Controlled Data Augmentation for Training Task-Oriented Dialog Systems with Low Resource Data

Sebastian Steindl and **Ulrich Schäfer**
Ostbayerische Technische Hochschule
Amberg-Weiden, Germany
{s.steindl,u.schaefer}@oth-aw.de

Bernd Ludwig
University Regensburg,
Germany
bernd.ludwig@ur.de

Abstract

Modern dialog systems rely on Deep Learning to train transformer-based model architectures. These notoriously rely on large amounts of training data. However, the collection of conversational data is often a tedious and costly process. This is especially true for Task-Oriented Dialogs, where the system ought to help the user achieve specific tasks, such as making reservations. We investigate a controlled strategy for dialog synthesis. Our method generates utterances based on dialog annotations in a sequence-to-sequence manner. Besides exploring the viability of the approach itself, we also explore the effect of constrained beam search on the generation capabilities. Moreover, we analyze the effectiveness of the proposed method as a data augmentation by studying the impact the synthetic dialogs have on training dialog systems. We perform the experiments in multiple settings, simulating various amounts of ground-truth data. Our work shows that a controlled generation approach is a viable method to synthesize Task-Oriented Dialogs, that can in turn be used to train dialog systems. We were able to improve this process by utilizing constrained beam search.

1 Introduction

The current success of Large Language Models (LLMs) is the result of multiple factors. One of them is the availability of high-quality data in large amounts. Specifically, the prospering adoption of these LLMs as chatbots, as pioneered by ChatGPT (OpenAI, 2022), was made possible due to the usage of human feedback during training in the RLHF framework (Christiano et al., 2017).

The category of Task-Oriented Dialog (TOD) systems describes a specific kind of chatbot that aims to help users achieve tasks, such as booking hotels or making reservations, using external services. It can therefore be seen as a language-based interface to these services.

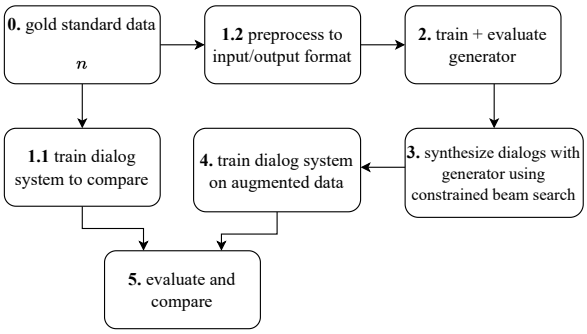

Figure 1: The general setup of the study. We use n gold standard dialogs to train a generator, which synthesizes dialogs that are in turn used to train a dialog system (steps 1.2 to 4.). To evaluate the improvement (step 5), we compare the performance of the dialog system to the baseline, where we only use the n dialogs for training (step 1.1).

In the recent past, different model architectures have been proposed to solve this problem. To evaluate them, the original MultiWOZ (Budzianowski et al., 2018) dataset and its successors are widely used as benchmarks. It contains more than ten thousand dialogs spanning multiple domains (e.g., attraction, hotel and restaurant), paving the way for the adoption of large end-to-end models.

While this benchmark allows for useful comparisons of different architectures and approaches, we deem this scenario of having multiple thousand dialogs to be infeasible for most real-life TOD use cases, where usually not nearly as many dialogs are available. This is due to the fact that the types of dialog needed to train a TOD system are different from those used to train, e.g., a social chatbot, since the TOD conversations need to include the usage of the external systems. Therefore, it is typically not only necessary to have two users (while using the Wizard-of-Oz technique; Kelley, 1984), but to have at least one of them interact with the external service to provide the necessary information to achieve the task.

Proceedings of the 2nd Workshop in Pattern-based Approaches to NLP in the Age of Deep Learning , pages 92–102

We therefore investigate the synthesis of suitable utterances from given annotations to train TOD systems. We call this method "Controlled Generation for Training" (CG4T). For this, we simulate lower resource scenarios based on the MultiWOZ dataset, allowing us to both evaluate the models on this common benchmark and also to test the method in different, more realistic settings.

With the objective of ensuring that the generated utterances contain the entities defined in the annotation, we additionally investigate the effect of constrained beam search during the generation phase. To sum up, in our work, we aim to answer four research questions (RQ):

- RQ 1: Can a pretrained sequence-to-sequence model be fine-tuned to generate synthetic dialogs based on annotations?
- RQ 2: Can constraining the beam search improve the decoding in comparison to a normal beam search?
- RQ 3: Can the synthetic data be used to train and improve the performance of a TOD system?
- RQ 4: What is the relation between the amount of real data and the usefulness of synthetic data generated this way?

2 Background and Related Work

Controlled Generation. The main goal of the approach presented in this paper is to generate text that fulfills certain requirements, i.e., fitting the annotations it is based on. The general problem of steering generation has been addressed in the young research field of Controllable Text Generation (CTG). The root motivation behind this is to have a Natural Language Generation (NLG) system be controllable by human-defined parameters, which becomes especially important due to their black-box character (Zhang et al., 2022). Some of the applications of CTG include NLG that adheres to a specific topic, emotion, formulating text from structured data (e.g., tables), and data augmentation (Zhang et al., 2022). For example, Keskar et al. (2019) train a model with control codes that allow to influence the generation by prepending them to the prompt.

In similar spirit is the research in so called model alignment. The alignment of a model describes its fine-tuning with the goal of increasing the probability of desired outputs, i.e., those aligning with the intended use and human preferences, and decreasing the chance of undesired outputs. This has recently been achieved both with the RLHF framework (Ouyang et al., 2022) and standard supervised training (Zhou et al., 2023).

TOD Systems and the MultiWOZ Dataset. Since the publication of the original MultiWOZ (Budzianowski et al., 2018) dataset, there have been multiple updated versions, mostly to fix label noise (Eric et al., 2020; Zhang et al., 2022; Ye et al., 2022). The setting introduced by the dataset allows for two types of evaluation: Dialog State Tracking (DST) and Response Evaluation. Since recent publications manage to get near-perfect results for the Response Evaluation as measured by the inform and success metric (Cheng et al., 2022), we decide to focus on the DST task.

There has been varied research (e.g., Bang et al., 2023; Zhao et al., 2022; He et al., 2022) on specific model architectures, using distinct approaches to solve the DST task. Kim et al. (2020) propose SOM-DST, which uses an explicit state memory and predicts the operation to perform for each slot at every turn, e.g., carryover or update. On the other hand, STAR (Ye et al., 2021) tries to leverage the correlations between the slots with a slot self-attention mechanism.

Constrained Beam Search. When generating outputs with an NLG model, the goal of exhaustively finding the sequence with the highest probability is infeasible in most cases. Since for a sequence of length m over a vocabulary with length v, the computational cost would be $\mathcal{O}(v^m)$. Accordingly, beam search is commonly adopted as an approximation and used to create output sequences with a trained LLM. Performing a beam search with a beam size of b consists of keeping b candidates, for which a greedy search is continued, and finally the candidate sequence with the highest probability is picked as the prediction. The constrained beam search (CBS; Kim, 2022) is a variant of this method, which tries to enforce the existence of specific words, so-called constraints, in the output. Since for the DST task it is important to have utterances containing the entities exactly as given in the annotation, we hope that adopting the CBS scheme in our method will increase the amount of annotations appearing in the generated utterances.

Synthetic Data Generation. The proposed ap-

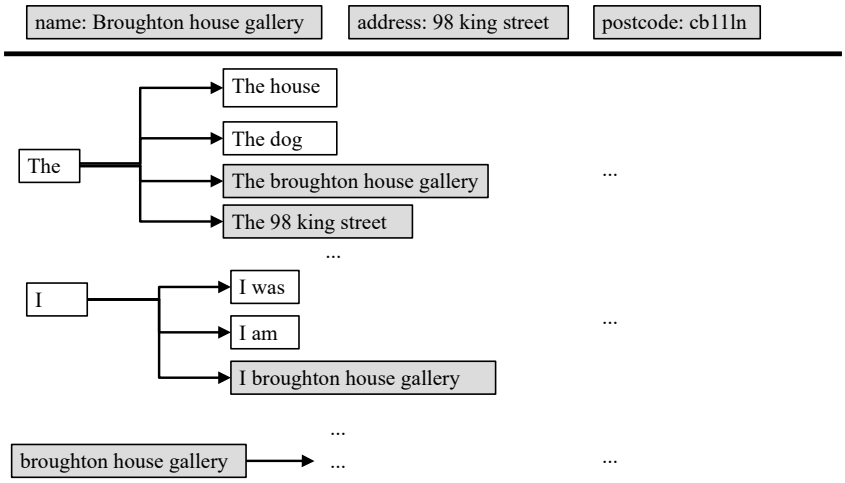

Figure 2: Simplified example step of the constrained beam search principle with three beams. Constraints marked with gray background, all constraints indicated on upper part.

proach is related to that of *bootstrapping*, which predates the Deep Learning era and is based on the triad of annotation a small amount of seed data, training a model and finally applying the model to unlabeled data, thereby generating silver-standard data (Tang and Surdeanu, 2023). Bootstrapping has recently been researched in Machine Learning applications, e.g., Tang and Surdeanu (2023); Eyal et al. (2021). The research into the generation of synthetic data in the form of dialogs has recently gained some interest, amplified by the advances in pretrained LLMs. Kim et al. (2022) distill a dataset containing 1.5 million social dialogs from a LLM. They use the LLM to first derive a short narrative and then again prompt the LLM to infer a dialog based on it. A similar approach has been studied by Miyazaki (2023), who address dialog generation based on story plots. To this end, they also investigate the prompting of LLMs. The synthesis of social dialogs via prompting LLMs was furthermore studied by Chen et al. (2023). They also extend this from dyadic conversations to multi-party dialogs, that contain more than two speakers.

3 Proposed Method and Materials

We use the MultiWOZ 2.4 (Ye et al., 2022) dataset as the basis for all of our experiments. This dataset is the result of multiple corrective iterations of the original MultiWOZ data (Budzianowski et al., 2018). It contains roughly ten thousand dialogs, of which two thousand in total are reserved as validation and test data. We make use of the official split between training, validation and test data. The dialogs were created with the Wizard-of-Oz (WOZ) technique (Kelley, 1984), where the interaction with a dialog system was imitated. While these can therefore be regarded as gold-standard data, the collection process is laborious and costly. To simulate a more realistic setting, we therefore randomly sample n dialogs from the training data and proceed with them as if they were the only collected conversations.

The goal of our method is to synthesize a large amount of dialogs based on a small amount of collected data. Instead of generating the whole dialog in one pass, we predict each utterance one after another based on the turn annotations with a sequence-to-sequence generator model. This can be described as using the annotations as a *blueprint* for the dialog.

The main steps of the method are given as 1.2, 2, 3, and 4 in Fig. 1. They first include the preprocessing of the data, so they fit a specified input/output format (x, y) for the sequence-to-sequence model (1.2). This format is depicted in Fig. 3. With this, we train a sequence-to-sequence model (2.) as the generator $\hat{y} = g_{CBS}(x)$. We write a generator model that uses the constrained beam search as its decoding strategy as $g_{CBS}(x)$. Using the trained $g_{CBS}(x)$ we perform the data augmentation by synthesizing dialogs using CBS (3.), and finally train the TOD system on this augmented data (4.). These steps are described in more detail below.

3.1 Input and Output of the Generator

Our generator model $\hat{y} = g(x)$ maps an input string x_i to an output utterance $\hat{y}_i$. An example of the input to and output of the generator is

in (x)	**<b_ctx>** **<b_bs>** train-leaveat: dontcare **<e_bs>** **<b_lbl>** train-leaveat: dontcare **<e_lbl>** **<b_pos>** 6 **<e_pos>** **<e_ctx>** Bot: there are 7 trains that can get you there , the earliest is levaing at 05:19 , the latest at 08:19 . do you have a time you would like to leave the station ? User:
out (y)	no , i do not care what time it leaves as long as it arrives in cambridge by 9:30 .
$\hat{y}$	i do not have a time to leave .

Figure 3: An example of the input x and output y that make up the training data as well as the output $\hat{y}g_{CBS}(x)$ of the trained generator with $n = 2500$. The typo at "levaing" is part of the dataset.

given in Fig. 3. Each x is made up of some meta-information, which we will call context, and the previous utterance. Thus, for an utterance y_i, the input is constructed as the concatenation of the context, the previous speaker and their utterance, and the speaker of y_i.

The context is constructed from

- the belief state (e.g., "train-leaveat: dontcare" in Fig. 3),
- the turns labels (e.g., the second "train-leaveat: dontcare" in Fig. 3 or "restaurant-name: the missing sock" as an example from a different utterance) and
- the turns index within the dialog (i.e., 6 in Fig. 3).

If the current utterance y_i is a system utterance, we also add the system act (e.g., "arrive: ?", if the system asks the user when he wants to arrive) to the context.

We introduce special tokens that indicate the start and end of a certain part of meta-information within the context. These markers are of the form *<b_META>* ... *<e_META>*, with META $\in$ {ctx, bs, lbl, pos, sysact} for context, belief state, label, position and system act, respectively.

To simulate the low resource settings, we randomly sample n dialogs from all train data X in the official MultiWOZ split to create a new training set X^n. We define $X^{\bar{n}} = X \setminus X^n$ to be the *blueprints* which we will use for the data augmentation.

3.2 Constrained Beam Search

With the aim of improving the inclusion of the annotations into the generated dialogs, we evaluate the effect of constrained beam search, which is implemented in the transformers library (Wolf et al., 2020). The main idea of CBS is to consider at every decoding step not only the highest-probability tokens, but also those defined as constraints. In our case, the constraints are the entities or strings contained in the annotations.

To avoid trivial but nonsensical outputs that simply concatenate the constraints, beams that do not (yet) fulfill the constraints are also kept in consideration during the decoding process. The implementation groups all candidates into so-called banks, depending on how close they are to fulfilling the constraint. Through a round-robin selection, b candidates, sourced from all banks, are preserved. Therefore, both outputs that are already closer to fulfilling the constraints, and those that are more sensible while being further from the constraints, are being considered. An example step of this constrained beam search generation is shown in Fig. 2. In this example, an undesired sequence that could be generated without the banks would be simply concatenating the annotation to output "broughton house gallery 98 king street cb11ln".

3.3 Data Augmentation Process

During the data augmentation process, we generate synthetic dialogs by creating each utterance within the dialog one after another. This is visualized in Fig. 4. The input string x, consisting of context and previous utterance, is created from the annotations of the current utterance and the previous utterance. During the training of the generator $g(x)$ or $g_{CBS}(x)$ (cf. step 2 in Fig. 1), we use the ground-truth previous utterance, since these come from available dialogs. The example output in 3 shows that the model correctly interprets the input x and generates an utterance $\hat{y}$ that contains the information that the user does not have to leave by a specific time. However, the information that the user has to arrive by a certain time was not given in the input (since it is missing in the annotation) and thus is also not represented in $\hat{y}$.

During the augmentation of the data, i.e., synthesizing new dialogs with the generator (cf. step 3 in Fig. 1), we use the previously model-generated utterance to keep the scenario realistic. Therefore, the evaluation of the models trained on the augmented data, does not rely on any ground-truth utterances.

In a real-world situation, the annotation for these

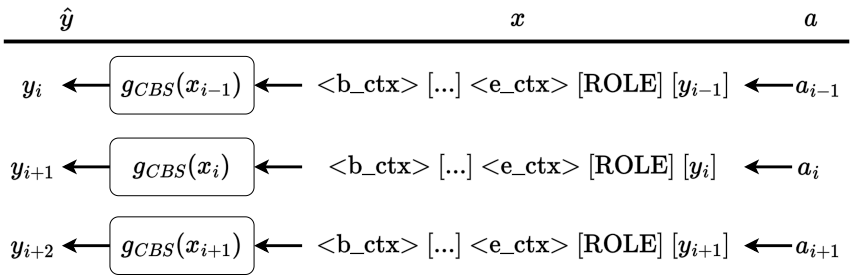

Figure 4: Visualization of the data augmentation process. The generator $g(x)$ creates the utterance predictions $\hat{y}$ from the input x, which is constructed based on the annotations a.

unknown dialogs $X^{\bar{n}}$, that we take from the unused MultiWOZ dialogs in the train split, will in general not be readily available. But the method needs annotations to be provided to the model as a *blueprint*. A solution to this is to have the dialog annotations created algorithmically. This is much simpler to do with traditional methods, which can be pattern-based, than to generate the dialog itself.

3.4 Training the TOD system

Once we have augmented the data by synthesizing all utterances for the dialogs in $X^{\bar{n}}$, we can train the TOD systems on this data. To this end, we keep the structure and annotations of the MultiWOZ train dataset but for all dialogs in $X^{\bar{n}}$, we replace the ground-truth utterance with the synthetic one, i.e., we set $y_i := g_{CBS}(x_i) \; \forall \, x_i \in x, \; \forall \, x \in X^{\bar{n}}$.

4 Experiments

Our experiments can be divided into two stages. The first stage consists of studying the generator and relates to RQ 1 (feasibility) and RQ 2 (constrained beam search). The second focuses on the dialog system and aims to answer RQ 3 (effectiveness). To evaluate the proposed method, we look at four different, simulated low(er) resource settings to answer RQ 4 (magnitude of data). In each of those, we assume that a certain amount n of annotated dialogs have been collected, which will be used for the CG4T approach. Concretely, we use $n \in \{150, 750, 1000, 2500\}$ in an attempt to understand the effect different magnitudes of data have.

4.1 Experiments on the Generator

To train our generator models, we randomly split X^n into a train, validation and test set, using a 70-10-20 split for each n, respectively. We further filter the training data so that it contains only utterances that have annotations. This has shown to be beneficial in preliminary experiments. Therefore, in each setting, the generator itself is trained and evaluated with a different amount of data. This is a necessary condition to realistically simulate the different settings. We chose the BART Base (Lewis et al., 2020) model as the architecture for our generator and used the pretrained checkpoint provided by the transformers library (Wolf et al., 2020).

We trained each generator model on its respective training data for 50 epochs using the Cross Entropy Loss, only saving the model with the lowest loss on the validation set. This was consistently achieved around the tenth epoch, after which the model started overfitting. An example of a completely synthesized conversation is given in Fig. 5.

User: i am looking for a train that leaves from cambridge on wednesday.
Bot: there are 5 trains peterborough on wednesday.
User: i need to leave peterborough after 12:45.
Bot: tr1879 leaves at 13:06 on wednesday.
User: i am also looking for a train.
Bot: i have a train that will get you to peterborough by 13:56. it will cost 16.50 pounds per ticket.
User: can you book that for 1 person?
Bot: yes, please.
User: i am also looking for a museum in the centre of town.
Bot: broughton house gallery is located at 98 king street, postcode cb11ln.
User: thanks.
Bot: you're welcome, i hope you enjoy your time in cambridge!

Figure 5: An example of a dialog synthesized by the generator. The ground-truth dialog is shown in Fig. 6 in the appendix A. Speaker roles are added in bold to improve readability.

To evaluate the generators, we investigate their language generation capabilities as well as their task completion, which we define as adhering to the annotation. We use BERTScore (Zhang et al.,

2019), which is especially suited to measure semantic similarity, to evaluate the text generation. This metric uses contextual BERT embeddings to calculate the similarity between the generated sentence and the ground-truth reference.

To measure the task-completion, we define the mean annotation metric as

$$Annot = \frac{1}{k}\sum_{j=1}^{k}(\frac{1}{l}\sum_{i=1}^{l} z(a_i, y_j)), \qquad (1)$$

with a_i being one annotation of all l annotations a, y_j is one of all k utterances and

$$z(a_i, y_j) = \begin{cases} 1 & a_i \text{ is in } y_j, \\ 0 & a_i \text{ is not in } y_j. \end{cases} \qquad (2)$$

In other words, we calculate the mean percentage of annotations that are part of the generated utterance. We define an annotation a_i as consisting of both the slot name and the slot value, since depending on the concrete utterance, containing the slot name can be desirable. However, the annotations are rather diverse, and the slot values do not only contain information entities, e.g., restaurant names or reservation times. So while in most cases we expect the text to contain exactly the slot value, there are also annotations where a slot has, for example, a boolean value. E.g., the slot "parking" for the domain hotel might have the value "yes". In this case, we do not necessarily want the utterance to contain the slot value but rather the slot name. Therefore, we check both the slot name and its value for occurrence. Still, it is possible that neither the slot name nor its value are supposed to be part of the utterance as-is. Take as an example the annotation "train-leaveat: dontcare". Therefore, to improve the meaningfulness of the metric, we also report it in proportion to the $Annot$ of the test dataset utterances (cf. Tab. 4). We write this as $Annot_R$.

4.2 Experiments on the Data Augmentation

The second stage of the experiments investigates if the synthetic dialogs generated in the previous experiments can improve the performance of TOD systems by means of data augmentation. Since the experiments on the constrained beam search have shown positive results, we adopted this method during the data augmentation.

That is, with $g_{CBS}(x)$ we synthesize dialogs for all $x \in X^{\bar{n}}$. To this end, we train two distinct recent model architectures, where the publications include an open-source code base. SOM-DST[1] (Kim et al., 2020) is an approach using explicit state memory and predicting the operation to perform for each slot at every turn. The second architecture is called STAR[2] (Ye et al., 2021), which introduces a slot self-attention mechanism to learn the correlations between the slots. This method predicts the slot-value-combination with the highest likelihood and can thus be classified as ontology-based, while SOM-DST can be classified as open-vocabulary, using only the dialog context (Ye et al., 2022). We trained both approaches with the hyperparameter setup provided by their authors.

We utilize the Joint Goal Accuracy (JGA) (Nouri and Hosseini-Asl, 2018) as a metric for this second stage of experiments. JGA is commonly used to evaluate models in the DST task. This metric measures, for each utterance, if the value for each slot is exactly and correctly predicted. Hence, the JGA is a rather strict evaluation metric. The results are reported in Tab. 5.

4.3 Descriptive Statistics of the Generated Data

We calculate the mean length and its standard deviation of the generated utterances and compare them to the ground-truth statistics in Tab. 1. These results show a trend of higher standard deviation with larger n. This makes sense, as it shows that a generator that was provided with more training data is capable of synthesizing utterances with higher diversity. The data at hand also shows that the system utterances are longer on average in every setting, including ground-truth. The average length of synthesized texts stays roughly the same in all n-scenarios for user and system utterances, respectively.

However, in all of these settings, they are shorter than in the ground-truth data. This can be explained by the absence of, e.g., fill words or additional information that does not concern the slots since the utterances are more tailored towards the annotation and are less likely to contain additional words. An example of this can be seen in the ground-truth dialog in Fig. 6 in the appendix A. The user says "i would like to go to peterborough and leave after 12:45, i have to attend a meeting beforehand", while in the generated utterance (cf. Fig. 5), the user says "i need to leave peterborough after 12:45".

[1] https://github.com/clovaai/som-dst
[2] https://github.com/smartyfh/DST-STAR

n	Length User	Length System
150	48.37 ± 17.56	55.35 ± 22.90
750	44.03 ± 20.55	56.89 ± 28.88
1000	40.11 ± 22.73	56.83 ± 28.45
2500	44.57 ± 23.90	61.79 ± 34.07
gt	61.99 ± 29.04	89.22 ± 38.68

Table 1: $\mu \pm \sigma$ of the length of user and system utterances for different n, where $n =$ gt shows the statistics of the ground-truth training data X. The length is measured as the number of characters including punctuation and white spaces.

	RTTR		MTLD	
n	User	System	User	System
150	2.29	3.02	26.18	36.07
750	2.42	4.09	32.08	43.76
1000	2.71	4.57	37.36	46.35
2500	3.33	6.17	46.10	55.67
gt	4.32	10.73	63.56	65.71

Table 2: Lexical richness measured as RTTR and MTLD of user and system utterances for different n, where $n =$ gt shows the statistics of the ground-truth training data X.

Besides the mix-up of departure and destination, the information about the user having to attend a meeting is also left out. This is to be expected, as the annotation does not track this information.

This can be seen as both a positive and a negative effect. On the one hand, the generator fails to imitate the length of the utterances. On the other hand, the synthesized texts are more concise.

We do not report the statistics on the number of turns since, due to the generation process, where we generate an utterance for each utterance-annotation existing in the data, the number of turns in all n-scenarios is the same as for the ground-truth data.

Furthermore, we evaluate the lexical diversity of the generated utterances using the root type-token ratio (RTTR; Guiraud, 1958) and the measure of textual lexical diversity (MTLD; McCarthy and Jarvis, 2010) with a threshold of 0.72. Both were computed with the LexicalRichness library[3]. The results, depicted in Tab. 2, are analogous to the findings regarding the utterance length. A higher n of available data consistently lead to higher lexical richness for both user and system utterances, as measured by both metrics. However, even in the setting of $n = 2500$, the utterances do not reach the lexical diversity of the ground-truth utterances. The fact that system utterances invariably show higher diversity can be explained by the usual course the dialogs have, in that the system provides slot values, which can be expected to have more lexical variance than the rest of the utterance since they contain, e.g., restaurant names or booking references.

5 Qualitative Error Analysis

To better understand the weaknesses of the generator, we performed a qualitative error analysis by comparing generated dialogs to their ground-truth. In the following, we will reference the errors with regard to the generated dialog in Fig. 5 and its ground-truth in Fig. 6 in the appendix A.

First, errors in the annotation will naturally be replicated in the generated utterance. As an example, while in the ground-truth the user requests the reference number for the ticket and the system delivers it, the generated dialog does not mention the reference number at all. However, since the annotation does not contain the reference number either, we cannot expect it to be generated. Second, we can see that sometimes the speaker role seems to not be completely taken into account, leading to formulations that are unexpected for the dialog system, such as the "yes, please" response the system gave to the user's request to book a train ticket. Lastly, while the concrete text of the slot values will in general be correct due to the constrained beam search, the embedding of them into context still contains errors, both semantically and syntactically. An example we can see in the generated dialog is first that the chatbot produces a faulty utterance "there are 5 trains peterborough on wednesday". Thus, at the same time, prematurely giving the number of trains for the destination but also not specifying it is the destination. The premature destination mention is due to it already being part of the belief state annotation for this turn.

To sum up, while the model in general achieves decent results, there are still multiple caveats, and the error analysis emphasizes the importance of the annotations.

6 Results and Discussion

Our experiments show that the model is able to generate utterances based on the provided annotations. These utterances resemble the language defined by

[3] https://github.com/LSYS/LexicalRichness

	F1		Precision		Recall	
n	$g_{CBS}(x)$	$g(x)$	$g_{CBS}(x)$	$g(x)$	$g_{CBS}(x)$	$g(x)$
150	87.84	**89.27**	89.12	**91.05**	86.65	**87.60**
750	89.41	**89.75**	91.36	**91.90**	**87.76**	**87.76**
1000	89.25	**89.60**	91.09	**91.77**	87.54	**87.60**
2500	89.56	**89.90**	91.44	**91.99**	87.81	**87.98**

Table 3: F1, Precision and Recall of the BERTScore for the different scenarios *with* and *without* constrained beam search as indicated by $g_{CBS}(x)$ and $g(x)$, respectively. Better marked in bold for each metric and n, respectively.

the training data, as is shown by the BERTScore (cf. Tab. 3). The improvements with increased n are negligible for this metric. This can be attributed to the fact that, due to fine-tuning a pretrained LLM, the model already had good general language generation capabilities to begin with. However, with regard to the average utterance length (cf. Tab. 1) and lexical richness (cf. Tab. 2), the generated text does not perfectly align with the training data. Nevertheless, we consider the model to have a good, but not perfect, language generation capability for this specific task. Besides language generation, RQ 1 also tends to the task-completion, which in this case is adhering to the annotations.

We report the results for the $Annot_R$ metric in Tab. 4. These show that the model is also capable of generating utterances that adhere to the annotations. The performances for the different n-scenarios are comparable. It is an important insight that the $Annot_R$ metric is more expressive. This makes up for a weakness of the metric, which stems from the different kind of annotations that exist.

The dialog in Fig. 5 shows that when few annotations are provided for an utterance, the generation can lead to an unfitting utterance. For example, when the user asks to book the train ticket for one person, the system answers with "yes, please". From the usual dialog flow the MultiWOZ conversations have, it is clear that this utterance usually belongs to a user and not the system. The generator mistakenly used this as a confirmation, even though it is unfitting for this speaker in this scenario.

Since both the language generation and the task-completion capabilities are sufficient, we answer RQ 1 positively: We successfully used a pretrained sequence-to-sequence model to generate synthetic dialogs based on annotations.

The effect of the constrained beam search is also shown in tables Tab. 3 and Tab. 4. As is to be expected from the method, constraining the generation led to slightly worse results in the BERTScore metrics. This is consistent over all n. However, regarding the $Annot$ metric, the constrained beam search improved the outcome substantially. With it, the $Annot_R$ relative to the test data was near-perfect over all n. A special case is $n = 150$, where $Annot_R > 1$. This means that on average, the utterances predicted by the generator model contained more annotations than the actual test data. This effect can be explained by the small sample size, as well as the fact that the ground-truth $Annot$ of the test data over all n is roughly 0.73. The large increase in $Annot$ score with a small decline in BERTScore metrics suggests that CBS is useful for annotation-based generation, which leads us to answer RQ 2 positively.

As a result, we utilized constrained beam search for the second stage of experiments. Tab. 5 shows the results for the selected TOD system architectures for the n-scenarios. We report both the results of using only these n dialogs, and additionally using the synthetic dialogs, i.e., with CG4T.

The first finding is that for all n and both architectures, CG4T did improve the JGA, therefore showing that the synthetic dialogs generated from annotations can indeed increase the performance of both open-vocabulary and ontology-based TOD systems. Consequently, we answer RQ 3 positively as well. Still, the results show that with larger n, the benefit gained through the usage of CG4T decreases. This is to be expected given the nature of the method, which led us to pose RQ 4. Finally, Tab. 5 demonstrates the significant differences in JGA between the two approaches.

Regarding RQ 4 we evaluate the combined results from experiments in stages one and two. Stage one showed that only a few gold standard dialogs are needed to get good language generation capability. Moreover, thanks to CBS, we can also attain satisfying results for task-completion. The second stage showed that the more dialogs available, the lesser the improvement. From this we

conclude that a) the less data one has, the more sense it makes to use this method and b) for this concrete scenario, the method is nonessential past a magnitude of 2500 dialogs.

	$Annot$		$Annot_R$	
n	$g_{CBS}(x)$	$g(x)$	$g_{CBS}(x)$	$g(x)$
150	**0.76**	0.69	**1.04**	0.94
750	**0.72**	0.63	**0.99**	0.87
1000	**0.71**	0.62	**0.98**	0.85
2500	**0.72**	0.61	**0.99**	0.84

Table 4: $Annot$ and $Annot_R$ in relation to the test data for the different scenarios *with* and *without* constrained beam search as indicated by $g_{CBS}(x)$ and $g(x)$, respectively. Better marked in bold for each metric and n, respectively.

	STAR		**SOM-DST**	
n	**as is**	**CG4T**	**as is**	**CG4T**
150	3.50	**42.40**	1.49	**23.83**
750	38.86	**57.92**	21.65	**31.79**
1000	45.71	**63.45**	25.73	**32.82**
2500	63.55	**64.97**	35.18	**36.02**
8420	74.46	-	41.69	-

Table 5: JGA for different scenarios of available training data for the two reference models. The columns marked "CG4T" report the results after applying the proposed method to extend the number of training dialogs to achieve $n = 8420$ with synthetic data. The columns marked "as is" report results without data augmentation. The last row shows the result when using the full training data. Thus, CG4T is not applicable. Better marked in bold for each setting and n, respectively.

7 Conclusion

In this work, we studied whether it is practicable to synthesize dialogs based on annotations to augment the collected ground-truth data for training a TOD system. To this end, we focused on four research questions regarding the feasibility (RQ 1), the effect of CBS during decoding (RQ 2), the performance improvement when using the synthetic dialogues to train a dialog system (RQ 3), and the relation between available data and the effect of the method (RQ 4).

We saw that even with small amounts of dialog, we can train a generator that creates utterances from annotations using a sequence-to-sequence strategy.

While the constrained beam search had slightly adverse effects on the language generation capabilities, it provided significant improvements to the task-completion, i.e., adhering to the annotation.

Augmenting the training data with the synthetic dialogues relevantly increased the performance of the dialog system as measured by the JGA metric.

The smaller the number of available conversations, the greater the effect of the CG4T approach. Past a scale of roughly 2500 collected ground-truth dialogs, we would discourage augmenting the training data in this way since the improvements are too small for the increased effort. Our research showed at multiple stages the importance of having annotations of the highest quality when using the proposed approach.

Limitations

While offering multiple advantages discussed above, our method also has some limitations and drawbacks.

First, since CG4T relies on training a generator model, it inevitably requires additional effort and computing time. One could also argue that this requires more specific knowledge (e.g., programming) than simply prompting a LLM such as ChatGPT to generate dialogs via an API.

Second, as demonstrated by our experiments, the proposed approach is only sensible up to a certain scale of existing data, and it does take at least some data to train the generator.

Third, the approach is heavily dependent on the annotation. As shown on multiple occasions in this work, if the annotation is lacking information, the synthetic utterance will naturally not contain it either. Therefore, to synthesize the dialogs that will be used for training, one has to have access to high-quality annotations.

Additionally, the approach needs to have the *blueprints* to the dialog in the form of annotations. If they do not exist in a real-world scenario, they have to be generated in advance. We argue that this is also feasible with traditional algorithms and heuristics.

Lastly, our experiments are limited to the MultiWOZ dataset. While it is reasonable to assume that the general behavior will be similar for conversations from other distributions, we did not perform experiments on this.

References

Namo Bang, Jeehyun Lee, and Myoung-Wan Koo. 2023. Task-optimized adapters for an end-to-

end task-oriented dialogue system. *arXiv preprint arXiv:2305.02468*.

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.

Maximillian Chen, Alexandros Papangelis, Chenyang Tao, Seokhwan Kim, Andy Rosenbaum, Yang Liu, Zhou Yu, and Dilek Hakkani-Tur. 2023. PLACES: Prompting language models for social conversation synthesis. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 844–868, Dubrovnik, Croatia. Association for Computational Linguistics.

Qinyuan Cheng, Linyang Li, Guofeng Quan, Feng Gao, Xiaofeng Mou, and Xipeng Qiu. 2022. Is MultiWOZ a solved task? an interactive TOD evaluation framework with user simulator. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1248–1259, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30:4299–4307.

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 Twelfth Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.

Matan Eyal, Asaf Amrami, Hillel Taub-Tabib, and Yoav Goldberg. 2021. Bootstrapping relation extractors using syntactic search by examples. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1491–1503, Online. Association for Computational Linguistics.

P. Guiraud. 1958. *Problèmes et Méthodes de La Statistique Linguistique*. Dodrecht: D. Reidel.

Wanwei He, Yinpei Dai, Min Yang, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022. Space-3: Unified dialog model pre-training for task-oriented dialog understanding and generation. *arXiv preprint arXiv:2209.06664*.

J. F. Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems*, 2(1):26–41.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Chan Woo Kim. 2022. Guiding Text Generation with Constrained Beam Search in Transformers. [Online; posted 11-March-2022].

Hyunwoo Kim, Jack Hessel, Liwei Jiang, Ximing Lu, Youngjae Yu, Pei Zhou, Ronan Le Bras, Malihe Alikhani, Gunhee Kim, Maarten Sap, et al. 2022. Soda: Million-scale dialogue distillation with social commonsense contextualization. *arXiv preprint arXiv:2212.10465*.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2020. Efficient Dialogue State Tracking by Selectively Overwriting Memory. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 567–582, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Philip M McCarthy and Scott Jarvis. 2010. Mtld, vocd-d, and hd-d: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior research methods*, 42(2):381–392.

Chiaki Miyazaki. 2023. Dialogue generation conditional on predefined stories: Preliminary results. *IEEE access : practical innovations, open solutions*, 11:85589–85599.

Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking model. *arXiv preprint arXiv:1812.00899*.

OpenAI. 2022. OpenAI: Introducing ChatGPT. [Online; posted 30-November-2022].

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Zheng Tang and Mihai Surdeanu. 2023. Bootstrapping neural relation and explanation classifiers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 48–56, Toronto, Canada. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2022. MultiWOZ 2.4: A Multi-Domain Task-Oriented Dialogue Dataset with Essential Annotation Corrections to Improve State Tracking Evaluation. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 351–360, Edinburgh, UK. Association for Computational Linguistics.

Fanghua Ye, Jarana Manotumruksa, Qiang Zhang, Shenghui Li, and Emine Yilmaz. 2021. Slot Self-Attentive Dialogue State Tracking. In *Proceedings of the Web Conference 2021*, WWW '21, pages 1598–1608, New York, NY, USA. Association for Computing Machinery.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2022. A survey of controllable text generation using transformer-based pre-trained language models. *arXiv preprint arXiv:2201.05337*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Jeffrey Zhao, Raghav Gupta, Yuan Cao, Dian Yu, Mingqiu Wang, Harrison Lee, Abhinav Rastogi, Izhak Shafran, and Yonghui Wu. 2022. Description-driven task-oriented dialog modeling. *arXiv preprint arXiv:2201.08904*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

A Example ground-truth Dialog

User: i need to take a train out of cambridge , i will be leaving town on wednesday .
Bot: there are 5 trains out of cambridge on wednesday . do you have a departure time in mind ?
User: i would like to go to peterborough and leave after 12:45 , i have to attend a meeting beforehand .
Bot: tr1879 leaves at 13:06 on wednesday . will that work for you ?
User: what is the price of the fair and could you tell me what is the arrival time into peterborough ?
Bot: the price is 16.50 pounds and the train will arrive in peterborough by 13:56 wednesday . would you like me to book a ticket for you ?
User: yes , 1 ticket please , can i also get the reference number ?.
Bot: i booked you 1 seat on tr1879 departing cambridge for peterborough on wednesday at 13:06 , reference: xu1qlhvw . can i help further today ?
User: i am also looking for a museum in centre . could you give me the address , postcode and phone number of 1 ?
Bot: the broughton house gallery is located at 98 king street , postcode cb11ln . their phone number is 01223314960 . is there anything else you need ?
User: that s all i need right now . thanks , you have been very helpful !

Figure 6: The ground-truth to the synthesized dialog in Fig. 5. Speaker roles are added in bold to improve readability.