Velocidapter: Task-oriented Dialogue Comprehension Modeling Pairing Synthetic Text Generation with Domain Adaptation

Taha Aksu[†‡], **Zhengyuan Liu**[‡], **Min-Yen Kan**[†], **Nancy F. Chen**[‡]
† National University of Singapore, Singapore
‡ Institute for Infocomm Research, A*STAR, Singapore
taksu@u.nus.edu,
liu_zhengyuan@i2r.a-star.edu.sg,
kanmy@comp.nus.edu.sg,
nfychen@i2r.a-star.edu.sg

Abstract

We introduce a synthetic dialogue generation framework, Velocidapter, which addresses the corpus availability problem for dialogue comprehension. Velocidapter augments datasets by simulating synthetic conversations for a task-oriented dialogue domain, requiring a small amount of bootstrapping work for each new domain. We evaluate the efficacy of our framework on a task-oriented dialogue comprehension dataset, MRCWOZ, which we curate by annotating questions for slots in the restaurant, taxi, and hotel domains of the MultiWOZ 2.2 dataset (Zang et al., 2020).

We run experiments within a low-resource setting, where we pretrain a model on SQuAD, fine-tuning it on either a small original data or on the synthetic data generated by our framework. Velocidapter shows significant improvements using both the transformer-based BERT-Base and BiDAF as base models. We further show that the framework is easy to use by novice users and conclude that Velocidapter can greatly help training over task-oriented dialogues, especially for low-resourced emerging domains.

1 Introduction

Humans perform dialogue interactions to accomplish common tasks: work email threads, nurse–patient conversations, customer service conversations, *etc.* (*cf.* Table 1). Systems that can comprehend and answer key questions about these dialogues can significantly speed up information extraction from such documents. However, studies in machine reading comprehension (MRC) largely focus on the written form of text, such as news articles, Wikipedia documents, *etc*. These are not directly applicable to dialogue comprehension. While there are datasets that incorporate dialogue components in MRC (Sun et al., 2020; Reddy et al., 2020; Choi et al., 2018), they are not representative of task-oriented dialogue. Such dialogue comprehension systems are currently constrained by the lack of annotated data.

U1: Hi I would like a British food restaurant in the *centre*.
S1: Sure, do you have a preference over the price range?
U2: Only the best for my family, we'll take the *expensive* one. Book us a table for 5 at 14:00 today.
S2: Sorry, I am afraid there is no such place, shall we try another cuisine?
U3: Let's try Italian instead.
S3: Caffe Uno is a very nice, expensive Italian restaurant in the center. Would you like a table?
U4: Actually, I think I will stick with *British* food.
S4: *Fitzbillies Restaurant* is an expensive place centrally located and serves British.
U5: Can you book me a table for Thursday for 5 people at *13:00*?
S5: Your reservation at Fitzbillies Restaurant is successful for 5 people at 13:00 today. Anything else I can help you with?
U6: No, that's all I need. Thanks for your help!
Q1: What type of food does the user want to have? A1: *British*
Q2: What part of town is the restaurant located at? A2: *Centre*
Q3: What is the preferred price range of the user? A3: *Expensive*
Q4: What time is the reservation for? A4: *13:00*
Q5: What is the name of the booked restaurant? A5: *Fitzbillies Restaurant*

Table 1: (top) Sample dialogue between a user and the system in the restaurant booking domain; (bottom) and its associated question–answer pairs. Italicized, colored words indicate answer spans in the text.

A task-oriented dialogue is a form of information exchange where the system obtains user preferences (*i.e.* slot values for attributes) by conversation. The dynamic flow between speakers in these dialogues introduces additional challenges such as: (1) Mind change: Speakers might state their preference over some attribute/slot two or more times (*cf.* Table 1 U3&U4: Italian → British food); (2) Topic drift: Speakers might change the topic of the conversation abruptly (*cf.* Table 1 U2: price range

Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 133–143
July 29–31, 2021.

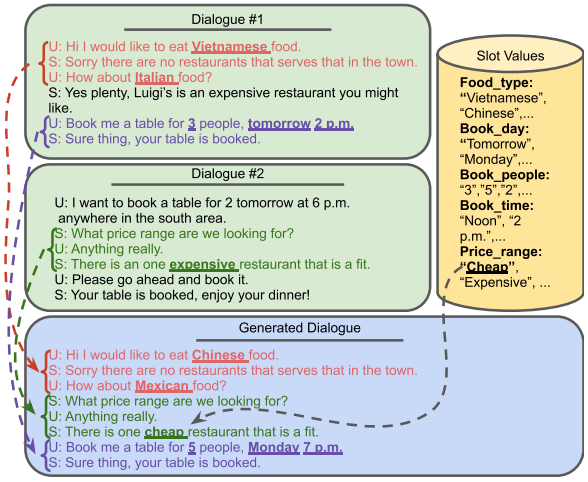

Figure 1: An example of how Velocidapter generates a synthetic dialogue using turn templates from two existing dialogues in the restaurant booking domain.

→ date and time); (3) Zero anaphora: Information is represented in several turns that are spoken by different speakers. Thus, speakers may use a gap in the text to refer back to a previous expression (*cf.* Table 1 U5: "book me a table ..." → "Fitzbillies Restaurant"); (4) Over-explanation: Decisions are taken real-time during the conversation thus speakers might make overly verbose explanations of their preferences (*cf.* Table 1 U2: "Only the best for my family...").

Among recent data augmentation studies, Liu et al. (2019) contribute the sole prior work explicitly on task-oriented dialogue comprehension. However, their synthetic data generation is scoped within a clinical scenario, with templates of inquiry–response pairs between nurses and patients. This limits dialogue-specific traits, such as mind change and co-reference, to consecutive turns only.

Inspired by this prior work, we introduce Velocidapter, which can augment a handful of task-oriented dialogues to a synthetic dataset that is larger by several orders of magnitudes. Figure 1 shows a simple, intuitive example of Velocidapter's synthetic generation in the restaurant booking domain. Different from prior work, we define templates as dialogue chunks (*i.e.* several contiguous turns), which we call *discourse templates*. This lets us design dialogue-specific challenges that span over multiple dialogue turns (*e.g.* mind change, zero anaphora, *etc.*). We further aim to expand prior work by addressing scalability issues for task-based dialogue comprehension by leveraging synthetic generation with a mutual concept: domain adaptation (DA). This pairing is synergistic: DA gives the model the necessary pretraining to generalize well, and the synthetic generation process yields sufficient data in the target domain to effectively fine-tune the model.

To use Velocidapter, a user extracts pairs of discourse templates from a few development dialogues in the target domain (*cf.* colored dialogue chunks within dialogues 1 and 2 in Figure 1), a value list for each slot (*cf.* slot values in Figure 1), and a question list for each slot. With these inputs, Velocidapter simulates a synthetic corpus of task-oriented dialogues by mixing turn templates from several dialogues and filling templates with values from the slot value list. Finally, it matches each dialogue to a set of questions according to the slots they contain. This synthetic dataset is then used to train or fine-tune a dialogue comprehension model in the target domain.

We contribute a new dataset, MRCWOZ, to evaluate our framework[1]. This dataset is generated from the existing large dialogue corpus, MultiWOZ 2.2 (Zang et al., 2020), which is used for DST (dialogue state tracking) task. We form training and test sets of MRCWOZ from the respective sets in MultiWOZ by annotating questions for each unique slot type in the restaurant, hotel, and taxi domains. Note that the formation of MRCWOZ is completely separate from our augmentation framework. We show that within a low resource setting, models using our framework significantly outperform models using original target data (raw data). Specifically, Velocidapter outperforms the raw training by 0.26, 3.82, and 13.23 F1 scores in the restaurant, hotel, and taxi domains, respectively. These gains are obtained at little human time cost and are robust: through a user study, we show that templates extracted by a novice human in under an hour, still lead to significant improvements over raw training.

To the best of our knowledge, this is the first study to make use of the inherent clustered structure of task-oriented conversations to augment a large set of instantiated dialogue datasets. Our framework is also the first to address dialogue-specific challenges that span over several turns within a machine comprehension perspective. We thus conclude that this approach potentially can greatly facilitate the rapid advancement of understudied task-oriented dialogue areas, which lack sufficient corpora.

[1]Framework and experimental data available at https://github.com/cuthalionn/Velocidapter

2 Related Work

Reading Comprehension. Corpora on reading comprehension are largely limited to written text, *e.g.*, SQuAD (Rajpurkar et al., 2018b), MARCO (Nguyen et al., 2016), RACE (Lai et al., 2017), TriviaQA (Joshi et al., 2017) and many others (Hermann et al., 2015; Hill et al., 2016; Richardson et al., 2013; Kociský et al., 2017; He et al., 2018). These datasets are all collections of written passages: SQuAD collects Q–A pairs for Wikipedia articles; MARCO collects pairs from Bing, along with context passages; RACE from English exams; and TriviaQA collects pairs with evidence documents.

A few incorporate a conversational component to the MRC task. DREAM (Sun et al., 2020), FriendsQA (Yang and Choi, 2019) and a study by Ma et al. (2018) are all dialogue comprehension datasets. Although a valuable source, these do not apply to task-oriented dialogue comprehension, as all three are open-domain and multi-party. In contrast, CoQa and QuAC do employ two-party dialogue; however, their task is to conversationally answer questions about a passage, diverging from our task definition (Reddy et al., 2020; Choi et al., 2018).

Synthetic Text Generation. Natural language generation (NLG) systems are basic components of text generation. These systems can be classified into three different categories by their approach: data-driven, rule-based, and template-based. The analysis in the English-to-English NLG challenge (Dušek et al., 2020) concluded that template-based systems outperform neural systems in terms of output diversity and complexity.

Liu et al. (2019) try to train a task-oriented dialogue comprehension model with data from a synthetic data generator that simulates human-human dialogues. However, their system is confined to turn-level transformations, limiting the information flow within the generated dialogue. Shah et al. (2018) also use a template-based approach: they simulate dialogue templates with a rule-based system and then use crowdsourced workers to fill in the templates, generating a dialogue corpus. This process requires manual work for each dialogue created.

Data-driven approaches largely lack the transparent controllability and diversity provided by a template-based approach (Dušek et al., 2020). There are, however, studies that tackle this problem. Wiseman et al. (2018); Ye et al. (2020) try to learn templates from data and use them to generate text. Peng et al. (2020) uses few-shot learning to train models that can be easily adapted to new domains. However, these are not convenient for use in our setting, as they all assume at least an unlabeled dataset in the domain to generate the synthetic data.

Domain adaptation (DA). With the recent increase in the number of large corpora, DA has attracted the attention of many MRC researchers. Zhao and Liu (2018) and Wiese et al. (2017) use models pretrained with the SQuAD dataset to increase performance in the target domain, utilizing small amounts of labeled data. In (Hazen et al., 2019), the authors pretrain models over the many large MRC corpora (SQuAD, NewsQA, *etc.*), then fine-tune them on the associated development set. Golub et al. (2017) and Wang et al. (2019) both use a data-driven approach generating synthetic questions on target unlabeled data and fine-tuning models on this synthetic data. In a variant, Li et al. (2019) instead *ensemble* pretrained language models, before appropriate fine-tuning.

3 Velocidapter: Data Generation Framework

Let us first formalize our task. Our goal is to create a task-oriented dialogue-augmentation framework F, that given a list of dialogue turn templates T, a slot label-value list S_V, and finally a slot label-question list S_Q, can generate a large dialogue comprehension dataset D. F creates individual synthetic dialogues in D by composing turn templates from T, filling these turn templates with values from S_V, and finally matching these to questions from S_Q. D then can be used to train or fine-tune a task-oriented dialogue comprehension model.

Task-oriented dialogues can be deconstructed as having dialogue units that convey slot values for particular attributes. We name these atomic units that are composed to creat dialogues in our framework as *discourse templates*. Velocidapter takes as input a set of manually-extracted discourse templates and outputs instantiated dialogues that are of orders of magnitudes larger in scale. This facilitates the robust training of large models from just a few dialogue instances. Figure 2 shows the end-to-end pipeline of our framework. To use Velocidapter a user extracts the turn templates from a small, task-oriented dialogue development set (*e.g.* in Figure 2 turn templates in 2A are extracted from dialogues in 1A), a list of values for each slot (2B), and a

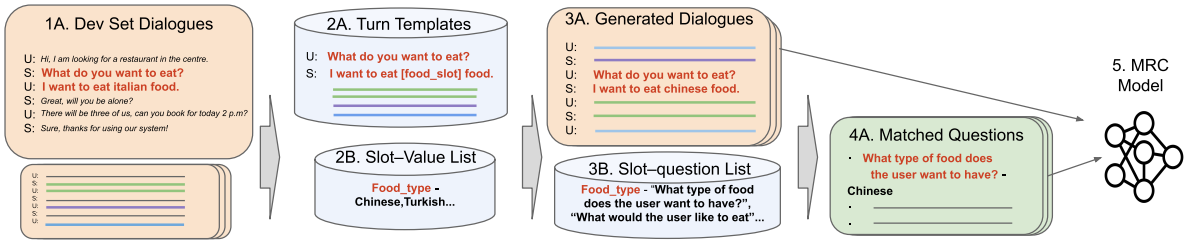

Figure 2: Velocidapter starts with manual turn template extractions from a small development set of dialogues (left). We additionally provide a list of questions and values for each possible slot (middle). Velocidapter then using the turn templates and slot values creates a new set of synthetic dialogues and matches each dialogue to their relevant question. This final synthetic dataset is then used to train/fine-tune an MRC model.

list of questions for each slot (3B). Velocidapter then generates individual dialogues (3A) and their associated Inquiry–Response pairs (4A), by executing three steps: (1) structured corpus construction, (2) dialogue template generation, and (3) dialogue corpus generation.

3.1 Structured Corpus Construction

Traditionally, creating a corpus is a painstaking process, involving the collection of data from authentic environments, creation of coding guidelines, followed by manual coding with checks. Velocidapter eases this by structuring this once-only manual process into three stages: discourse template construction, slot value enumeration, and question construction. We review these steps grounded with examples taken from the restaurant domain of the MRCWOZ dataset.

3.1.1 Discourse Template Construction

We classify the discourse templates into two forms of communication: 1) Salutation and 2) Request–Response. Salutation templates provide the pragmatic framing of the conversation, such as a greeting and farewell (*i.e.* "Hello, I am looking for a restaurant to dine in"), whereas request–response templates concern information exchange through requests (by system or user) and responses; Table 3 depicts some sample request–response templates. Each request–response template is associated with at least one slot label, where each slot label consists of a base and an optional *arbitrary* prefix separated by a dash (*e.g.* arbitrary-food_type, price_range, city_area). The base determines the values and questions that the slot will be matched to. The prefix *arbitrary* indicates that this placeholder's value is not the final answer for the subject slot label. The framework considers this keyword to guarantee two conditions: (1) that two slots with the same base, are filled with different values, and (2) that the final answer is indexed to point to the one that is not arbitrary.

There is no restriction on the number of turns a discourse template can contain. This feature is useful in designing complex conversations that may be expected in the test set. Table 3 shows examples of such turn templates. The first sample in the table illustrates a frequent phenomenon in dialogue, where the user over-explains the background of a slot decision. The correct area slot in this discourse template is given by the latter slot label *city_area*. The second sample shows a discourse template with four turns that instantiates another common flaw where a user changes a decision she made in an earlier turn. The framework expects each request–response discourse template to start with a system turn. However, by supporting multiple turns in a discourse template, Velocidapter allows for mixed initiative, where the user can change the conversation topic (*cf.* final sample in Table 3).

3.1.2 Slot Label–Value List Construction

The slot label–value list (Figure 2-2B) is a mapping from each label to its possible values. The slot label–value list must have an entry for each unique slot label introduced in the discourse templates, along with its possible filler values. The left hand side of the Table 2 shows shortened lists for three slot labels *food_type*, *city_area* and *price_range*.

3.1.3 Question List Construction

The question list (Figure 2-3B) is provided to match each dialogue to a set of questions. The question list must also have an entry for each slot label introduced in the discourse templates, along with the possible questions that refer to the label. Table 2 (R) shows question lists of three slot labels in the restaurant booking domain.

Slot Label	Slot Values
food_type	Turkish Mexican
city_area	Centre Noth South East
price_range	Expensive Cheap Moderate

Slot Label	Questions
food_type	What type of food does the user want to have? What would the user like to eat?
city_area	What part of the town does the user willing to dine in? In which area does the user want to reserve the restaurant?
price_range	What is the preferred price range of the user? Which price range is the user comfortable with?

Table 2: (L) Snippet of a Sample Slot Label–Value List which includes a corresponding entry for each unique slot defined. (R) Snippet of a Sample Slot Label–Question List which includes a corresponding entry for every unique slot label defined.

Speaker	Turn
Over-Explanation:	
System	Which part of the city would you favor?
User	The *arbitrary-city_area* is too far from my place, I think *city_area* would work the best.
Mind Change:	
System	What cuisine would you like to try?
User	Lets try *arbitrary-food_type*, please.
System	Okay, sounds good.
User	Sorry, I want to have *food_type* type instead.
Mixed-Initiative:	
System	What are you planning to eat?
User	I am planning to eat *food_type*.
System	Sure thing, I can check for that.
User	Please find me a place that is in *price_range* price range.

Table 3: Sample Request–Response Discourse Templates. Each Request–Response template provides an information exchange between the user and system over at least one slot label (*i.e. food_type*).

3.2 Dialogue Template Generation

The dialogue template generation uses discourse templates provided by the user in the previous section to create the dialogue templates. The system starts by choosing a salutation discourse template from the template pool. It then iteratively chooses a request–response template to add to the dialogue template (constrained to not add duplicate slot labels), until a predetermined lower boundary is reached. A generated dialogue template in the restaurant domain can be seen on the left-hand side of Table 4. As each extracted template is an information exchange about certain slot labels and does not depend on previous or next templates, adding them one-by-one creates a fluent and coherent dialogue that can feature common conversational phenomena, such as mind change, during the discourse template construction process.

3.3 Dialogue Corpus Generation

The final step, dialogue generation, fills the dialogue templates generated in the last step using the list of slot label–value pairs. The process is randomized, but also constrained to avoid select values for any previously instantiated label. The framework permutes each dialogue template by filling in a range of slot values until it exceeds a predetermined user-specified count. Each generated dialogue is stored with a list of questions according to the slot labels they contain. The right hand side of Table 4 illustrates a generated fully-instantiated dialogue. This ends the synthetic data generation process. By running this process many times, we can create an arbitrarily-large dataset that can be used to train a dialogue comprehension model.

4 Experiments

To evaluate we need a dataset for dialogue comprehension. Unfortunately, no suitable dataset exists for this purpose, so we pick an existing dialogue dataset and retrofit it for our evaluation purposes. We start with the MultiWOZ dataset, commonly used for DST.

We choose a range of domains from MultiWOZ to work with to showcase domain agnostic feature of our framework. We leave hospital and police domains out, following past work (Campagna et al., 2020) since they lack correct annotations and validation and test sets. From the remaining five we choose restaurant, hotel, and taxi domains as their pools of slot labels show very few overlaps thus resulting in a diverse dataset. The resulting corpus contains 2,409 dialogues, averaging 8.92 turns per dialogue, and 12.2 tokens per turn. But since MultiWOZ does not come with dialogue comprehension questions natively, we supply our our own hand annotated questions as detailed next.

Speaker	Turn
User	Hello, I would like to find a place to dine in, there will be *restaurant_bookpeople* of us.
System	What cuisine would like to try?
User	Let's try *food_type*, please.
System	Okay sounds good.
User	Sorry, I want to have *food_type* instead.
System	Which part of the city would you favor?
User	The *arbitrary-city_area* is too far from my place, I think *city_area* would work the best.
System	Okay, does *restaurant_bookday* sound good?
User	Yes, that should work.
System	Great, your booking is successful. Anything else I can help you with?
User	This is all I wanted for today, thank you.
System	Thanks, good bye.

Speaker	Turn
User	Hello, I would like to find a place to dine in, there will be 4 of us.
System	What cuisine would like to try?
User	Let's try British, please.
System	Okay sounds good.
User	Sorry, I want to have Italian instead.
System	Which part of the city would you favor?
User	The center is too far from my place, I think south would work the best.
System	Okay, does Friday sound good?
User	Yes, that should work.
System	Great, your booking is successful. Anything else I can help you with?
User	This is all I wanted for today, thank you.
System	Thanks, goodbye.

Table 4: (L) Velocidapter-generated dialogue template, using the user-provided discourse templates. (R) Fully-instantiated Velocidapter-generated dialogue, created by filling the generated dialogue template in (L).

	Train		**Test**	
Domain	# Dial	# S–Q	# Dial	# S–Q
Hotel	650	2859	71	318
Restaurant	1250	4495	65	316
Taxi	321	965	52	157

Table 5: Domain specific dialogue (Dial.) and slot–question (S–Q) number statistics of MRCWOZ for both train and test splits. As there is a question corresponding to each slot in a dialogue, their numbers are identical.

For each slot type in MultiWOZ, we manually create a list with a few questions. We then match each dialogue to a set of questions according to the slots present in the dialogue to create our MultiWOZ dialogue comprehension dataset, which we term MRCWOZ. As a result of this process, MRCWOZ pairs each dialogue with an average of 4.2 questions. The domain-specific statistics of MRCWOZ data can be seen in Table 5. This resultant training and testing split are identical with MultiWOZ. Note specifically that this generation process is completely separate from the dialogue augmentation in Velocidapter that we evaluate.

We also randomly sample a small development set, *vel_dev* containing few dialogue (*e.g.* 2–10 dialogues) from the training set of each domain to extract turn templates for Velocidapter. During sampling, we ensure that the final set of dialogues cover all possible slots encountered in the test set so that the trained model will be exposed to each slot at least once (*e.g.* food_type, booking_day, *etc.*).

We fine-tune the BERT-Base (Devlin et al., 2019a) and BiDAF models (Seo et al., 2016) in experiments representing three different scenarios/datasets: (1) In a high-resource scenario on MRCWOZ, which serves as an upper bound for our experimental setup. We term the models that are fine-tuned with this dataset WOZ_Large; (2) In a low resource setting on the small *vel_dev* set, which uses only a handful of dialogues. We term models fine-tuned with this other training set WOZ_Small. (3) In our proposed setting on our framework's synthetic dataset. We term models trained with this set as Velocidapter. Considering that our synthetic data is generated by templates extracted from the *vel_dev* set, this is a low resource scenario. Moreover, we also train a model version that also has its respective pre-trained versions on SQUAD, we add an "SQ" prefix to the name of each model to denote them: (1′)-SQ+WOZ_Large (2′)-SQ+Velocidapter (3′)-SQ+WOZ_Small.

The careful reader will note that the second and third settings are directly comparable, as they both utilise the same *vel_dev* dataset, but our framework multiply augments this initial dataset to a large volume of synthetic data.

We evaluate the performance of models on the MRCWOZ test set using the proposed F_1 and exact match (EM) metrics as in SQuAD (Rajpurkar et al., 2018b), using the official evaluation scripts provided.

4.1 Implementation Details

We use BERT-Base for the larger portion of our experiments. BERT-Base is a transformer-based language representation model pretrained in an unsupervised manner, often followed by finetuning in the target domain. Since our data is formatted following the SQuAD dataset, we use the official script provided by Devlin et al. (2019b) to train our

	Restaurant		Hotel		Taxi	
Training Setting	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**
High Resource						
WOZ_Large	97.99	97.78	94.99	94.63	99.78	99.35
SQ+WOZ_Large	97.27	96.51	97.27	96.51	98.18	97.43
Low Resource						
WOZ_Small	55.21	52.23	23.28	21.45	46.38	39.10
SQ+WOZ_Small	84.14	81.01	81.40	79.8	70.19	67.30
Velocidapter	70.46	66.77	80.45	78.54	64.24	62.17
SQ+Velocidapter	**84.40**	**81.70**	**85.22***	**84.85***	**83.42***	**81.40***
User Study						
SQ+Velocidapter	83.50	81.50	86.0*	84.80*	75.30*	70.0

(a) BERT-Base, all three domains.

	Restaurant	
Training Setting	**F1**	**EM**
High Resource		
WOZ_Large	97.93	97.46
SQ+WOZ_Large	98.02	97.46
Low Resource		
WOZ_Small	14.51	12.65
SQ+WOZ_Small	30.23	27.84
Velocidapter	22.93	21.20
SQ+Velocidapter	**36.15***	**31.64***

(b) BiDAF, restaurant domain.

Table 6: (a) Results of all three training settings on all three domains of the MRCWOZ dataset using the BERT-Base model, including the user study. (b) Results of all three training settings on the restaurant domain of the MRCWOZ dataset using the BiDAF model. Each result is an average of 5 runs. The first two rows show rich resource, upper bound results. The next 4 rows show low resource setting results. The last row in (a) is showing the results of training with novice templates from our user study. For each column, the upper-bound result is underlined and the best result in the low-resource setting is **bolded**. SQ+Velocidapter results are marked with an asterisk if significant when compared against SQ+WOZ_Small ($p < .05$).

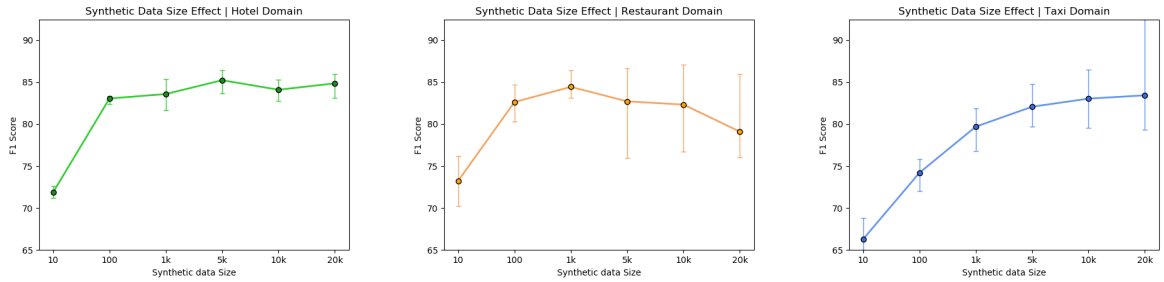

Figure 3: Plots showing synthetic data size effect in each domain: hotel, restaurant, taxi from left to right. The F_1 scores are averages over 5 different training sessions with 5 different synthetic datasets. The vertical ticks give a notion of experimental variance, denoting the maximum and minimum scores across the 5 runs.

model. During training we use the default hyperparameters that proved best in the original paper. We set the total number of steps for the original and synthetic training equal so that the their comparison stays fair.

To demonstrate that our framework is model-agnostic, we also demonstrate our technique on BiDAF (Seo et al., 2016). This is a hierarchical model that forms multiple levels of context representations using attention in both directions: context-to-query and query-to-context. During training, we again use the same hypermeter set that was facilitated within the paper and limit the training of both synthetic and original training to 20k steps.

4.2 Results

Table 6a gives the main results of our experiments. For the BERT-Base model, these suggest that both with and without pretraining, our framework outperforms other models in all three domains under the low-resource setting. The performance improvement introduced by our framework is larger in the taxi domain than the hotel and the restaurant domains. We believe this is due to the out of vocabulary (OOV) challenge being more significant in the former. Because our framework enriches the dialogue templates with diverse set of slot values, it addresses unseen vocabulary problem.

We repeat the restaurant domain experiments using the recurrent BiDAF model. Table 6b shows that the BiDAF model performs poorly in comparison to the BERT. This phenomenon parallels results on the SQuAD leaderboard where transformer-based models over-perform the recurrent BiDAF model by large (Rajpurkar et al., 2018a). Our framework is still able to boost the performance by a significant margin, showing that it works in a

model-agnostic manner.

4.3 Synthetic Data Size Effect

Similar to Liu *et al.* (2019), we find that the amount of synthetic data generated does not linearly benefit the model. To find the optimal amount for each domain, we set the size of the synthetic dataset as a hyperparameter during BiDAF experiments and plot the results in Figure 3. We hypothesize that the reason for the differing optima across domains is indicative of the coverage of the development sets from which we choose the dialogue turn templates. As these sets only have a few examples for each slot, they are not representative of the entire dataset. Although the augmentation process results in a more comprehensive set that improves the results, the synthetic data is (greatly) biased towards the examples in this small development set. When this bias becomes too pronounced through over-augmentation, we posit that the generalization of the model suffers. Hence, the synthetic data generation has still an ideal size that achieves optimal results in the low-resource setting, outperforming raw training over the development set.

Our analysis also points to the possibility of improvement by optimizing the choice of examples to cater for coverage and representativeness over the dataset's instance space. This can be achieved through a pipelined setting where the model directs the augmentation framework to create dialogues similar to which it shows low confidence on within the development set. We leave this as a field of study for future work.

4.4 Error Analysis in the Taxi Domain

We compare the two methods SQ+WOZ_Small and SQ+Velocidapter trained on BiDAF model qualitatively by analyzing errors made by the models on the taxi domain test set. We characterize the system errors to get a better sense of the overall causes (and potential solutions):

- **Partial value match** are errors that occur when the model predicts the slot only partially (an inexact match). An example is predicting the destination in the sentence "I want a ride to *Shanghai restaurant*" as "Shanghai" (partial) or "ride to Shanghai restaurant" (overshot).

- **Value mismatch** happens when the model predicts a value that is sound and appropriate for the given question but is not the ground-truth answer. This happens frequently by confusing destination and source places in the taxi domain; **Slot mismatch** is a common error where the model answers a question for one slot with another slot. Some observed patterns are replying with time when the question is asking for a place, and vice versa;

- **Former value match** occurs when the user states a value for some slot and then change their mind either in the same turn or in another upcoming turn and the model confuses the answer with the preceding value;

- **Overly long match**, this error type only happens within the SQ+WOZ_Small model, the prediction covers a very long span which takes up several turns;

- **Minor errors** (Rows 6–8) constitute the majority of the errors made by Velocidapter. These errors are small discrepancies from the ground truth such as punctuation, capitalization, or missing determiners.

- Finally, **Unrelated** errors occur when the answer provided by the system is unrelated to the question in any way.

Error type	**SQ + WOZ_Small**	**SQ + Velocidapter**
1. Partial value match	6	1
2. Value mismatch	28	7
3. Slot mismatch	7	5
4. Former value match	3	3
5. Overly long match	2	-
6*. Missing article "the"	3	21
7*. Capital letter mistakes	1	1
8*. Punctuation mistakes	1	9
9. Unrelated	3	2
Total	**54**	**49**

Table 7: Distribution of errors over error types made by the SQ+WOZ_Small and SQ+Velocidapter models in the taxi domain test set. Minor error types are marked with a star.

From Table 7, we see that Velocidapter significantly reduces the incidence of many major dialogue-specific errors (Rows 1–5), indicating that the dialogue structure is smoother. It is also evident that the biggest difference in performance is in value-based errors. This proves that enriching templates

with a diverse set of values increases model robustness. When we omit minor error types and run McNemar's test, the results indicate that Velocidapter shows statistically significant improvements over WOZ_Small with a 99% confidence level. We believe this is fair since such minor errors are less indicative of dialogue quality, and concern surface realization and inconsistencies in annotated slots. Including every error type in McNemar's test, the difference between the two systems becomes insignificant. We believe that further improvements to Velocidapter that may include additional language model (LM) smoothing may help address minor errors. LMs can also further diminish value-based errors by masking values with place holders and filling in with LM predictions, increasing the diversity of values.

4.5 User Study

Velocidapter's minimal dependence on human labor can be seen as an advantage or a disadvantage. We view our method as a means of providing a choice point to task-oriented dialogue systems designers that yields performance improvement with little manual investment. As we have argued that our framework is easy to replicate, we conduct a user study with two computer science graduate student participants who were aware of the nature of our research. Both students are not co-authors nor did they have any expertise in authoring dialogues. As training for the annotation process, we first narrated the written instructions[2], then performed a sample template construction with each subject. The subjects then followed the instructions to construct new templates from a few dialogues in a target domain, after which our team performed some post-formatting to facilitate the automation. The actual template construction took between 10–40 minutes, mostly dependent on the number of the dialogues being processed (*e.g.* 2 for taxi, 7 for restaurant). On average subjects generated 0.8 templates per minute. With these templates, Velocidapter leverages these starting few dialogues to create a training dataset 4 orders of magnitude larger.

Results of our user study correlate well with our experiments done using the author-generated data: our framework outperforms SQ+WOZ_Small substantially for hotel and taxi domains (*cf.* Table 6a, last row) at the 95% significance level, whereas the difference for restaurant results are not significant (observation and discussion in Section 4.2). Additionally, the participants reported more familiarity with the process on later domains, pointing towards further amortization of time cost.

[2]Instruction manuscript available at https://github.com/cuthalionn/Velocidapter.

5 Conclusion

In this work, we introduce a template-based augmentation framework for the task-oriented dialogue comprehension task. Our framework, Velocidapter, combines the two mutually beneficial concepts of synthetic data generation and domain adaptation to strategically utilize limited human input to greatly enrich sparse dialogue training data. Velocidapter leverages the turn-based nature of dialogue to strategically involve humans-in-the-loop to greatly reduce error in a robust fashion. It can be used to augment task-specific domain dialogues in the low-resource, few-shot setting by generating several orders of magnitude larger datasets, substantially decreasing dialogue-specific errors of a model (*e.g.* partial value match, value mismatch, *etc.*). This process only requires a little manual intervention: under an hour's time of a novice human creator for each new domain. Our experiments indicate that Velocidapter is a viable approach in addressing the data gap in comprehension of task-oriented dialogue systems. In the future, we look forward to using our framework on other task-oriented dialogue tasks. We further want to discover the automated extraction of dialogue chunks and generation of templates which can also benefit from controlled text generation techniques.

Acknowledgments

Work by the first author was supported by the SINGA scholarship, administered by A*STAR. We also want to acknowledge the assistance from Abhinav Ramesh Kashyap and Xinyuan Lu for participating in our user study reliability experiments.

References

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica S. Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on*

Empirical Methods in Natural Language Processing, pages 2174–2184, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. Bert: Pre-training of deep bidirectional transformers for language understanding.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech Language*, 59.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. *ArXiv*, abs/1706.09789.

Timothy J. Hazen, Shehzaad Dhuliawala, and Daniel Boies. 2019. Towards domain adaptation from limited data for question answering using deep neural networks. *ArXiv*, abs/1911.02655.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2018. DuReader: a Chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 37–46, Melbourne, Australia. Association for Computational Linguistics.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340.

F. Hill, Antoine Bordes, S. Chopra, and J. Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. *CoRR*, abs/1511.02301.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. arXiv preprint arXiv:1705.03551.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *CoRR*, abs/1712.07040.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. arXiv preprint arXiv:1704.04683.

Hongyu Li, Xiyuan Zhang, Yibing Liu, Yiming Zhang, Quan Wang, Xiangyang Zhou, Jing Liu, Hua Wu, and Haifeng Wang. 2019. D-NET: A pre-training and fine-tuning framework for improving the generalization of machine reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 212–219, Hong Kong, China. Association for Computational Linguistics.

Zhengyuan Liu, Hazel Lim, Nur Farah Ain Suhaimi, Shao Chuen Tong, Sharon Ong, Angela Ng, Sheldon Lee, Michael R, Macdonald, Savitha Ramasamy, Pavitra Krishnaswamy, Wai Leng Chow, and Nancy F. Chen. 2019. Fast prototyping a dialogue comprehension system for nurse-patient conversations on symptom monitoring. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 24—31, Minneapolis, Minnesota. Association for Computational Linguistics.

Kaixin Ma, Tomasz Jurczyk, and Jinho D. Choi. 2018. Challenging reading comprehension on daily conversation: Passage completion on multiparty dialog. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2039–2048, New Orleans, Louisiana. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. In *arXiv:2002.12328*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018a. Know what you don't know: Unanswerable questions for squad.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018b. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784—-789, Melbourne, Australia. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2020. Coqa: A conversational question answering challenge. In *Transactions of the Association for Computational Linguistics (Volume 7)*, pages 249–266.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2020. Dream: A challenge dataset and models for dialogue-based reading comprehension. In *Transactions of the Association for Computational Linguistics (Volume 7)*, pages 217–231.

Huazheng Wang, Zhe Gan, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, and Hongning Wang. 2019. Adversarial domain adaptation for machine reading comprehension. In *EMNLP/IJCNLP*.

Georg Wiese, Dirk Weissenborn, and Mariana Neves. 2017. Neural domain adaptation for biomedical question answering. *ArXiv*, abs/1706.03610.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, Brussels, Belgium. Association for Computational Linguistics.

Zhengzhe Yang and Jinho D. Choi. 2019. FriendsQA: Open-domain question answering on TV show transcripts. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 188–197, Stockholm, Sweden. Association for Computational Linguistics.

Rong Ye, Wenxian Shi, Hao Zhou, Zhongyu Wei, and Lei Li. 2020. Variational template machine for data-to-text generation. In *International Conference on Learning Representations*.

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

Helen Jiahe Zhao and Jiamou Liu. 2018. Finding answers from the word of god: Domain adaptation for neural networks in biblical question answering. *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.