Analysis on Knowledge Aware Graph Neural Network for Dialogue State Tracking Across Specific Domains

Norman Wen
normanqw@umich.edu

Zhixiang Teoh
zhteoh@umich.edu

Alex Li
xelail@umich.edu

Abstract

Dialogue state tracking is essential and useful in building today's dialogue systems by helping to extract useful information about a dialogue, especially from user utterances. The 2021 novel hybrid Knowledge-Aware Graph-Enhanced GPT-2 (KAGE-GPT2) architecture augments GPT-2 with cross-domain inter-slot relationships and dependencies learned from Graph Attention Networks that could otherwise be lost in sequential prediction. By nature, the MultiWOZ dialogue state tracking dataset is a multi-domain dataset. KAGE-GPT2 was reported to have improvements in dialogue state tracking performance in MultiWOZ 2.0 against strong baseline models. In this paper, we evaluate the strong KAGE-GPT2 novel hybrid model on specific individual target domains in MultiWOZ and analyze the results against that obtained from evaluating the model on the multi-domain problem. Since KAGE-GPT2 was trained and evaluated on MultiWOZ 2.0, which has since been shown to have errors and substantial noise, we also compare the results of model evaluation on MultiWOZ 2.1, an updated version of the dataset that addressed these errors and noise.

1 Introduction

In a dialogue, there is a large amount of information being exchanged in a single sentence. When a user utters a sentence such as, "There is a restaurant called No Thai near State Street that sells meals for $10 to $12," we can glean a lot of information, namely entity attributes called "slots" (Budzianowski et al., 2018), from this—such as the *restaurant name*, the *restaurant location*, and the *price range* of meals. This goes for any sentence in the domain of restaurants. In general, we would like a dialogue system to be able to keep track of critical slot-value pairs such as the ones defined above. A slot is defined to be an entity attribute. We call our collection of slots our ontology. For the single domain problem, a dialogue state for that utterance is defined as a set of (slot, tuple) pairs. For the example utterance, the dialogue state is given by (*restaurant name*, No Thai), (*restaurant location*, State Street), (*price range*, $10 to $12). For the multi-domain problem, we track the domain associated with each dialogue state. We define the multi-domain static-ontology dialogue state tracking problem as follows: given a user-system dialogue of user and system utterances and a static ontology, output the dialogue state—a set of (domain, slot, value) tuple—for each user utterance. Dialogue state tracking is beneficial for building multi-domain task-oriented dialogue systems, for example, generating system utterances in response to user utterances.

GPT-2 augmented with relational (Lin et al., 2021) representations derived from Graph Attention Networks have been shown to produce high joint[1] (54.86%) and slot[2] (97.47%) accuracy on the MultiWOZ 2.0 dataset (Lin et al., 2021), by building on Dialogue State Tracking via Knowledge-Aware Graph Enhanced Question Answering (Zhou and Small, 2019) and addressing the limitations in accurately predicting slot values that occur early on arising from GPT-2's causal-based modelling (Lin et al., 2021).

Throughout this project, we attempted to implement several methods to improve the KAGE-GPT2 model with varying degrees of success. Furthermore, we analyzed the performance of Lin et al's pre-trained KAGE-GPT2 model to investigate dialogue-domain specific performance and cross-dialogue domain performance on the newer[3] Multi-

[1] *Slot Accuracy* measures the ratio of successful slot value predictions among all the slots of each dialogue turn in ground-truth (Lin et al., 2021).

[2] *Joint Goal Accuracy* compares the predicted belief state to the ground truth at every dialogue turn. The output is considered correct only if all the predicted slot values exactly match the ground truth values (Lin et al., 2021).

[3] Compared to MultiWOZ 2.0. At the time of writing, the

WOZ 2.1 dataset.

2 Related work

2.1 Slot-Utterance Matching for Universal and Scalable Belief Tracker

Lee et al. developed a universal and scalable belief tracker wherein one single belief tracker can serve to handle any domain and slot type. They named their solution Slot-Utterance Matching for Universal and Scalable Belief Tracker or SUMBT for short. SUMBT first encodes system and user utterances pairs using BERT as a contextual semantics encoder. SUMBT then uses multi-head attention for the attention mechanism to retrieve relevant information corresponding to the domain-slot-type from the utterances. Finally, as this model deals with turn-level predictions, the model needs to incorporate previous belief states into generating the current new belief states. The authors incorporate an RNN whose inputs are the aforementioned output from the attention layer and the previous belief states, and the output of this RNN is a vector that is fed through a normalization layer, and whose final output is close to the target slot values semantics vector.

The authors trained and tested SUMBT on WOZ 2.0 corpus, yielding a joint accuracy of 0.910, which surpassed the baseline methods: `BERT+RNN`, a model without a contextual encoding layer, and `BERT+RNN+Ontology` which takes advantage of an ontology-utterance matching network that performs element-wise multiplications between the encoded ontology and utterances.

2.2 Knowledge-Aware Graph-Enhanced GPT-2 (KAGE-GPT2)

KAGE-GPT2 is a hybrid model inspired by the graph-based approach of Dynamic Knowledge Graph-Enhanced Dialogue State Tracking Question and Answering (DSTQA) that employs a dynamically-evolving knowledge graph to learn relationships between (domain, slot) pairs explicitly. The model takes a three-step approach at each user utterance turn: (1) pass the dialogue history and a serialization of the static ontology (as a string of (slot, <placeholder>) pairs) to GPT-2 to generate features for all possible domain-slots and values in the static ontology; (2) feed the resultant features into a Graph Attention Network (GAT) to learn relationships between (domain, slot) pairs and values similar to DSTQA; and (3) feed the utterance string to the GPT-2 model to predict the dialogue state, incorporating the GAT features learned in the previous step (Lin et al., 2021). Adding this intermediate step of passing through a GAT mitigates the decrease in performance caused by GPT-2's causality. Also, it has been shown to capture inter-slot dependencies, improve predictions at intermediate dialogue turns, and improve the predictions of correlated slots.

latest version is MultiWOZ 2.2 (Zang et al., 2020).

3 Dataset

The Multi-Domain Wizard of Oz (MultiWOZ) dataset is a fully-labelled collection of human-human written conversations spanning multiple domains and topics. It is the first widely used multi-domain dialogue dataset for the DST task (Balaraman et al., 2021). It comprises dialogues in seven domains: Attraction, Hospital, Police, Hotel, Restaurant, Taxi, and Train (the latter four of which are extended domains that include the subtask Booking), collected using the Wizard-of-Oz approach (Budzianowski et al., 2018). The dialogues cover between one and five domains per dialogue, greatly varying in length and complexity. 10438 dialogues were released, of which 3406 are single-domain, and 7,032 are multi-domain. At about 10 thousand dialogues, it is considerably larger than all previous annotated task-oriented corpora.

Since its first release, MultiWOZ has gone through several iterations. In particular, since MultiWOZ 2.0 that KAGE-GPT2 used, a new schema has been added, slot values standardized, annotation errors corrected, span annotations standardized, active intents and requested slots for each user turn annotated, and user and system actions fixed and added in MultiWOZ 2.2 (Zang et al., 2020). Performances of state-of-the-art models like TRADE, SGD-baseline, and DS-DST are similar upon the updates and is a compelling reason for using the cleaned MultiWOZ 2.2 dataset for fairer comparison between our proposed GPT-3 model and KAGE-GPT2.

4 Approaches

4.1 Adapting and Substituting the transformer model

As mentioned in §2.2, Lin et al. utilized GPT-2 to obtain the value of the embedding for each slot

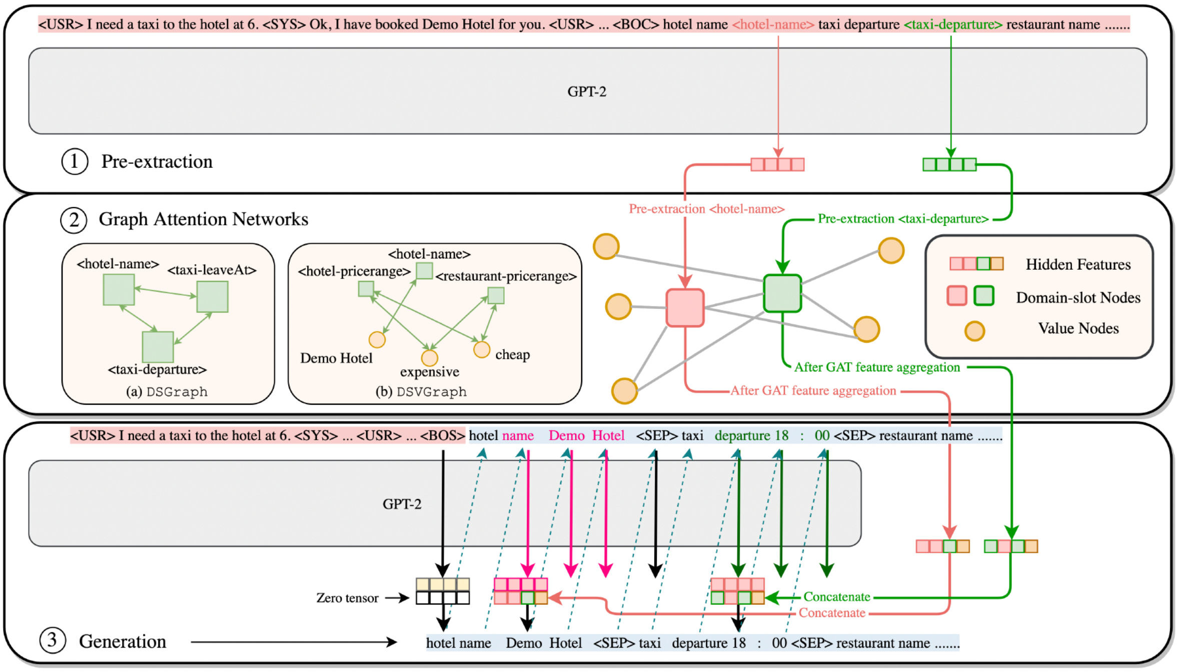


Figure 1: Training workflow of the KAGE-GPT2 model proposed and built by the KAGE-GPT2 authors (Lin et al., 2021). From the authors: (1) the pre-extraction layer is where the model extracts domain-slot embeddings (e.g., *hotel-name*) from dialogue history; (2) the GAN layer is where inter-slot relations are learned from the domain-slot embeddings passed from (1); (3) generation layer is where the updated domain-slot features are fed into GPT2 to generate the predicted dialogue state of slot values causally.

name in the first step of the model; in the third step of the model, the GPT-2 transformer is used again to obtain an embedding of the combined user's and system's utterances. We hypothesized that substituting the transformer model with a similar but more sophisticated one may let the model generate a 'richer' and more 'meaningful' embedding both for the Graph Attention Network in step 2 and the final prediction step.

We considered and have attempted to adapt Lin et al.'s model with the variants of GPT models with various complexities as in table 1.

	# of layer	# of parameters
GPT-3	96	175 Billion
GPTNeoX	44	20 Billion
GPT-2-XL	48	1557 Million
GPT-2 Large	36	774 Million
GPT-2 Medium	24	1558 Million
GPT-2	12	117 Million

Table 1: GPT models with various complexities that we attempted to train on the multi-domain problem.

4.2 Analysis of Dialogue State Tracking for Specific Domains

We recognize that the KAGE-GPT2 method performed decently well for its time as it achieved a joint accuracy of 54.86% and a slot accuracy of 97.47% (Lin et al., 2021). These results were produced when the model was trained and tested across five domains: attraction, hotel, restaurant, taxi, and train. We individually tested the author's pre-trained GPT-2 model against each of the five aforementioned domains. The results received from each domain were compared to results tested against all domains. The goal of these experiments was to find if the pre-trained GPT-2 model performs better on certain specific domains than others and if the model performs better when restricted to an individual domain than when run on a multi-domain ontology.

We tested the author's pre-trained GPT2 model on two subsets of dialogues from the original test dataset to do this domain-specific analysis for each of the five experimental domains. The first subset was dialogues classified as being in exactly and only the target domain. In contrast, the second was dialogues classified as being in the specified domain and zero or more other non-target domains.

Support for two new command-line arguments, `-test_domain` and `-test_consider_other_domains`, was added to the evaluation script. Specifying a target domain in evaluation would modify the slots passed to the training algorithm to be learned and evaluated and trim the test dataset to only include the previously mentioned subset of dialogues for evaluation.

4.3 Substituting MultiWOZ 2.0 with MultiWOZ 2.1

MultiWOZ 2.1 corrects four main dialogue state error types in the original MultiWOZ 2.0 dataset that in practice, has been found to have substantial noise (Eric et al., 2019)—delayed annotations of slot values one or more turns after an initial appearance in user utterances, multi-annotations of slot values where only one is correct, mis-annotations of slot values, typographically-inconsistent annotations, and forgotten slot values that never occur in the dialogue state despite being mentioned in user utterance(s). Additionally, the newer dataset includes annotations for user utterances instead of the existing annotations for system dialogue acts.

The MultiWOZ 2.1 authors found consistent drops in the test set joint state accuracies for various Joint State Tracker models (e.g., Flat Joint State Tracker, Hierarchical Joint State Tracker, and TRADE) due to the newer dataset causing models to generate more incorrect slot value predictions when the target label is *none* or *dontcare*. In this paper we compared the results obtained from evaluating Lin et al.'s KAGE-GPT2 pre-trained model on individual domains, unions of domains (with a specified target domain), and the original multi-domain dataset, on the MultiWOZ 2.0 dataset, to results obtained from the evaluation on the MultiWOZ 2.1 dataset. We analyzed the differences to see if they matched the MultiWOZ 2.1 authors' findings.

The authors also found the largest slot accuracy decrease from MultiWOZ 2.0 to MultiWOZ 2.1 occurred for the **restaurant-name** slot. In this paper, we also evaluated the KAGE-GPT2 model on the individual domain of *restaurant*, and thus compared the results obtained from the original MultiWOZ 2.0 dataset to that obtained from the newer data-set to see if these same discrepancies are apparent on this model.

5 Evaluation and Results

The two performance metrics used are joint goal accuracy and slot accuracy. Joint goal accuracy, or joint accuracy, is computed by assigning a value of 1 or 0 to each dialogue turn depending on whether the predicted dialogue state (also called a belief state) matches the ground-truth belief state—that is, whether all slot-value predictions of a dialogue turn match all slot-value pairs in the ground-truth belief state—then computing the average of these boolean indicator values across all dialogue turns for a dialogue. Slot accuracy is computed at a finer-grained level by computing the ratio of correct slot-value predictions of each turn, then computing the average of these ratios.

5.1 Evaluation and Results from the Transformer Substitution Experiments

In essence, we drastically underestimated the effort and resources needed to substitute these models to adapt the original paper authors' 8000-line code base to work with these new models effectively.

Unfortunately, we did not yield many satisfactory results in this arduous process. We initially considered using GPT-3 as mentioned in our original project proposal. However, the plan to substitute GPT-2 to GPT-3 was unfortunately put on halt as we realized that Huggingface does not provide direct support to embed GPT-3 in our codebase as it was unbeknownst to us that GPT-3 is not an open-source model. This would require us to use OpenAI's custom API, which would require us to rewrite almost the majority of the 8000-line code base, which we ultimately decided was not economical.

Hence, we focused on finding an alternative model to GPT-3. We found out more about GPT-NeoX, an alternative model with roughly 20 billion trainable parameters. We thought this would be a model with 'decent' complexity. Even though the number of parameters GPTNeoX has is one degree of magnitude less than GPT-3, GPTNeoX still has about two degrees of magnitude more parameters than our baseline GPT-2 model, which we thought would ultimately lead to improved performance.

However, we encountered several issues attempting to conform the author's codebase to utilize GPTNeoX. The following is a non-exhaustive list of problems encountered during our development process:

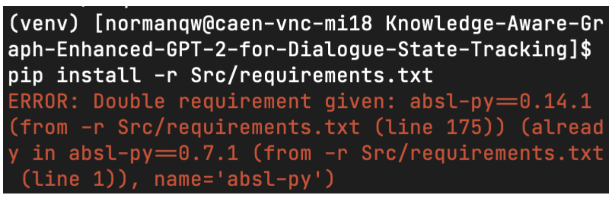
5.1.1 Conflicting `requirements.txt` Provided by the Authors

We naturally started with using the author's Github repository. The first issue we encountered was correctly setting up a functional python environment using the author-provided `requirements.txt`. First, the author did not specify which version of Python, the training environment that was originally used, and whether the environment should be set up in `conda` or `pip`. Thus, we created a permutation of these setup environments by choosing a specific Python version, one of Python 3.6.8, 3.7, 3.8, 3.9, and 3.10, a specific package manager, e.g. `conda` or `pip`. This was worsened by the fact that we had to circumnavigate different restrictions on the various computation platform that we are limited to, namely local Mac environments, Google Colabatory, CAEN, and Great Lakes Slurm HPC Clusters, which we further elaborate on in the 'Resource Limitations' section. We had to create more than 20 `conda`/`pip` environments to find a suitable environment for each computation platform.

However, in every environment that we tried in the series of permutations, if we used the author's requirements file unmodified, we would inevitably encounter the following issues:

- There is one specific requirement line named `pkg-resources==0.0.0`. After extensive research, we concluded that this specific requirement is likely to be a bug resulting from the authors' specific Linux distribution (Wright, 2016).

- For whatever reason, notwithstanding the previous issue, all of the authors' requirements are specified using ==, which is likely to be the result of a `pip freeze` of the authors' local environment. However, this seems to have created unnecessarily strict requirements such that the most recent versions of `pip` can no longer resolve the dependencies conflicts, as shown in the figure below.



```
(venv) [normanqw@caen-vnc-mi18 Knowledge-Aware-Graph-Enhanced-GPT-2-for-Dialogue-State-Tracking]$ pip install -r Src/requirements.txt
ERROR: Double requirement given: absl-py==0.14.1 (from -r Src/requirements.txt (line 175)) (already in absl-py==0.7.1 (from -r Src/requirements.txt (line 1)), name='absl-py')
```

Figure 2: An example of various package conflicts we had to manually resolve one by one in the beginning stage of the setting up relevant environments

This is not just the only conflict but one amongst the tens of dozens of conflicts we encountered along the way. Thus, we had to make assumptions about which packages are necessary to be kept, such as `torch`, `tensorboard`, `transformers` to use Huggingface's library functions. We had to keep trying to fail to see which versions of which packages were essential to the execution of the program while not breaking `CUDA` and `transformers` compatibility. As mentioned, the discrepancies between the different versions will be one of our main struggles throughout this project.

- These conflicts may surface differently on different computation platforms, further increasing the confusion and difficulty associated with the setup process. For instance, it is easier to set up `CUDA` on Great Lakes than on Google Colab, as every module needs to be loaded 'from scratch". In contrast, an initial uninstallation process needs to take place on Google Colab before using `wget` to obtain an archived version of Pytorch with older `CUDA` compatibility.

5.1.2 Deprecation of Certain `Huggingface` Functions

One of the other main issues that we experienced was the need to deal with the discrepancy caused by the difference in the version of the `transformers` libraries used by the author, `3.5.1`, which is 25 version releases behind the latest version 4.5.1, which enables us easier access to an implementation of the `GPTNeoX` model. However, as `transformers` library iterated, the code file structures shifted around, and many functions were renamed or removed as specific implementation details in library functions changed. In the latest 4.5.1 version of `transformers`, two functions were called in the paper authors' code-base in their KAGE-GPT2 model file, specifically `_init_sequence_length_for_generation`, and `_update_seq_length_for_generation` were removed from the `GenerationMixin` class in `transformers/src/transformers/generation/utils.py` which are inherited from the general class for pre-trained models. We had to trace through the source code function call after function call to investigate the best way to fix such compatibility issues, which may result in further knock-on effects. We appended those two aforementioned functions to the original implementation of the `KAGE_GPT2.py` model (tra, 2020).

Unfortunately, even though our training script seemed to be able to execute normally when generating the validation results, for unknown reasons, the transformers based on the newer 4.25.1 version cannot reliably generate a slot-value pair in the eventual output layer. However, we were eventually able to fix such issues only in environments installed with the older 3.5.1 version of the transformer. By using differential testing techniques, we concluded that presumably unknown latent changes to the Huggingface library caused the discrepancy in the output dialogue generation.

We realized that since most of the changes in the library code were not within our purview, it would not be worth the risk and time to hack the code-base further to work with `transformers` version 4.25.1. However, this meant that we had to revert to the authors' `transformers` versions which meant that we could no longer use the GPT-NeoX model. Subsequently, we looked for more available native models on version 3.5.1, which we would not need to implement from scratch. Hence, as mentioned in §4.1, we experimented with variants of GPT-2 models: GPT-2-Extra-Large, GPT2-Large, and GPT2-Medium as the next set of targets of the transformer substitution experiments.

5.1.3 Resource Limitations

We faced quite some severe limitations with resources throughout the project. Unfortunately, due to the aforementioned difficulties in getting a custom model to run, we did not have too much uncongested time using the Great Lakes computing cluster. We often had to wait more than 24 hours for a simple less-than-1-hour testing script to start running. Despite the difficulties, we fully debugged the training script on Great Lakes for our experiments for substituting GPT-2 for other GPT-2 variants. However, we encountered an unforeseen difficulty in fitting a Large Language Model through Great Lakes. We attempted to finetune GPT-2-XL, GPT-2 Large models on Great Lakes. However, even with a training batch size of 1, a dialogue in the MultiWOZ dataset may be too long , the intermediate variables may not fully fit into the 48GB of storage provided by one NVIDIA A40 GPU. Even though it is technically possible to utilize multiple GPUs while training, it would be nearly impossible to have an accurate estimate of how much rewriting needs to be completed to have a fully functional code-base again especially given that older versions of `transformers` may not be suited to perform multi-GPU tasks. Due to time limitations as well, we were only able to finetune the GPT-2-Medium model, which would fit successfully the memory constraint using a single GPU.

```
graphAttentionLSIGFBatch_modified
  y = torch.matmul(z, h)  # B x P x N x F
RuntimeError: CUDA out of memory. Tried to allocate 314.00
MiB (GPU 0; 44.37 GiB total capacity; 41.68 GiB already
allocated; 146.50 MiB free; 42.87 GiB reserved in total by
PyTorch)
```

Figure 3: Training GPT-2-XL/GPT-2 Large model would cause a single-GPU instance on Great Lakes to run out of memory

Meanwhile, we experienced significant lags while attempting to finetune/test our models on Great Lakes. We had to rely on other computing platforms such as Google Colabatory and CAEN. We spent $74.99 on Google Colab to purchase enough computing credits to sanity-test the author's GPT-2 training and testing script, finetuning the various GPT-2 variant models aforementioned, and running our domain-specific analysis scripts.

The difficulty in using a GPU is not the only issue we encountered throughout the experiments and analysis runs. As we also needed to store our trained model, permanent storage devices became a significant issue. The authors' pre-trained GPT-2 model alone took up around 40GB of space, but we were only given around 80GB of storage in our `/home` directory. Unfortunately, once a user's `/home` directory becomes full, it would be impossible to perform basically operations. It would even be possible to run `rm -rf` as presumably removing files would require writing to the file meta-data. It took significant effort to restore the file system to its normal state. As we also wanted to share our work on Great Lakes, we attempted to use the `/scratch/eecs595f22_class_-root/eecs595f22_class/shared_-data/` directory, we did not know that there is an implicit storage limit for the `shared_data` directory, our saved training models along with the authors' models somehow also exceeded the limit allowed. Hence, we had to migrate our code base again to our individually allocated folder in the `/scratch` directory. As we also took advantage of Google Colab, we had to utilize the Google

Drive File System. However, Google Drive File System is not the best at addressing filename conflicts and multiple users writing to the same file simultaneously. It was often the case that on the GUI, the files and directories may appear to have the same name, but in the actual underlying file system, they have different names. We learnt this the hard way by accidentally deleting the wrong version of our modified files.

5.2 Evaluation with GPT-2-Medium

Table 2 lists the joint goal and slot accuracies obtained by evaluating the author's pre-trained model and our adaptation of the GPT2-Medium model on the MultiWOZ 2.0 dataset.

Utilized Models	**Accuracy (%)**	
	Joint	**Slot**
GPT-2	42	95.70
GPT-2-Medium	34	95.13

Table 2: Results of evaluating pre-trained KAGE-GPT2 model and the KAGE-GPT2-Medium model on the MultiWOZ 2.0 dataset.

Unfortunately, we only had enough time and computational resources to finetune the GPT-2-Medium model for one epoch rather than eight epochs for GPT-2. However, it seems that the variant model has already achieved a similar level of accuracy as the regular model. This corroborates with the initial goals of these experiments that a larger transformer model may yield higher accuracy.

5.3 Evaluation on Specific Target Domains

Table 3 lists the joint goal and slot accuracies obtained by evaluating the author's epoch eight pre-trained model on the MultiWOZ 2.0 data-set for dialogues in specific individual target domains. In both tables, 3 and 4, the "full dataset, all domains" row correspond to running the model on the author's original unmodified code and dataset. In table 3, every other row corresponds to evaluating the model on dialogue data whose `domains` field contains only and exactly the target domain. In contrast, in table 4, every other row corresponds to evaluating the model on dialogue data whose `domains` field contains at least the target domain, but additionally zero or more other non-target domains from the list of domains in the static ontology.

Domain(s)	**Accuracy (%)**	
	Joint	**Slot**
Full dataset, all domains	42	95.70
Attraction domain	90.70	99.69
Hotel domain	51	96.93
Restaurant domain	64	98.50
Taxi domain	83	99.33
Train domain	70	98.67

Table 3: Results of evaluating pre-trained KAGE-GPT2 model on specific individual target domains on the MultiWOZ 2.0 dataset.

Domain(s)	**Accuracy (%)**	
	Joint	**Slot**
Full dataset, all domains	42	95.70
Attraction + other domain(s)	71	98.83
Hotel + other domain(s)	42	95.70
Restaurant + other domain(s)	42	95.70
Taxi + other domain(s)	34	95.13
Train + other domain(s)	78	98.67

Table 4: Results of evaluating pre-trained KAGE-GPT2 model on specific individual target domains and zero or more other non-target domains on the MultiWOZ 2.0 dataset.

5.4 Evaluation on Updated Dataset

Table 5 lists the joint goal accuracies obtained by evaluating the same epoch 8 pre-trained model on both the MultiWOZ 2.0 and MultiWOZ 2.1 datasets, for dialogues in specific individual target domains. Table 5 is analogous to table 3 in that each row corresponds to evaluating the model on dialogue data whose domains field contains only and exactly the target domain; while table 6 is analogous to table 4 in that each row corresponds to evaluating the model on dialogue data whose `domains` field contains the target domain and zero or more other non-target domains.

Domain(s)	Joint accuracy (%)	
	WOZ 2.0	WOZ 2.1
Full data, all domains	42	34
Attraction	90.70	86.05
Hotel	51	36
Restaurant	64	53
Taxi	83	64
Train	70	65

Table 5: Joint accuracy results of evaluating pre-trained KAGE-GPT2 model on specific individual target domains, on MultiWOZ 2.0 versus MultiWOZ 2.1.

Domain(s)	Joint accuracy (%)	
	WOZ 2.0	WOZ 2.1
Full data, all domains	42	34
Attraction + other(s)	71	41
Hotel + other(s)	42	34
Restaurant + other(s)	42	34
Taxi + other(s)	34	25
Train + other(s)	78	42

Table 6: Joint accuracy results of evaluating pre-trained KAGE-GPT2 model on specific individual target domains and zero or more other non-target domains, on MultiWOZ 2.0 versus MultiWOZ 2.1.

Tables 8 and 7 are analogous to tables 6 and 5 in what their rows correspond to, respectively; but, instead of joint goal accuracy, they list the slot accuracies obtained by evaluating the same epoch eight pre-trained model on both the MultiWOZ 2.0 and MultiWOZ 2.1 datasets.

Domain(s)	Slot accuracy (%)	
	WOZ 2.0	WOZ 2.1
Full data, all domains	95.70	93.90
Attraction	99.69	99.53
Hotel	96.93	96.17
Restaurant	98.50	98.03
Taxi	99.33	98.5
Train	98.67	98.50

Table 7: Slot accuracy results of evaluating pre-trained KAGE-GPT2 model on specific individual target domains, on MultiWOZ 2.0 versus MultiWOZ 2.1.

Domain(s)	Slot accuracy (%)	
	WOZ 2.0	WOZ 2.1
Full data, all domains	95.70	93.90
Attraction + other(s)	98.83	97.10
Hotel + other(s)	95.70	93.90
Restaurant + other(s)	95.70	93.90
Taxi + other(s)	95.13	93.67
Train + other(s)	98.67	97.67

Table 8: Slot accuracy results of evaluating pre-trained KAGE-GPT2 model on specific individual target domains and zero or more other non-target domains, on MultiWOZ 2.0 versus MultiWOZ 2.1.

6 Discussion

In this section, we analyze and discuss the trends and patterns in the obtained data.

6.1 Analyzing Evaluation Results with the GPT-2-Medium model

In table 2, we observe that the joint and slot accuracies obtained with the GPT-2 Medium model are worse than the accuracy given with the GPT-2 model. There is a joint accuracy loss of 8% and slot accuracy loss of 0.6% with the GPT-2-Medium model. This is due to the amount of time training on the GPT-2-Medium model. The author's pre-trained model ran for eight epochs, while the GPT-2-Medium model ran for only one epoch. They both share the same hyperparameters. Since the GPT-2-Medium model does well for only having one epoch of training time and almost having the same slot accuracy, we predict that it'll surpass the author's implementation given enough time. We couldn't run the model long enough due to lack of time, as one epoch takes five hours to train.

6.2 Analyzing Evaluation Results on Specific Target Domains

6.2.1 Joint Accuracy Less Than Slot Accuracy

In both tables 3 and 4, joint goal accuracies are consistently less than slot accuracies for all experiments. This trend also aligns with the results observed by the KAGE-GPT2 authors. This is expected since joint goal accuracy has much stricter requirements than slot accuracy, and slot accuracy is, in some sense, simply a finer-grained and more relaxed metric—anywhere a dialogue turn has a slot accuracy ratio less than 100%, it would have a joint accuracy indicator of 0.

6.2.2 Better Performance on Single Target Domain

In the single target domain case, evaluation of the epoch eight pre-trained models produced higher joint goal and slot accuracies compared to the original multi-domain problem. Joint goal accuracies obtained from the evaluation of the five individual target domains range of 51-90.70%. They are all higher than the 42% joint accuracy on the multi-domain dataset, while slot accuracies obtained are in the range of 96.93-99.69% and are all higher than the 95.70% slot accuracy on the multi-domain dataset.

This is expected because of two reasons: (1) the model is effectively being evaluated on a small subset test dataset of dialogues, particularly in the single target domain case where the size of the subsets are much smaller, hence statistically, the accuracies are naturally higher since there is much less room for the model to make incorrect slot predictions; and (2) the restricted single target domain problems likely contain much fewer examples of *none* or *dontcare*, and the model is much more likely to make slot-value prediction errors due to the possibility of being confused by other domains present in multi-domain dialogue examples.

6.2.3 Better Performance When Excluding All Non-Target Domains

This second reason should also explain why performance in the single target domain case (table 3) is better than that in the joint target + other non-target domain(s) case (table 4), as seen from the results where slot accuracy in the former are in the range 96.93-99.69% while that in the latter is in the range 95.13-98.83%; and joint goal accuracy in the former are in the range 51-90.70% while that in the latter is in the range 34-78%.

6.2.4 Best and Worst Target Domains

Evaluation of the model on the **attraction** domain produced the best joint goal and slot accuracies across the board, as seen in tables 3 and 4, while evaluation of the model on the **hotel** domain in the single target domain case produced the worst joint and slot accuracies.

6.3 Analyzing Evaluation Results on Updated Dataset

6.3.1 Lower Performance on MultiWOZ 2.1 Across All Domains

Tables 5, 6 and 7 illustrate that joint goal and slot accuracies obtained from the evaluation of the KAGE-GPT2 model on MultiWOZ 2.1 are lower than on MultiWOZ 2.0 across the board. In the single target domain case, joint accuracies on MultiWOZ 2.1 are lower than on MultiWOZ 2.0 across all individual domains, including joint domain sets (i.e., domain sets including the target domain and zero or more non-target domains). On average, however, the disparities between the joint goal accuracies are larger on the joint domain sets (table 6) than on single target domains (table 5).

Slot accuracy on MultiWOZ 2.1 was also lower than on MultiWOZ 2.0 across all individual target domains and all joint domain sets, as seen in tables 7 and 8 respectively, except in the evaluation results on single target domain of restaurant, as seen in table 7, with slot accuracy of 98.03% on MultiWOZ 2.1 falling short of the slot accuracy of 98.50% on MultiWOZ 2.0. These general results are expected and align with the MultiWOZ 2.1 authors' findings of joint accuracy drops from 2.0 to 2.1 when evaluated on baseline joint state tracker models like Flat Joint State Tracker and Hierarchical Joint State Tracker (Eric et al., 2019).

7 Lessons Learned

7.1 Taking a Shortcut May actually Result in a Detour

The goal of this project was frankly not too ambitious in my original opinion. We thought we could naively think that we may simply substitute the given GPT-2 Python class in the authors' codebase by using the GPTNeoX model. In the initial phase of the project, we had this illusion after reading through the documentation of both models and found out that their APIs are largely the same. This false sense of 'security' was further strengthed by our relative lack of experience dealing with large modern libraries such as `transformers`. We falsely thought that as long as the APIs resemble each other, the scope of the work shall be relatively limited.

In hindsight, perhaps it would have been faster to start a brand new code-base and use more modern libraries to implement what the authors' model was, and perform the transformer substitution experiments. Perhaps with the help of using newer libraries, we would have an easier time also attempting to convert the code base to work with multiple GPUs, which is called for to finetune any meaningful Large Language models in hindsight. While this conclusion is only a hypothesis, it was true that we should not have limited the scope of the project so earlier on, and fixated on getting the substitution experiments to work.

7.2 Asking for help earlier on

As we spent a long time simply failing and trying to set up the various code bases and environments, it did not occur to us that we were rather 'behind' in terms of overall progress. If we reached out for suggestions from Professor Chai or other GSIs sooner, we would have realized earlier on that we should have a set of diversified approaches, such as modifying the Graph Attention Network, experimenting with alternative models such as using an RNN belief state tracker or just basing our project on a different paper.

7.3 Time and Resource Management

We severely underestimated the time required to create an environment, adapt models, fine-tune, test, and analyze the various models. Even though we did not start the project that late, we should have perhaps started the project one or two weeks earlier. In the meantime, we should have used our computational resources more efficiently in hindsight. If we have realized that Great Lakes has a huge backlog, we should have made the case to use Google Colab earlier on.

8 Conclusion

We faced significant challenges attempting to improve the existing novel hybrid KAGE-GPT2 dialogue state tracking model to use GPT-3, largely due to versioning conflicts and deprecation of critical functions in the main transformers library used by the original KAGE-GPT2 authors. In light of this, we focused our analysis on evaluating the KAGE-GPT2 model, trained on a multi-domain problem, on specific individual target domains as well as these individual domains with zero or more other non-target domains (i.e., what we call "joint domains"). Our experiments found that the multi-domain KAGE-GPT2 model is extremely effective on single-domain problems. The model is much less likely to make erroneous slot-value predictions when not confused by other non-target domains. This is further substantiated by our results on joint domains having a slightly higher joint goal and slot accuracies compared to the results on single target domains.

We also evaluated the pre-trained KAGE-GPT2 model on the newer and updated MultiWOZ 2.1 addresses errors and removing noise from the original MultiWOZ 2.0 dataset. Our results in this experiment align with findings by the MultiWOZ 2.1 authors, with a joint goal and slot accuracies being lower on the newer dataset.

References

2020. hugging_face/transformers. https://huggingface.co/transformers/v4.2.2/_modules/transformers/generation_utils.html,.

Vevake Balaraman, Seyedmostafa Sheikhalishahi, and Bernardo Magnini. 2021. Recent neural methods on dialogue state tracking for task-oriented dialogue systems: A survey. In *SIGDIAL*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, 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 (EMNLP)*.

Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Peter Ku, Anuj Kumar Goyal, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tur. 2019. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Weizhe Lin, Bo-Hsiang Tseng, and Bill Byrne. 2021. Knowledge-aware graph-enhanced GPT-2 for dialogue state tracking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7871–7881, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Craig Wright. 2016. pip freeze includes "pkg-resources==0.0.0" (ubuntu server 16.04 lts). https://github.com/pypa/pip/issues/4022.

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

Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *arXiv preprint arXiv:1911.06192*.