Schema Augmentation for Zero-Shot Domain Adaptation in Dialogue State Tracking

Anonymous ACL submission

Abstract

Zero-shot domain adaptation for dialogue state tracking (DST) remains a challenging problem in task-oriented dialogue (TOD) systems, where models must generalize to target domains unseen at training time. Current large language model approaches for zero-shot domain adaptation rely on prompting to introduce knowledge pertaining to the target domains. However, their efficacy strongly depends on prompt engineering, as well as the zero-shot ability of the underlying language model. In this work, we devise a novel data augmentation approach, Schema Augmentation, that improves the zero-shot domain adaptation of language models through fine-tuning. Schema Augmentation is a simple but effective technique that enhances generalization by introducing variations of slot names within the schema provided in the prompt. Experiments on MultiWOZ and SpokenWOZ showed that the proposed approach resulted in a substantial improvement over the baseline, in some experiments achieving over a twofold accuracy gain over unseen domains while maintaining equal or superior performance over all domains.

1 Introduction

Dialogue State Tracking (DST) plays a pivotal role in task-oriented dialogue systems by maintaining a structured representation of the user's goals, intents, and preferences as a conversation progresses. As these systems interface with external APIs, such as booking platforms or food ordering services, accurate DST is crucial for ensuring successful goal-oriented interactions. Traditionally, DST models were domain-specific, relying on large amounts of annotated data for every domain. The rise of powerful multi-task instruction-tuned large language models (LLMs) has enabled zero-shot DST, which does not require task-specific training (Feng et al., 2023; Yi et al., 2024; Hosseini-Asl et al., 2020). However, zero-shot DST results have been mixed, and prior work has argued that general purpose models remain limited in their ability to replace specialized systems for this task (Heck et al., 2023).

Zero-shot domain adaptation offers a promising middle ground. In this problem, training is allowed, but only some domains are available at training time. This allows for methods that retain the benefits of fine-tuning while addressing the challenge of domain-specific data scarcity. Prior work has investigated this method, but has required slots to be filled one at a time, either by generating slot-specific prompts (Aksu et al., 2023) or reformulating slot-filling as QA (Li et al., 2021; Lin et al., 2021). However, zero-shot domain adaptation methods for end-to-end DST - modeling the entire dialogue state at once, given the dialogue context - is an area which remains underexplored. To this end, we focus on training methods for improving zero-shot domain adaptation with large language models for end-to-end DST. We develop a novel method using data augmentation techniques, and demonstrate the effectiveness of our method on two popular DST datasets, MultiWOZ 2.2 (Zang et al., 2020) and SpokenWOZ (Si et al., 2024).

Our contributions are threefold:

1. We introduce Schema Augmentation, a data augmentation technique for zero-shot domain adaptation that obtains up to a twofold improvement over a strong baseline.

2. We propose a new metric, Target Goal Accuracy, to evaluate domain adaptation performance in task-oriented dialogue systems.

3. We conduct ablation studies to analyze the factors contributing to the effectiveness of our method.

2 Problem Formulation

The goal of Dialogue State Tracking (DST) is to predict the dialogue state given the dialogue text

as input. Dialogue states are constructed from sets of slots $\mathcal{S}$ and values $\mathcal{V}$, with each slot belonging to a particular domain d. A dataset $\mathcal{X} = \{(x_i, y_i)\}$ consists of dialogues x_i and dialogue states y_i. A dialogue state is a set of k distinct slot/value pairs:

$$y_i = \{(s_{ik}, v_{ik})\}_k : s \in \mathcal{S}, v \in \mathcal{V}, \quad (1)$$

The goal is to learn a function that maps dialogue to dialogue state, $\pi(x) = y$. We also have a set of domains $\mathcal{D}$ and say that a dialogue state belongs to a set of domains, $y \in \mathcal{D}$, if $\exists\ (s_k, v_k) \in y : s_k \in d$ for some $d \in \mathcal{D}$. For zero-shot domain adaptation, we assume there is a set of target domains $\mathcal{D}_T$ from which no data is available during training. Thus we have a target subset of our test data

$$\mathcal{X}_{\text{target}} = \{(x, y) : y \in \mathcal{D}_T\} \quad (2)$$

For each slot $s \in \mathcal{S}$ we are given a description of the slot and a list of possible values it can take. This information, referred to as the *schema*, defines the structure of slot names, descriptions, and possible values, and is typically provided in the prompt.

3 Evaluation Metrics

For evaluation, we utilize Joint Goal Accuracy (JGA) (Henderson et al., 2014), the most commonly used DST metric. The main drawback of JGA is that it looks at all slots and domains, and thus does not measure domain adaptation performance directly. To alleviate this, we introduce a new metric: Target Goal Accuracy (TGA), a subgoal version of JGA that considers only those slots belonging to the target domains.

- **Joint Goal Accuracy (JGA):** The percent of turns in which the entire state is accurately predicted. We include all dialogue states, including empty states, to be comprehensive.
- **Target Goal Accuracy (TGA):** The percent of turns in which all slots belonging to the target domains are accurately predicted. We do not include turns that have no target-domain slots active in the ground truth dialogue state, as these empty states would dominate the overall metric.

Formally, we define these metrics in terms of hits over a dataset $\mathcal{X}$, i.e samples for which the model predicted the correct overall state:

$$\mathcal{H} = \{(x, y) \in \mathcal{X} :\ \pi(x) = y\} \quad (3)$$

$$\text{JGA} = \frac{|\mathcal{H}|}{|\mathcal{X}|} \quad (4)$$

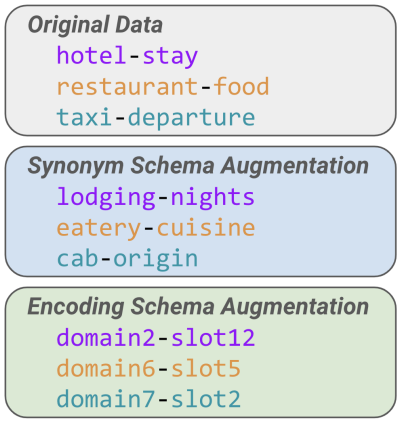

Figure 1: Examples of domain and slot replacements for both Schema Augmentation types: SSA (blue box) and ESA (green box). A full example of augmented prompts can be found in Appendix

For TGA we are only interested in the target subset:

$$\mathcal{H}_T = \{(x, y) \in \mathcal{X}_{\text{target}} :\ \pi(x) = y\} \quad (5)$$

$$\text{TGA} = \frac{|\mathcal{H}_T|}{|\mathcal{X}_{\text{target}}|} \quad (6)$$

JGA gives us a standardized measure of overall performance on the DST task, while TGA highlights domain adaptation abilities of each method. Our goal is to improve TGA without degradation of JGA.

4 Our Approach: Schema Augmentation

Domain adaptation is often addressed by leveraging target domain data for fine-tuning (Dingliwal et al., 2021; Zeng et al., 2021) or as few-shot examples during inference (Yang et al., 2023; Chen et al., 2023; Rastogi et al., 2019; Hu et al., 2022). However, reliably annotated data for unseen domains is often unavailable. To address this, we propose to achieve zero-shot adaptation through data augmentation. New domains differ from existing ones in their labeling schema and dialogues. Large language models (LLMs), pre-trained on vast amounts of natural language, are likely robust to variations in dialogue, so we focus on differences in schema.

Our method, *Schema Augmentation*, generates augmented data by altering domain and slot names to enhance the model's robustness to schema changes. We introduce two types of Schema Augmentation. First, in **Synonym Schema Augmentation (SSA)**, we replace domain and slot names with synonyms. Second, in **Encoding Schema Augmentation (ESA)**, we replace domains and slots

with non-semantic codes, requiring the model to rely on slot descriptions and possible values to predict the dialogue state accurately. Figure 1 shows examples of each method.

For both types of augmentation, we experiment with two variations: single and multi. In single-SSA and single-ESA, we use a static synonym or code replacement for each domain and slot. In multi-SSA and multi-ESA, we randomly select the replacement for each sample from a pre-determined list. Gemini (Gemini Team, 2023) provides the synonym lists, while we create encodings by appending integers to 'domain' or 'slot.' In multi-ESA, we ensure non-overlapping encodings for each domain and slot by using disjoint ranges of integers. The replacements used in both cases are provided in Appendix A.

5 Experimental Setup

5.1 Modeling

We perform dialogue state tracking end-to-end using large language models (LLMs). Building on LDST (Feng et al., 2023), our prompts include an instruction, DST schema, and dialogue. We represent dialogue state outputs as textual JSON, and during evaluations we parse the output generations using regular expressions (`regex`). Any response that does not contain valid JSON is considered incorrect by all metrics. Details of our experimental hyperparameters and compute are included in Appendix A.

We choose state-of-the-art instruction-tuned models from two popular families, Gemma (Gemma Team, 2024) and Mistral (Jiang et al., 2023). We use the variants that are most recent at the time of writing: `gemma-2-9b-t` and `Mistral-7B-Instruct-v0.3`.

For comparison with prior work, we follow Feng et al. (2023) and build a prompt consisting of instructions, schema, and dialogue. The schema includes slot names, descriptions, and possible values. See Appendix A for full examples of our prompts.

5.2 Datasets

We conduct our experiments on two open-source dialogue state tracking datasets. **MultiWOZ** (Budzianowski et al., 2018) is a multi-domain task-oriented dialogue dataset comprising annotated dialogues across eight domains including hotel booking, restaurant reservation, and taxi ordering. Each dialogue is annotated with the dialogue state at each turn. To alleviate dataset noise, several updated versions of MultiWOZ have since been released. We use MultiWOZ 2.2 (Zang et al., 2020), a popular, high quality version of the original. Additionally, we use **SpokenWOZ** (Si et al., 2024), a spoken dialogue TOD dataset inspired by MultiWOZ. SpokenWOZ dialogues were collected from crowdworkers engaging in spoken conversations and includes text transcriptions from an automatic speech recognition (ASR) system. We perform our experiments on the audio transcriptions.

In order to best simulate the domain adaptation scenario, we select holdout domains that minimize slot overlap with the training domains. To this end, we choose `{taxi,train,bus}` as our holdout domains. Tables showing slot overlap are included in Appendix A.

6 Results

Our results are shown in Table 1. We compare our method to several baselines: (0) Zero-shot prompting using LDST prompt from Feng et al. (2023); (1) Standard fine-tuning with no schema, where the input is only the dialogue text; and (2) Fine-tuning using the LDST schema. These baselines are measured against our four methods on both datasets for both models studied. We observe strong performance across the board with both variations of SSA and ESA for both methods, with the strongest performance being achieved with multi-ESA, which boosts TGA from 18.2% to 40.7% on MultiWOZ with Gemma-2.

7 Ablation Studies

Our results in Table 1 raise the following questions:

1. Why does schema augmentation (both SSA and ESA) improve domain adaptation?

2. Why does ESA improve adaptation more than SSA?

To shed light on these questions, we perform ablation studies by shuffling descriptions and possible values at test time and observing the performance impact. This ablation tests the dependence of each method on the information content of the descriptions and values, without introducing confounding variables related to mismatch between train and test samples.

Figure 2 shows the ablation results for Gemma-2 on MultiWOZ 2.2. To answer the Question 1, with

XID	Method	Model	MultiWOZ 2.2 JGA%	MultiWOZ 2.2 TGA%	SpokenWOZ JGA%	SpokenWOZ TGA%
0	Zero-shot prompting (LDST schema)	Mistral	8.1	7.2	10.2	6.8
		Gemma-2	13.8	16.5	8.2	0.2
1	Fine-tuning, no schema (dialogues only)	Mistral	29.7	0.5	24.6	0.3
		Gemma-2	31.1	0.4	25.4	1.2
2	LDST schema[1] (no augmentation)	Mistral	37.2	15.2	26.9	8.1
		Gemma-2	37.9	18.2	32.4	25.7
3	Single-SSA (Our method)	Mistral	35.5	15.1	30.9	19.3
		Gemma-2	42.1	31.0	32.5	24.2
4	Multi-SSA (Our method)	Mistral	39.6	24.9	27.3	12.5
		Gemma-2	*42.8*	34.5	30.6	28.8
5	Single-ESA (Our method)	Mistral	38.7	25.0	29.3	14.3
		Gemma-2	42.7	*36.3*	*33.7*	*29.4*
6	Multi-ESA (Our method)	Mistral	40.4	26.0	29.4	15.4
		Gemma-2	**45.2**	**40.7**	**34.1**	**30.3**

Table 1: Results for baselines and schema augmentation (out method). We use Huggingface implementations of `gemma-2-9b-it` and `Mistral-7B-Instruct-v0.3`. Each experiment is given an experiment ID (XID) for later reference. All experiments use the LDST schema (Feng et al., 2023) except the no schema case (XID 1).

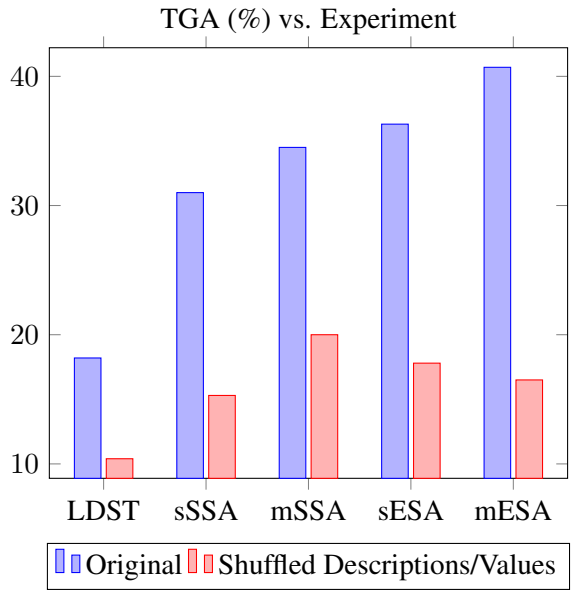

Figure 2: Ablation results for Gemma-2 on MultiWOZ 2.2. Experiment names are abbreviated. LDST=no augmentation; single/multi are shortened to s/m.

description/value shuffling, we observe a greater drop in TGA for Schema Augmentation (min: -14.5%, max: -24.2%) than for the LDST baseline (-7.8%). This suggests that Schema Augmentation encourages the model to pay more attention to the descriptions and values of the slots. This becomes particularly useful for target domain samples for which the model has not seen any training data. Additionally, we observe that TGA drop is greater for the ESA (sESA: -18.5%, mESA: -24.2%) than for SSA (sSSA: -15.7%, mSSA: -14.5%), suggesting that ESA more strongly encourages the model to attend to descriptions and values provided in the schema.

8 Conclusion

We developed Schema Augmentation, a data augmentation technique for zero-shot domain adaptation in dialogue state tracking. To assess its effectiveness, we introduced Target Goal Accuracy (TGA), a metric that evaluates performance specifically on unseen target domains. Our method significantly boosts TGA compared to baseline approaches across two widely-used datasets, demonstrating robustness to out-of-distribution information in prompts during test time. Ablation studies further illuminate the key factors contributing to these improvements. These results underscore the potential of Schema Augmentation as a simple yet powerful tool for enhancing domain adaptation in task-oriented dialogue systems. We recommend future work to explore similar augmentation strategies to improve generalization to unseen domains in other applications.

[1] We re-implemented the LDST prompting method using our models.

Limitations

Our work has several limitations. We evaluated Schema Augmentation on two instruction-tuned models, Gemma and Mistral, but did not explore a broader range of models, so the generalizability of our findings to other architectures is unclear. Additionally, while we tested on two task-oriented dialogue datasets (MultiWOZ and SpokenWOZ), both are based on similar domains, and further testing on more diverse datasets is needed. Additionally, we only used one set of holdout domains, whereas our experiments could be repeated using different sets of the available domains as holdouts. Due to compute constraints, we limited fine-tuning to models with fewer than 10 billion parameters, which may affect performance compared to larger models. Moreover, our experiments were confined to English-language datasets, leaving the effectiveness of Schema Augmentation in multilingual or non-English contexts unexplored. Lastly, the scope of hyperparameter tuning was limited by available resources, and further exploration of fine-tuning configurations could yield even more insights.

Ethics Statement

This work aims to improve dialogue state tracking in task-oriented systems, with potential applications in real-world settings like customer service or healthcare. Ensuring the fairness and robustness of these models is crucial to avoid biased or harmful outcomes, especially for underrepresented groups. Additionally, while our method enhances model performance in unseen domains, careful consideration is required before deploying such models in sensitive areas where errors could have significant consequences. Finally, the environmental impact of training large models is an important factor, and more sustainable practices in AI research should be prioritized.

References

Taha Aksu, Min-Yen Kan, and Nancy F Chen. 2023. Prompter: Zero-shot adaptive prefixes for dialogue state tracking domain adaptation. *arXiv preprint arXiv:2306.04724*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Derek Chen, Kun Qian, and Zhou Yu. 2023. Stabilized in-context learning with pre-trained language models for few shot dialogue state tracking. *arXiv preprint arXiv:2302.05932*.

Saket Dingliwal, Bill Gao, Sanchit Agarwal, Chien-Wei Lin, Tagyoung Chung, and Dilek Hakkani-Tur. 2021. Few shot dialogue state tracking using meta-learning. *arXiv preprint arXiv:2101.06779*.

Yujie Feng, Zexin Lu, Bo Liu, Liming Zhan, and Xiao-Ming Wu. 2023. Towards llm-driven dialogue state tracking. *arXiv preprint arXiv:2310.14970*.

Gemini Team. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemma Team. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Michael Heck, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, Shutong Feng, Christian Geishauser, Hsien-Chin Lin, Carel van Niekerk, and Milica Gašić. 2023. Chatgpt for zero-shot dialogue state tracking: A solution or an opportunity? *arXiv preprint arXiv:2306.01386*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)*, pages 263–272.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *Advances in Neural Information Processing Systems*, 33:20179–20191.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A Smith, and Mari Ostendorf. 2022. In-context learning for few-shot dialogue state tracking. *arXiv preprint arXiv:2203.08568*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Shuyang Li, Jin Cao, Mukund Sridhar, Henghui Zhu, Shang-Wen Li, Wael Hamza, and Julian McAuley. 2021. Zero-shot generalization in dialog state tracking through generative question answering. *arXiv preprint arXiv:2101.08333*.

Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, et al. 2021. Zero-shot dialogue state tracking via cross-task transfer. *arXiv preprint arXiv:2109.04655*.

Pushpendre Rastogi, Arpit Gupta, Tongfei Chen, and Lambert Mathias. 2019. Scaling multi-domain dialogue state tracking via query reformulation. *arXiv preprint arXiv:1903.05164*.

Shuzheng Si, Wentao Ma, Haoyu Gao, Yuchuan Wu, Ting-En Lin, Yinpei Dai, Hangyu Li, Rui Yan, Fei Huang, and Yongbin Li. 2024. Spokenwoz: A large-scale speech-text benchmark for spoken task-oriented dialogue agents. *Advances in Neural Information Processing Systems*, 36.

Yuting Yang, Wenqiang Lei, Pei Huang, Juan Cao, Jintao Li, and Tat-Seng Chua. 2023. A dual prompt learning framework for few-shot dialogue state tracking. In *Proceedings of the ACM Web Conference 2023*, pages 1468–1477.

Zihao Yi, Jiarui Ouyang, Yuwen Liu, Tianhao Liao, Zhe Xu, and Ying Shen. 2024. A survey on recent advances in llm-based multi-turn dialogue systems. *arXiv preprint arXiv:2402.18013*.

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. *arXiv preprint arXiv:2007.12720*.

Jiali Zeng, Yongjing Yin, Yang Liu, Yubin Ge, and Jinsong Su. 2021. Domain adaptive meta-learning for dialogue state tracking. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2493–2501.

A Appendix

A.1 Hyperparameters

For our experiments, we fine-tune both our models using the AdamW optimizer with a learning rate of 2e-4 and warmup ratio of 0.03. Due to compute constraints, we use LoRA (Hu et al., 2021) to train adapters while keeping base weights frozen. We use a LoRA $r = 2$ and $\alpha = 2$ with a dropout of 0, and adapter weights added to all linear layers. In all experiments, the modes is fine-tuned to convergence in each experiment. We achieve this by evaluating on the validation split each epoch and choosing an early stopping patience of 1. This ensures that each experiment yields the best model and comparisons between methods are fair.

All experiments use a random seed of 42 and deterministic algorithms were used everywhere possible to ensure minimal variation between runs. All accuracy metrics reported had less than 1% variance across all runs.

A.2 Compute

All fine-tuning and inference was run on Nvidia A40 GPUs with 48GB GDDR6 memory. Fine-tuning took 1-2 hours on 8 GPUs in parallel with pytorch distributed data parallel (DDP).

A.3 Slots and Domains

Tables 2 and 3 show the domain and slot combinations for the two datasets. Taxi, train, and bus were chosen as holdout domains due to many slots in common with each other and few slots in common with other domains.

A.4 Replacements

The full list of synonym and encoding replacements for slots and values are shown in Figures 3 and 4.

A.5 Schema Augmentation Examples

Full examples of the original prompt, SSA prompt, and ESA prompt for an example from MultiWOZ 2.2 are shown in Figures 5, 6, and 7.

A.6 Licenses

All datasets and models used are free and open-source. MultwiWOZ and SpokenWOZ are available under MIT license. Gemma-2 and Mistral are available under the Apache 2.0 license. All models and datasets are intended for reseach purposes, which is consistent with this work. Datasets are widely used and reported not to contain personal information or offensive content.

	attraction	bus	hospital	hotel	police	restaurant	taxi	train
address	✓		✓	✓	✓	✓		
area	✓			✓		✓		
arriveby							✓	✓
bookday				✓		✓		
bookpeople				✓		✓		✓
bookstay				✓				
booktime						✓		
day		✓						✓
department			✓					
departure		✓					✓	✓
destination		✓					✓	✓
duration								✓
entrancefee	✓							
food						✓		
internet				✓				
leaveat		✓					✓	✓
name	✓			✓	✓	✓		
openhours	✓							
parking				✓				
phone	✓		✓	✓	✓	✓	✓	
postcode	✓		✓	✓	✓	✓		
price								✓
pricerange				✓		✓		
ref				✓		✓		✓
stars				✓				
trainid								✓
type	✓			✓			✓	

Table 2: Domain-Slot Combinations for MultiWOZ.

```
    {
    "slots": {
        "pricerange": ["cost_range", "expense_range", "price_level"],
        "type": ["category", "classification", "kind"],
        "parking": ["parking"],
        "day": ["day", "bookday"],
        "bookday": ["day", "bookday"],
        "people": ["guests", "individuals", "persons"],
        "bookpeople": ["guests", "individuals", "persons"],
        "stay": ["nights", "duration", "length_of_visit"],
        "bookstay": ["nights", "duration", "length_of_visit"],
        "internet": ["wifi", "online_access", "broadband"],
        "name": ["name"],
        "area": ["location", "region", "zone", "neighborhood", "district"],
        "stars": ["rating", "grade"],
        "arriveby": ["arrival_time", "scheduled_arrival"],
        "leaveat": ["departure_time"],
        "destination": ["arrival_point", "end_point", "final_stop"],
        "departure": ["origin", "start_point", "beginning_location"],
        "food": ["cuisine", "style", "type"],
        "time": ["reservation", "slot"],
        "booktime": ["reservation", "slot"],
        "department": ["section", "division", "unit"],
        "email": ["email_address"],
        "idnumber": ["ID", "identification_number"],
        "phone": ["phone", "cell_number", "mobile_number"],
        "phonenumber": ["phone", "cell_number", "mobile_number"],
        "platenumber": ["license_plate", "plate"],
        "address": ["street_number"],
        "postcode": ["zipcode", "postal_code"],
        "ref": ["booking_number", "confirmation_number", "reservation_number"],
        "entrancefee": ["entry_fee", "admission_fee"],
        "openhours": ["business_hours", "hours_of_operation", "schedule"]
    },
    "domains": {
        "hotel": ["lodging", "accommodation", "motel"],
        "train": ["rail"],
        "attraction": ["sight", "landmark", "tourist_spot"],
        "restaurant": ["eatery", "diner", "cafe", "bistro", "food_place"],
        "hospital": ["medical_center", "health_facility"],
        "taxi": ["cab", "car", "uber", "lyft"],
        "profile": ["user", "account"],
        "police": ["cops", "law_enforcement"]
    }
}
```

Figure 3: Domain and slot synonym replacements for SSA.

```
{
    "slots": {
        "pricerange": ["slot010", "slot011", "slot012", "slot013", "slot014"]
        "type": ["slot020", "slot021", "slot022", "slot023", "slot024"]
        "parking": ["slot030", "slot031", "slot032", "slot033", "slot034"]
        "day": ["slot040", "slot041", "slot042", "slot043", "slot044"]
        "bookday": ["slot050", "slot051", "slot052", "slot053", "slot054"]
        "people": ["slot060", "slot061", "slot062", "slot063", "slot064"],
        "bookpeople": ["slot070", "slot071", "slot072", "slot073", "slot074"],
        "stay": ["slot080", "slot081", "slot082", "slot083", "slot084"],
        "bookstay": ["slot090", "slot091", "slot092", "slot093", "slot094"],
        "internet": ["slot100", "slot101", "slot102", "slot103", "slot104"],
        "name": ["slot110", "slot111", "slot112", "slot113", "slot114"],
        "area": ["slot120", "slot121", "slot122", "slot123", "slot124"],
        "stars": ["slot130", "slot131", "slot132", "slot133", "slot134"],
        "arriveby": ["slot140", "slot141", "slot142", "slot143", "slot144"],
        "leaveat": ["slot150", "slot151", "slot152", "slot153", "slot154"],
        "destination": ["slot160", "slot161", "slot162", "slot163", "slot164"],
        "departure": ["slot170", "slot171", "slot172", "slot173", "slot174"],
        "food": ["slot180", "slot181", "slot182", "slot183", "slot184"],
        "time": ["slot190", "slot191", "slot192", "slot193", "slot194"],
        "booktime": ["slot200", "slot201", "slot202", "slot203", "slot204"],
        "department": ["slot210", "slot211", "slot212", "slot213", "slot214"],
        "email": ["slot220", "slot221", "slot222", "slot223", "slot224"],
        "idnumber": ["slot230", "slot231", "slot232", "slot233", "slot234"],
        "phone": ["slot240", "slot241", "slot242", "slot243", "slot244"],
        "phonenumber": ["slot250", "slot251", "slot252", "slot253", "slot254"],
        "platenumber": ["slot260", "slot261", "slot262", "slot263", "slot264"],
        "address": ["slot270", "slot271", "slot272", "slot273", "slot274"],
        "postcode": ["slot280", "slot281", "slot282", "slot283", "slot284"],
        "ref": ["slot290", "slot291", "slot292", "slot293", "slot294"],
        "entrancefee": ["slot300", "slot301", "slot302", "slot303", "slot304"],
        "openhours": ["slot310", "slot311", "slot312", "slot313", "slot314"]
    },
    "domains": {
        "attraction": ["domain10", "domain11", "domain12", "domain13", "domain14"],
        "hotel": ["domain20", "domain21", "domain22", "domain23", "domain24"],
        "hospital": ["domain30", "domain31", "domain32", "domain33", "domain34"],
        "police": ["domain40", "domain41", "domain42", "domain43", "domain44"],
        "profile": ["domain50", "domain51", "domain52", "domain53", "domain54"],
        "restaurant": ["domain60", "domain61", "domain62", "domain63", "domain64"],
        "taxi": ["domain70", "domain71", "domain72", "domain73", "domain74"],
        "train": ["domain80", "domain81", "domain82", "domain83", "domain84"]
    }
}
```


Figure 4: Domain and slot synonym replacements for ESA.

```
### Schema:
- Slot: hotel-pricerange; Description: price budget of the hotel; Possible values:
↪  ['expensive', 'cheap', 'moderate']
- Slot: hotel-type; Description: what is the type of the hotel; Possible values:
↪  ['guesthouse', 'hotel']
- Slot: hotel-parking; Description: whether the hotel has parking; Possible values:
↪  ['free', 'no', 'yes']
- Slot: hotel-bookday; Description: day of the hotel booking; Possible values:
↪  ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday']
- Slot: hotel-bookpeople; Description: number of people for the hotel booking;
↪  Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: hotel-bookstay; Description: length of stay at the hotel; Possible values:
↪  ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: hotel-stars; Description: star rating of the hotel; Possible values: ['0',
↪  '1', '2', '3', '4', '5']
- Slot: hotel-internet; Description: whether the hotel has internet; Possible
↪  values: ['free', 'no', 'yes']
- Slot: hotel-name; Description: name of the hotel; Possible values: []
- Slot: hotel-area; Description: area or place of the hotel; Possible values:
↪  ['centre', 'east', 'north', 'south', 'west']
- Slot: hotel-address; Description: address of the hotel; Possible values: []
- Slot: hotel-phone; Description: phone number of the hotel; Possible values: []
- Slot: hotel-postcode; Description: postal code of the hotel; Possible values: []
- Slot: hotel-ref; Description: reference number of the hotel booking; Possible
↪  values: []
- Slot: restaurant-pricerange; Description: price budget for the restaurant;
↪  Possible values: ['cheap', 'expensive', 'moderate']
- Slot: restaurant-area; Description: area or place of the restaurant; Possible
↪  values: ['centre', 'east', 'north', 'south', 'west']
- Slot: restaurant-food; Description: the cuisine of the restaurant you are looking
↪  for; Possible values: []
- Slot: restaurant-name; Description: name of the restaurant; Possible values: []
- Slot: restaurant-bookday; Description: day of the restaurant booking; Possible
↪  values: ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday',
↪  'sunday']
- Slot: restaurant-bookpeople; Description: how many people for the restaurant
↪  reservation; Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: restaurant-booktime; Description: time of the restaurant booking; Possible
↪  values: []
- Slot: restaurant-address; Description: address of the restaurant; Possible
↪  values: []
- Slot: restaurant-phone; Description: phone number of the restaurant; Possible
↪  values: []
- Slot: restaurant-postcode; Description: postal code of the restaurant; Possible
↪  values: []
- Slot: restaurant-ref; Description: reference number of the restaurant booking;
↪  Possible values: []
```

Figure 5: Original data sample schema

```
### Schema:
- Slot: lodging-cost_range; Description: price budget of the hotel; Possible values:
↪  ['expensive', 'cheap', 'moderate']
- Slot: lodging-category; Description: what is the type of the hotel; Possible
↪  values: ['guesthouse', 'hotel']
- Slot: lodging-parking; Description: whether the hotel has parking; Possible
↪  values: ['free', 'no', 'yes']
- Slot: lodging-day; Description: day of the hotel booking; Possible values:
↪  ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday']
- Slot: lodging-guests; Description: number of people for the hotel booking;
↪  Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: lodging-nights; Description: length of stay at the hotel; Possible values:
↪  ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: lodging-rating; Description: star rating of the hotel; Possible values:
↪  ['0', '1', '2', '3', '4', '5']
- Slot: lodging-wifi; Description: whether the hotel has internet; Possible values:
↪  ['free', 'no', 'yes']
- Slot: lodging-name; Description: name of the hotel; Possible values: []
- Slot: lodging-location; Description: area or place of the hotel; Possible values:
↪  ['centre', 'east', 'north', 'south', 'west']
- Slot: lodging-street_number; Description: address of the hotel; Possible values:
↪  []
- Slot: lodging-phone; Description: phone number of the hotel; Possible values: []
- Slot: lodging-zipcode; Description: postal code of the hotel; Possible values: []
- Slot: lodging-booking_number; Description: reference number of the hotel booking;
↪  Possible values: []
- Slot: eatery-cost_range; Description: price budget for the restaurant; Possible
↪  values: ['cheap', 'expensive', 'moderate']
- Slot: eatery-location; Description: area or place of the restaurant; Possible
↪  values: ['centre', 'east', 'north', 'south', 'west']
- Slot: eatery-cuisine; Description: the cuisine of the restaurant you are looking
↪  for; Possible values: []
- Slot: eatery-name; Description: name of the restaurant; Possible values: []
- Slot: eatery-day; Description: day of the restaurant booking; Possible values:
↪  ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday']
- Slot: eatery-guests; Description: how many people for the restaurant reservation;
↪  Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: eatery-reservation; Description: time of the restaurant booking; Possible
↪  values: []
- Slot: eatery-street_number; Description: address of the restaurant; Possible
↪  values: []
- Slot: eatery-phone; Description: phone number of the restaurant; Possible values:
↪  []
- Slot: eatery-zipcode; Description: postal code of the restaurant; Possible values:
↪  []
- Slot: eatery-booking_number; Description: reference number of the restaurant
↪  booking; Possible values: []
```

Figure 6: SSA schema

```
### Schema:
- Slot: domain2-slot1; Description: price budget of the hotel; Possible values:
↪  ['expensive', 'cheap', 'moderate']
- Slot: domain2-slot2; Description: what is the type of the hotel; Possible values:
↪  ['guesthouse', 'hotel']
- Slot: domain2-slot3; Description: whether the hotel has parking; Possible values:
↪  ['free', 'no', 'yes']
- Slot: domain2-slot5; Description: day of the hotel booking; Possible values:
↪  ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday']
- Slot: domain2-slot7; Description: number of people for the hotel booking;
↪  Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: domain2-slot9; Description: length of stay at the hotel; Possible values:
↪  ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: domain2-slot13; Description: star rating of the hotel; Possible values:
↪  ['0', '1', '2', '3', '4', '5']
- Slot: domain2-slot10; Description: whether the hotel has internet; Possible
↪  values: ['free', 'no', 'yes']
- Slot: domain2-slot11; Description: name of the hotel; Possible values: []
- Slot: domain2-slot12; Description: area or place of the hotel; Possible values:
↪  ['centre', 'east', 'north', 'south', 'west']
- Slot: domain2-slot27; Description: address of the hotel; Possible values: []
- Slot: domain2-slot24; Description: phone number of the hotel; Possible values: []
- Slot: domain2-slot28; Description: postal code of the hotel; Possible values: []
- Slot: domain2-slot29; Description: reference number of the hotel booking;
↪  Possible values: []
- Slot: domain6-slot1; Description: price budget for the restaurant; Possible
↪  values: ['cheap', 'expensive', 'moderate']
- Slot: domain6-slot12; Description: area or place of the restaurant; Possible
↪  values: ['centre', 'east', 'north', 'south', 'west']
- Slot: domain6-slot18; Description: the cuisine of the restaurant you are looking
↪  for; Possible values: []
- Slot: domain6-slot11; Description: name of the restaurant; Possible values: []
- Slot: domain6-slot5; Description: day of the restaurant booking; Possible values:
↪  ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday']
- Slot: domain6-slot7; Description: how many people for the restaurant reservation;
↪  Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: domain6-slot20; Description: time of the restaurant booking; Possible
↪  values: []
- Slot: domain6-slot27; Description: address of the restaurant; Possible values: []
- Slot: domain6-slot24; Description: phone number of the restaurant; Possible
↪  values: []
- Slot: domain6-slot28; Description: postal code of the restaurant; Possible values:
↪  []
- Slot: domain6-slot29; Description: reference number of the restaurant booking;
↪  Possible values: []
```

Figure 7: ESA schema

	attraction	**hospital**	**hotel**	**profile**	**restaurant**	**taxi**	**train**
area	✓		✓		✓		
arriveby						✓	✓
day			✓		✓		✓
department		✓					
departure						✓	✓
destination						✓	✓
email				✓			
food					✓		
idnumber				✓			
internet			✓				
leaveat						✓	✓
name	✓		✓	✓	✓		
parking			✓				
people			✓		✓		✓
phonenumber				✓			
platenumber				✓			
pricerange			✓				
stars			✓				
stay			✓				
time					✓		
type	✓		✓				

Table 3: Domain-Slot Combinations for SpokenWOZ.