predicting the cases with no reasoning (e.g., ‘‘Checkout please’’) is important because it makes the integration with the overall dialogue system simpler where the model can delegate to the domain specific dialogue policy. In our experiments, we observe an accuracy of 100% on these cases; however, this value can vary by increasing the size of out-of-domain space/vocabulary.

6 Conclusions

In this paper, we proposed an architecture for the integration of a reasoning model in task-oriented dialogue systems. We formulated the problem as a sequence prediction problem given a user query and context, and presented an approach for generating data and fine-tuning generative models to reason over a set of facts in the dialogue context. We demonstrated our approach for a shopping assistant and reported experimental results for different formulations of the problem. We showed that these models can learn to do logical reasoning to 1) answer questions from the dialogue context when all the information is available, 2) extract constraints when partial information is available, and 3) delegate to the dialogue policy when no reasoning is required.

For future work, we plan to investigate the application of our method to other reasoning tasks (e.g., temporal and spatial reasoning). We also plan to experiment with additional models to compare performances with the ones presented in this work,

to further investigate the complexity of the task at hand. Moreover, we would like to test our models on more challenging and realistic testsets, for example by adding noise in the current synthetic data or by performing a data collection with human annotators. Furthermore, we plan to explore how logical reasoning can be used to disambiguate with the user when multiple conclusions can be made.

References

- Forough Arabshahi, Jennifer Lee, Mikayla Gawarecki, Kathryn Mazaitis, Amos Azaria, and Tom Mitchell. 2020. Conversational neuro-symbolic commonsense reasoning. *arXiv preprint arXiv:2006.10022*.
- Tarek R Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Daniel Lowd, Priscila Machado Vieira Lima, et al. 2017. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv preprint arXiv:1711.03902*.
- Harry Bunt, Jan Alexandersson, Jean Carletta, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Kiyong Lee, Volha Petukhova, Andrei Popescu-Belis, Laurent Romary, Claudia Soria, and David Traum. 2010. [Towards an ISO standard for dialogue act annotation](#). In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).
- Xiuying Chen, Zhi Cui, Jiayi Zhang, Chen Wei, Jianwei Cui, Bin Wang, Dongyan Zhao, and Rui Yan. 2020. [Reasoning in dialog: Improving response generation by context reading comprehension](#). *CoRR*, abs/2012.07410.
- Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*.
- Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. 2020. [Mutual: A dataset for multi-turn dialogue reasoning](#). *CoRR*, abs/2004.04494.
- David Ding, Felix Hill, Adam Santoro, and Matt Botvinick. 2020. Object-based attention for spatio-temporal reasoning: Outperforming neuro-symbolic models with flexible distributed architectures. *arXiv preprint arXiv:2012.08508*.
- Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. 2019. Neural logic machines. *arXiv preprint arXiv:1904.11694*.
- Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. [A knowledge-grounded neural conversation model](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).
- Karthik Gopalakrishnan, Behnam Hedayatnia, Qiang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tür. 2019. [Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations](#). In *Proc. Interspeech 2019*, pages 1891–1895.
- Xiaoxiao Guo, Tim Klinger, Clemens Rosenbaum, Joseph P Bigus, Murray Campbell, Ban Kawas, Kartik Talamadupula, Gerry Tesauro, and Satinder Singh. 2017. Learning to query, reason, and answer questions on ambiguous texts.
- Rujun Han, Xiang Ren, and Nanyun Peng. 2020. Deer: A data efficient language model for event temporal reasoning. *arXiv preprint arXiv:2012.15283*.
- Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*.
- Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. [Cosmos QA: Machine reading comprehension with contextual commonsense reasoning](#). 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 2391–2401, Hong Kong, China. Association for Computational Linguistics.
- Aishwarya Kamath and Rajarshi Das. 2018. A survey on semantic parsing. *arXiv preprint arXiv:1812.00978*.
- Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. [Unifiedqa: Crossing format boundaries with a single QA system](#). *CoRR*, abs/2005.00700.
- Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. *arXiv preprint arXiv:1908.05852*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Arvind Neelakantan, Semih Yavuz, Sharan Narang, Vishaal Prasad, Ben Goodrich, Daniel Duckworth, Chinnadhurai Sankar, and Xifeng Yan. 2019. [Neural assistant: Joint action prediction, response generation, and latent knowledge reasoning](#). *CoRR*, abs/1910.14613.
- Duc Nghia Pham, John Thornton, and Abdul Sattar. 2008. Modelling and solving temporal reasoning as propositional satisfiability. *Artificial Intelligence*, 172(15):1752–1782.

- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *CoRR*, abs/1910.10683.
- Kyle Richardson, Hai Hu, Lawrence Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8713–8721.
- Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. 2019. Clutrr: A diagnostic benchmark for inductive reasoning from text. *arXiv preprint arXiv:1908.06177*.
- Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019. Quartz: An open-domain dataset of qualitative relationship questions. *arXiv preprint arXiv:1909.03553*.
- Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. 2020. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. *arXiv preprint arXiv:2006.06609*.
- Zhong Victor, Xiong Caiming, and Socher Richard. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.
- Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. 2019. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554. PMLR.
- 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*.
- Yaqi Xie, Ziwei Xu, Mohan S Kankanhalli, Kuldeep S Meel, and Harold Soh. 2019. Embedding symbolic knowledge into deep networks. *arXiv preprint arXiv:1909.01161*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

A Example Interaction with the Shopping Assistant

The following is an example interaction with the shopping assistant with our reasoning model integrated with the dialogue policy.

Speaker	Utterance
User	I want to buy apples
Agent	The top 3 relevant products are: Organic Honeycrisp Apple, Organic Gala Apple, and Organic Pink Lady Apple. Which one would you like to add to the cart?
User	Which one is the cheapest?
Agent	The second one, Organic Gala Apple with a price of \$2.49.
User	Which one is the most expensive?
Agent	The first one, Organic Honeycrisp Apple with a price of \$3.99
User	Is the first one cheaper than the second one?
Agent	No. The first one, Organic Honeycrisp Apple, costs \$3.99 but the second one, Organic Gala Apple, costs \$2.49.
User	Is the second one cheaper?
Agent	Yes, that is correct. The second one, Organic Gala Apple, costs \$2.49.
User	Give me something cheaper
Agent	Top relevant product is Fuji Apple. Would you like to add it to the cart?
User	Does it cost less than two dollars?
Agent	Yes, that is correct. Fuji Apple costs \$1.89.
User	Add it to my cart
Agent	I have added Fuji Apple to your cart. What else would you like to buy?
User	Buy grapes
Agent	The top 2 relevant products are: Red Seedless Grapes and Conventional Cut Grapes. Which one would you like to add to the cart?
User	Is the second one more popular?
Agent	No. The second one, Conventional Cut Grapes has a rating of 4.3 but the first one, Red Seedless Grapes has a rating of 4.5.
User	Add the highest rated one to my cart
Agent	I have added Red Seedless Grapes to your cart. What else would you like to buy?
User	Find cheese
Agent	The top 2 relevant products are: Zola and Muller. Which one would you like to add to the cart?
User	Anything vegan and cheaper than five dollars?
Agent	I couldn't find anything vegan with price less than five dollars. But I found Vegan Artisan cheese with price of \$7.99. Would you like to add it to the cart?
User	Sure
Agent	I have added Vegan Artisan cheese to your cart. What else would you like to buy?
User	What's in my cart?
Agent	You have 3 items in the cart with total amount of \$12.87. Do you want to hear detailed description?
User	Checkout
Agent	I have placed your order.

B Training Hyper-parameters

In this section, we provide the parameters that are used to fine-tune the transformer models in this

work. The following table summarizes the key parameters that are used during the fine-tuning of Roberta-base and UnifiedQA-T5-small pretrained models. For the optimizer, we use AdamW (Loshchilov and Hutter, 2017).

Model	Batch Size	Epoch	Optimizer	Learning Rate	Weight Decay
Roberta-Base	192	15	AdamW	10^{-5}	0.1
UnifiedQA-T5-small	1024	20	AdamW	10^{-5}	0.0

C Discussion

C.1 Generalization to non-synthesis datasets

One of the directions that currently we are working on is to create realistic (human based) conversations with logical reasoning use cases during the interactions with the dialog systems. This type of dataset can help us to evaluate the proposed idea with higher degree of confidence. Since no matter how much one spends time on generating synthetic datasets, there will always be some uncontrolled structures introduced by design of data simulation mechanisms that can corrupt the fair evaluation of deep neural network models and their learning process. However, we believe the True/False scenarios in our current study are less prone to this type of issues and are quite helpful in understating of reasoning capabilities such as negation, numerical comparison, or inclusion/exclusion of categorical values of our proposed algorithm, since model needs to learn the reasoning procedure. In other words, the only way to come up with the right prediction by model is to apply the underlying reasoning procedure to formulate the output True/False results. We will consider: a) better algorithms for generating training data, and b) more realistic general purpose possibly human in the loop training data to make the data generation more general and less domain specific, for future exploration.

C.2 Error Analysis

During our evaluation, we observed that the Transformer models (such as Roberta and T5) performance degrades when the length of the reasoning context increases, i.e., the number of items in the context for reasoning are longer. Also based on the results on Table 8, we see that increasing the number of items in reasoning context leads to performance degradation. Another issue with Transformer models or in general LM models is during the output generation process beyond the

True/False scenario. When the size of the output sequence length increases, e.g., there are several items that all satisfy the user-query. The prediction misses some of the items in the response after the length of the output sequence (number of predicted tokens/words) meets some threshold. This issue is related to both long sequence generation of LM models and also reasoning ability when the multiple items match the user-query’s criteria which mostly occurs when the number of items in context are larger.

C.3 Generalization to unseen attribute with common values

One of the aspect that we like to understand is the scalability/generalization of the proposed trained reasoning model to unseen attributes during the test time. There are two possibility for a new attribute: (1) doesn’t shares values and keywords that user may use to describe the attribute compared to the attributes that are used during the training process e.g., `color` attribute for experiment in Section 5 ¹. (2) shares same values but keywords that user may use to describe the attribute doesn’t overlap with any of the ones used during the training process, e.g., `calorie` ². It would be very challenging to teach model in a few-shot manner to learn about attributes from bucket (1). However, based on our initial experiments we have seen that model can easily generalize to the attributes from bucket (2), by fine-tuning to small number of examples in a few-shot manner. For example, we fine-tuned the model which only trained for `diet`, `flavor`, `price`, and `rating` attributes and fine-tuned using only 100 new reasoning context examples which had `calorie` attribute as well. Table 9 summarize the model performance before and after fine-tuning. The test set used for this analysis only has user-query about calories and includes 3K examples about Calorie attribute.

Model	EM accuracy
Before fine-tuning	33%
After fine-tuning	80%

Table 9: Model EM accuracy performance before/after fine-tuning to new attribute `calorie`.

¹For query about the `color` user may use keywords such as: [darker, lighter, warmer, red, blue, ..., etc.] one, and attribute values are red, blue, dark blue, ..., etc. which doesn’t overlap with none of the attributes that we have already in our training dataset, i.e., `diet`, `flavor`, `price`, and `rating`

²For query about the `calories` user may use keywords such as: [healthier, higher calories, more energetic..., etc.] one, and attribute values are numeric value that are shared possibly with `price` and `rating` [considering we have done unit normalization for attributes]