Extracting and Inferring Personal Attributes from Dialogue

Zhilin Wang, Xuhui Zhou, Rik Koncel-Kedziorski, Alex Marin, Fei Xia

University of Washington

{zhilinw, xuhuizh, kedzior, amarin, fxia}@uw.edu

Abstract

Personal attributes represent structured information about a person, such as their hobbies, pets, family, likes and dislikes. In this work, we introduce the tasks of extracting and inferring personal attributes from human-human dialogue. We first demonstrate the benefit of incorporating personal attributes in a social chit-chat dialogue model and task-oriented dialogue setting. Thus motivated, we propose the tasks of personal attribute extraction and inference, and then analyze the linguistic demands of these tasks. To meet these challenges, we introduce a simple and extensible model that combines an autoregressive language model utilizing constrained attribute generation with a discriminative reranker. Our model outperforms strong baselines on extracting personal attributes as well as inferring personal attributes that are not contained verbatim in utterances and instead requires commonsense reasoning and lexical inferences, which occur frequently in everyday conversation.

1 Introduction

Personal attributes are structured information about a person, such as what they like, what they have, and what their favorite things are. These attributes are commonly revealed either explicitly or implicitly during social dialogue as shown in Figure 1, allowing people to know more about one another. These personal attributes, represented in the form of knowledge graph triples (*e.g.* I, has_hobby, volunteer) can represent large numbers of personal attributes in an interpretable manner, facilitating its usage by weakly-coupled downstream dialogue tasks (Li et al. 2014; Qian et al. 2018; Zheng et al. 2020a,b; Hogan et al. 2021).

One such task is to ground open-domain chit-chat dialogue agents to minimize inconsistencies in their language use (*e.g.*, **I like cabbage** →(next turn) →**Cabbage is disgusting**) and make them engaging to talk with (Li et al. 2016; Zhang et al. 2018; Mazaré et al. 2018; Qian et al. 2018; Zheng et al. 2020a,b; Li et al. 2020; Majumder et al. 2020). Thus far, personalization in chit-chat has made use of dense embeddings and natural language sentences. While KG triples have been shown to be capable of grounding Natural Language Generation (Moon et al. 2019; Koncel-Kedziorski et al. 2019), they have yet to be used to personalize chit-chat dialogue agents.

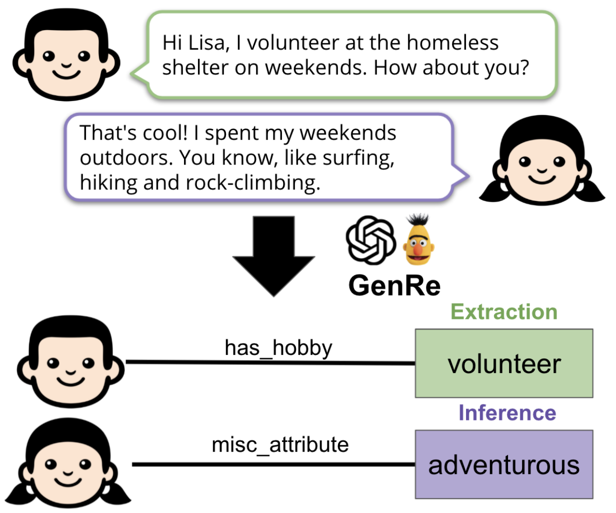
Hi Lisa, I volunteer at the homeless shelter on weekends. How about you?
That's cool! I spent my weekends outdoors. You know, like surfing, hiking and rock-climbing.
GenRe
Extraction
has_hobby
volunteer
Inference
misc_attribute
adventurous

Figure 1: Overview of obtaining personal attribute triple from utterances using our model GenRe. Attribute values are contained within the utterance in the EXTRACTION task, but not the INFERENCE task.

Personal attributes can also help task-oriented dialogue agents to provide personalized recommendations (Mo et al. 2017; Joshi, Mi, and Faltings 2017; Luo et al. 2019; Lu et al. 2019; Pei, Ren, and de Rijke 2021). Such personalized recommendations have only been attempted for single-domain tasks with a small set of one-hot features (< 30). Personalization across a wide range of tasks (recommending food, movies and music by multi-task dialogue agents such as Alexa, Siri and Assistant) however can require orders of magnitude more personal attribute features. This makes KG triples ideal for representing them, given the advantages of this data structure in selecting and utilizing pertinent features (Li et al. 2014; Hogan et al. 2021).

In Section 2, we show how personal attributes in the form of KG triples can improve the personalization of open-domain social chit-chat agents as well as task-oriented dialogue agents. In the former case, personal attributes can be utilized to improve chat-bot consistency on the PersonaChat task (Zhang et al. 2018). In the latter case, we suggest how our personal attributes can support personalization in multi-task, task-oriented dialogue settings.

Building upon the usefulness of personal attributes, we investigate how personal attributes can be predicted from dialogue. An important bottleneck for this step lies in the poor coverage of relevant personal attributes in existing labeled datasets. Therefore, we introduce two new tasks for identifying personal attributes in Section 3. As shown in Figure 1, the EXTRACTION task requires determining which phrase in an utterance indicate a personal attribute, while the INFERENCE task adds further challenge by requiring models to predict personal attributes that are not explicitly stated verbatim in utterances. This is common in conversational settings, where people express personal attributes using a variety of semantically related words or imply them using commonsense reasoning. We analyze how these factors allow personal attributes to be linked to utterances that express them.

To tackle these tasks, we propose a simple yet extensible model, **GenRe**, in Section 4. GenRe combines a constrained attribute generation model (that is flexible to accommodate attributes not found verbatim in utterances) with a discriminative reranker (that can contrast between highly similar candidates). Our experiments in Section 5 suggests that such design allows our model to outperform strong baseline models on both the EXTRACTION and INFERENCE tasks. Finally, in Section 6, detailed ablation studies demonstrate the value of our model components while further analysis identifies future areas for improvement.

2 Applications of Personal Attributes

Personal attributes can make social chit-chat agents more consistent and engaging as well as enable task-oriented agents to make personalized recommendations. In this section, we use personal attributes from the DialogNLI task to demonstrate how they can improve chit-chat agent consistency and provide information for personalizing task-oriented dialogue agents.

2.1 Source of Personal Attributes

DialogNLI (Welleck et al. 2019) contains samples of PersonaChat utterances (Zhang et al. 2018), each paired with a manually annotated personal attribute triple. Each triple consists of a head entity, a relation, and a tail entity. These triples were initially annotated to identify entailing, contradicting and neutral statements within the PersonaChat corpus. For instance, a statement labelled with (I, [has_profession], chef) will contradict with another statement labelled with (I, [has_profession], engineer). The three largest groups of relations are: a. *has_X* (where X = *hobby, vehicle, pet*) b. *favourite_Y* (where Y = *activity, color, music*) c. *like_Z* (where Z = *read, drink, movie*).

2.2 Consistency in Chit-chat agents

PersonaChat (Zhang et al. 2018) was created to improve the personality consistency of open-domain chit-chat dialogue agents. PersonaChat was constructed by giving pairs of crowdworkers a set of English personal attribute related sentences and asking them to chat in a way that is congruent with those sentences. Models were then trained to generate dialogue responses that are in line with those expressed by crowdworkers using the provided persona information as context.

Methods We fine-tune the generative version of Blender 90M (a transformer-based model trained on multiple related tasks) on PersonaChat, which is currently the state-of-the-art generative model on this task (Roller et al. 2020). We prepend a corresponding DialogNLI personal attribute before each utterance (*i.e.* **+Per. Attr.**), in order to better direct the model in generating a suitable response. This modification is relatively minimal to demonstrate the informativeness of personal attributes, while keeping the model architecture and hyperparameter fine-tuning the same as in the original work (details in Appendix A.1).

Metrics We follow Roller et al. (2020) and Dinan et al. (2019). Metrics for **+Per. Attr.** setting consider both personal attributes and utterances. **Hits@1** uses the hidden states of the generated output to select the most likely utterance amongst 20 candidates (the correct utterance and 19 randomly chosen utterances from the corpus). **Perplexity** reflects the quality of the trained language model. **F1** demonstrates the extent of the overlap between the generated sequence and the ground truth sequence.

	Hits@1 ↑	Perplexity ↓	F1 ↑
Blender	32.3	11.3	20.4
+ Per. Attr.	35.2*	10.4*	20.6*

Table 1: Effects of using personal attributes to augment Blender on Personachat. Higher is better for Hits@1 and F1; lower is better for perplexity. *Significantly different from Blender with 5 runs based on a two-tailed t-test ($p<0.05$).

Fact 1	I love cats and have two cats
Fact 2	I've a hat collection of over 1000 hats.
Blender	My cats names are all the hats i have
+ Per. Attr.	My cats are called kitties
Fact 1	I am a doctor.
Fact 2	My daughter is a child prodigy.
Blender	My daughter is prodigy so she gets a lot of accidents.
+ Per. Attr.	I've seen a lot of accidents.

Table 2: Examples of incorrect utterances generated by Blender by mixing up two facts.

Results As shown in Table 1, including personal attributes can improve performance on the PersonaChat task. An inspection of the generated utterances suggests that including personal attributes into Blender can more effectively inform the model which persona statement to focus on during generation. This can prevent Blender from including information in irrelevant persona statements (*e.g.* by mixing up facts from two unrelated persona statements), as in Table 2.

2.3 Personalization in Task-oriented dialogue

While personalization has been incorporated into single-task settings (Joshi, Mi, and Faltings 2017; Mo et al. 2017; Luo et al. 2019; Lu et al. 2019; Pei, Ren, and de Rijke 2021), there has been no attempt for personalization in multi-task settings. This is against the background in which multi-task dialogue is rapidly becoming the standard in task-oriented dialogue evaluation (Byrne et al. 2019; Rastogi et al. 2019; Zang et al. 2020; Shalyminov et al. 2020). To overcome this gap, we show how our dataset can lay a foundational building block for personalization in multi-task dialogue.

Methods We used several popular datasets on multi-task task-oriented dialogue (Zang et al. 2020; Shalyminov et al. 2020; Byrne et al. 2019; Rastogi et al. 2019). From each dataset, we manually observed its tasks and categorized them into several overarching domains, as shown in Table 3. We then created a mapping between the various domains and datasets available for personalizing task-oriented dialogue (including ours). Domains that are not supported by any dataset are omitted.

Results

Dataset	Domains	#Unique features
Ours	Restaurants, Movies, Music, Sports, Recreation, Shopping	5583
Ours	Restaurants *only*	206
Joshi, Mi, and Faltings (2017)	Restaurants	30
Mo et al. (2017)	Restaurants	10
Lu et al. (2019)	Shopping	7

Table 3: Domains covered by various datasets for personalizing task-oriented dialogue. #Uniques features refers to the number of unique attribute-values (*e.g.* the specific food people like) that can be used for personalization.

Compared to existing datasets in Table 3, our dataset is capable of personalizing recommendations in a much larger number of domains. These domains include restaurants and shopping, which have been explored by existing datasets, as well as movies, music, sports and recreation, which have thus far been overlooked. For domains that have been previously explored, such as restaurants, our dataset contains a more diverse set of possible personal attribute values (*e.g.* the foods people like), which can support it to personalize recommendations in more realistic manners. The multi-domain nature of our dataset means that personalization can be applied on multiple tasks involved in a single request, such as planning a weekend trip, as shown in Figure 2. In particular, it allows related tasks to share information (*e.g.,* recommending pizzerias near the stadium where a particular baseball game was played). To utilize personal attributes, task-specific APIs (*e.g.,* Restaurant API) can request task-relevant personal attributes (*e.g.,* like_food) from a centralized Personal Attributes API, before applying them towards personalization in a similar manner to the single-task setting (Mo et al. 2017; Joshi, Mi, and Faltings 2017; Lu et al. 2019).

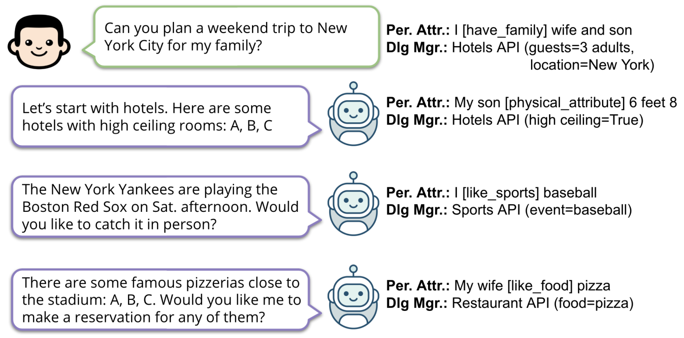
Can you plan a weekend trip to New York City for my family?
Per. Attr.: I [have_family] wife and son
Dlg Mgr.: Hotels API (guests=3 adults, location=New York)
Let's start with hotels. Here are some hotels with high ceiling rooms: A, B, C
Per. Attr.: My son [physical_attribute] 6 feet 8
Dlg Mgr.: Hotels API (high ceiling=True)
The New York Yankees are playing the Boston Red Sox on Sat. afternoon. Would you like to catch it in person?
Per. Attr.: I [like_sports] baseball
Dlg Mgr.: Sports API (event=baseball)
There are some famous pizzerias close to the stadium: A, B, C. Would you like me to make a reservation for any of them?
Per. Attr.: My wife [like_food] pizza
Dlg Mgr.: Restaurant API (food=pizza)

Figure 2: Example conversation involving personalization in multi-task Task-Oriented Dialogue.

3 Personal Attribute Tasks

Having motivated the importance of personal attributes, we propose the task of obtaining them from natural language sentences. We first explain how we formulate two complementary tasks from DialogNLI data and then formally define our tasks. Finally, we analyze the task datasets to gather insights into the linguistic phenomena that our tasks involve.

3.1 Extraction and Inference Tasks

By re-purposing the DialogNLI dataset, our tasks seek to extract these personal attribute triples from their paired utterances. We first used a script that obtains pairs of personal triples and utterances. Next, we combined relations with similar meanings such as like_food and favourite_food and removed under-specified relations such as favourite, have and others. Finally, we removed invalid samples with triples containing *None* or $< blank >$ and removed prefix numbers of tail entities (*e.g.* 11 dogs), since the quantity is not important for our investigation.

We formulate two tasks by partitioning the DialogNLI dataset into two non-overlapping subsets. Here, each **sample** refers to a sentence paired with an annotated triple. Train/dev/test splits follow DialogNLI, with descriptive statistics shown in Table 4. The dataset for the EXTRACTION task contains samples in which both the head and tail entities are spans inside the paired sentence. An example is (I, [has_profession], receptionist) from the sentence "I work as a receptionist in my day job". We formulate the EXTRACTION task in a similar way to existing Relation Extraction tasks such as ACE05 (Wadden et al. 2019) and NYT24 (Nayak and Ng 2020). This allows us to apply modeling lessons learned from Relation Extraction.

The complementary set is the dataset for the INFERENCE task, for which the head entity and/or the tail entity cannot be found as spans within the paired sentence. This is because the head and tail entities are paraphrased from the original sentence, or derived based on commonsense reasoning used by annotators. An example of a paraphrased triple is (I, [physical_attribute], tall) from the sentence "I am in the 99th height percentile", while one based on commonsense reasoning is (I, [want_job], umpire) from the sentence "my ultimate goal would be calling a ball game".

	EXTRACTION	INFERENCE
Samples		
train	22911	25328
dev.	2676	2658
test	2746	2452
Unique elements		
head entities	88	109
relations	39	39
tail entities	2381	2522
Avg. words		
head entities	1.03	1.08
relations	1.00	1.00
tail entities	1.20	1.28
sentences	12.9	12.2

Table 4: Statistics of the dataset for the two tasks.

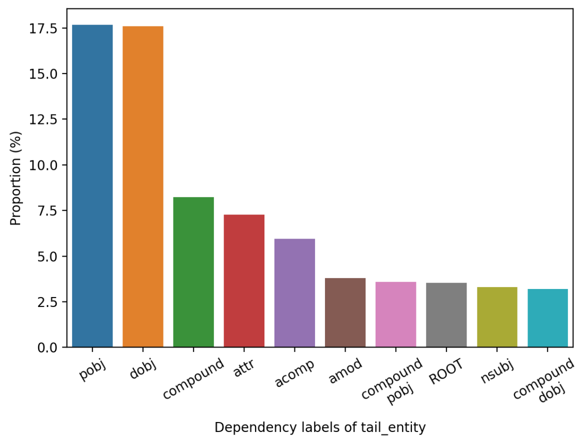

Figure 3: Bar plot for 10 most common dependency labels of tail entities

The INFERENCE task is posed as a challenging version of the EXTRACTION task that tests models' ability to identify pertinent information in sentences and then make commonsense inferences/paraphrases based on such information. An existing task has sought to predict personal attributes that are not always explicitly found within sentences (Wu et al. 2019). However, it did not distinguish between personal attributes that can be explicitly found within sentences (*i.e.* EXTRACTION) from those that cannot (*i.e.* INFERENCE) . We believe that, given that the inherent difficulty of identifying the two types of personal attributes are greatly different, it is helpful to pose them as two separate tasks. In this way, the research community can first aim for an adequate performance on the simpler task before applying lessons to make progress at the more challenging task. This is also the first time that personal attributes that are not explicitly contained in sentences are shown to be derivable from words in the sentence using commonsense/lexical inferences.

3.2 Formal Task Definition

Given a sentence S, we want to obtain a personal-attribute triple in the form of `(head entity, relation, tail entity)`. The relation must belong to a set of 39 predefined relations. In the EXTRACTION subset, the head entity and tail entity are spans within S. Conversely, in the INFERENCE subset, the head entity and/or the tail entity cannot be found as spans within S.

3.3 Dataset Analysis

We analyze the datasets to obtain insights into how the tasks can be approached. Because the majority of head entities (93.3%) are simply the word "I", our analysis will focus on tail entities.

Dataset for the EXTRACTION task We use dependency parses of sentences to understand the relationship between words within tail entities and the sentence ROOT. Dependency parsing was chosen because it is a well-studied syntactic task (Nivre et al. 2016) and has been shown to contribute to the relation extraction task (Zhang et al. 2017). Dependency parses and labels associated with each dependent word were identified using a pre-trained transformer model from spaCy.[1] The parser was trained on data annotated with the ClearNLP dependency schema that is similar to Universal Dependencies (Nivre et al. 2016).[2]

As shown in Figure 3, objects of prepositions (pobj) and direct objects (dobj) each comprise of 17.5% of tail entities, followed by compound words (compound), attributes (attr) and adjectival complements (acomp), plus 138 other long-tail labels. The range of grammatical roles as well as the fact that one third of tail entities do not involve nouns (see Figure in Section A.2) also suggest that the tail entities in our dataset go beyond proper nouns, which are what many Relation Extraction datasets (*e.g.*, ACE05 and NYT24) are mainly concerned with. Such diversity in grammatical roles played by tail entities means that approaches based on parsing/named entity recognition alone are unlikely to be successful in the EXTRACTION task.

Dataset for the INFERENCE task A qualitative inspection of the dataset showed that inferences can be made on the basis of semantically-related words and commonsense inferences, as shown in examples discussed in Section 3.1. To better understand how tail entities can be inferred from the sentence in the INFERENCE subset, we analyze the relationship between words in the tail entity and words in the sentence. 79.2% of tail entities cannot be directly identified in the sentence. We performed a few transformations to identify potential links between the tail entity and the sentence. *ConceptNet_connect* refers to words with highest-weighted edges on ConceptNet to sentence words while *ConceptNet_related* refers to words that have closest embedding distances to sentence words. Details of their preparation are in Appendix A.3. As in Table 5, our analysis shows that a model that can perform well on the INFERENCE task requir-

[1]https://spacy.io/

[2]https://github.com/clir/clearnlp-guidelines/blob/master/md/specifications/dependency_labels.md

ing both WordNet semantic knowledge (Fellbaum 1998) as well as ConceptNet commonsense knowledge (Speer, Chin, and Havasi 2017).

Transformation	**Example** (sentence→tail entity)	**%**
ConceptNet_related	mother →female	**71.3**
ConceptNet_connect	wife →married	56.8
WordNet_synonym	outside →outdoors	39.5
WordNet_hypernym	drum →instrument	5.04
WordNet_hyponym	felines →cats	4.17
Same_stem	swimming →swim	43.3

Table 5: Proportion (%) of tail entities that can be related to sentence words after applying each transformation.

4 GenRe

This section proposes GenRe, a model that uses a unified architecture for both the EXTRACTION and the INFERENCE tasks. We use a simple and extensible generator-reranker framework to address the needs of the two tasks. On one hand, a generative model is necessary because head and/or tail entities cannot be directly extracted from the sentence for INFERENCE dataset. On the other hand, preliminary experiments using a Generator in isolation showed that a large proportion of correct triples are among the top-k - but not top-1 - outputs (see Table 9). A Reranker can be used to select the most likely triple among the top-k candidate triples, leading to a large improvement in performance (see Table 7).

4.1 Generator

We use an autoregressive language model (GPT-2 small) as our Generator because its extensive pre-training is useful in generating syntactically and semantically coherent entities. The small model was chosen to keep model size similar to baselines. We finetune this model to predict a personal attribute triple occurring in a given input sentence. Specifically, we treat the flattened triples as targets to be predicted using the original sentence as context. The triple is formatted with control tokens to distinguish the head entity, relation, and tail entity as follows:

$\mathbf{y}$ = [HEAD], $t^{head}_{1:m}$, [RELN], t^{reln}, [TAIL], $t^{tail}_{1:k}$

where {[HEAD],[RELN], [TAIL]} are control tokens, $t^{head}_{1:m}$ is the head entity (a sequence of length m), t^{reln} is a relation, and $t^{tail}_{1:k}$ is the tail entity.

During evaluation, we are given a sentence as context and seek to generate a personal attribute triple in the flattened format as above. To reduce the search space, we adopt a constrained generation approach. Specifically, after the [RELN] token, only one of 39 predefined relations can be generated, and so the output probability of all other tokens is set to 0. After the [TAIL] token, all output tokens not appearing in the input sentence will have zeroed probabilities in the EXTRACTION task. Conversely for the INFERENCE task, the only allowed output tokens after the [TAIL] token are those which have appeared following the predicted relation in the training data. For example, tail entities that can be generated with a [physical_attribute] relation include "short", "skinny" or "wears glasses", as these examples occur in the training data. We imposed this restriction to prevent the model from hallucinating attributes that are not associated to the predicted relation (such as "dog" with [physical_attribute]). Despite limiting the model's ability to generate novel but compatible tail entities (and thereby upper-bounding maximum possible recall to 75.7%), this approach in balance helped to improve model performance. Implementation details are in Appendix A.4.

4.2 Reranker

We use BERT-base as the Reranker because its bi-directionality allows tail tokens to influence the choice of relation tokens. Furthermore, BERT has demonstrated the best commonsense understanding among pre-trained language models (Petroni et al. 2019; Zhou et al. 2020). These benefits have led to many relation extraction models using BERT as part of the pipeline (Wadden et al. 2019; Yu et al. 2020; Ye et al. 2021).

For each S, we obtain the L most likely sequences using the Generator, including the context sentence. Each sequence is labelled as correct or incorrect based on whether the predicted triple (head entity, relation, tail entity) matches exactly the ground-truth triple. Incorrect sequences serve as challenging negative samples for the Reranker because they are extremely similar to the correct sequence since they were generated together. We fine-tune a BERT model with a binary cross-entropy loss function to classify whether sequences are correct. During inference, we select the sequence with the highest likelihood of being correct as our predicted sequence. We set L to 10 in all experiments. Implementation details are in Appendix A.5.

5 Experiments

We first explain the metrics used in the experiments. Next, we introduce the baseline models. Finally, we examine how GenRe compares to baseline models to understand its advantages.

5.1 Metrics

Micro-averaged Precision/Recall/F1 were calculated following Nayak and Ng (2020), in which a sample is considered correct only when all three elements (head_entity, relation and tail entity) are resolved correctly. We chose these metrics because we are interested in the proportion of all predicted personal attributes that have been correctly identified (precision) and of all ground truth personal attributes (recall). F1 is considered as an aggregate metric for precision and recall.

5.2 Baseline Models

Generative models can be used for both the EXTRACTION and the INFERENCE tasks.

WDec is an encoder-decoder model that achieved state-of-the-art performance in the NYT24 and NYT29 tasks

(Nayak and Ng 2020). The encoder is a Bi-LSTM, while the decoder is an LSTM with attention over encoder states. An optional copy mechanism can be used: when used, the decoder will only generate tokens found in the original sentence. The copy mechanism was used on the EXTRACTION dataset but not on the INFERENCE dataset (given their better empirical performance).

GPT2 is an autoregressive language model that we build GenRe on. We use the same configuration as in GenRe.

Extractive models can be used only for the EXTRACTION task, because they select for head and tail entities from the original sentence.

DyGIE++ is a RoBERTa-based model that achieved state-of-the-art performance in multiple relation extraction tasks including ACE05 (Wadden et al. 2019). It first extracts spans within the original sentence as head and tail entities. Then, it pairs up these entities with a relation and passes them through a graph neural network, with the head and tail entities as the nodes, and relations as the edges. This allows information flow between related entities before passing the triple through a classifier.

PNDec is an Encoder-Decoder model that achieved close to SOTA performance in NYT24 and NYT29 (Nayak and Ng 2020). It uses the same encoder as WDec but uses a pointer network to identify head and tail entities from the original sentence, which it pairs with possible relation tokens to form a triple that is subsequently classified.

All baseline models were trained on our datasets using their suggested hyper-parameters.

5.3 Model Results

	EXTRACTION			INFERENCE		
	P	R	F1	P	R	F1
GenRe	**68.0**	**52.4**	**59.2**	**46.5**	**35.4**	**39.2**
Generative						
WDec	57.0	49.0	52.7	33.6	34.7	34.1
GPT2	50.9	31.1	38.6	31.3	17.3	22.3
Extractive						
DyGIE++	60.8	50.9	55.3			
PNDec	63.1	49.5	55.5			

Table 6: Performance on the test set. GenRe has significantly higher F1 than all baseline models with 5 runs based on a two-tailed t-test ($p < 0.05$).

The top-performing baseline models on the EXTRACTION dataset are the extractive models, which select spans within the sentence and classify whether an entire triple is likely to be correct. Because there are only a small number of spans within the sentence, this approach can effectively limit its search space. On the other hand, extractive models cannot solve the INFERENCE task, because the underlying assumption that head and tail entities must be found within the sentence does not hold. Conversely, generative models perform more poorly on the Extraction task but are capable on the INFERENCE task. This is because generation happens in a left-to-right manner, meaning that some elements of the triple have to be generated without knowing what the other elements are. Our approach of combining a Generative model with a BERT-base Reranker that is akin to models used by Extractive approaches combines the best of both worlds. Not only does it perform well on the Extraction task ($\geq$ 3.7 F1 points over baselines), it also excels on the Inference task by performing $\geq$ 5.1 F1 points over baselines.

6 Analysis

We first conduct an ablation study to better understand the contribution of constrained generation and the Reranker, by measuring the performance of our model when each component is removed. Then, we seek to understand how errors are made on predicted personal attribute relations to identify future areas of improvement.

6.1 Ablation Study

Table 7 shows that both the Reranker and constrained generation contribute to the performance of GenRe. In particular, the constrained generation plays a larger role on the EXTRACTION dataset while the Reranker plays a greater role on the INFERENCE dataset.

	EXTRACTION			INFERENCE		
	P	R	F1	P	R	F1
GenRe	68.0	52.4	59.2	46.5	35.4	39.2
- Constr. Gen	53.5	40.7	46.2	37.2	27.1	31.4
- Reranker	67.6	41.0	51.0	31.0	22.3	25.9

Table 7: Ablation study for Reranker and constrained generation.

Constrained generation has a large impact on the EXTRACTION dataset (+13.0% F1), likely because it very much restricts the generation search space to spans from the context sentence. On the INFERENCE dataset, the original search space cannot be effectively limited to tokens in the context sentence. Therefore, applying the heuristic that only tail entities associated with a particular relation (in the training set) can be decoded is useful, even though it upper bounds maximum recall to 75.7%, which is much higher than the achieved 35.4%. Compared to the EXTRACTION dataset, the improvement on the INFERENCE dataset is smaller (+7.8% F1), since the range of tail entities that can be decoded after imposing the constraint is greater.

The Reranker is needed because, many times, the correct triple can be generated by the Generator but might not be the triple that is predicted to have the highest likelihood, as shown in Table 9. In particular, the predicted most likely relation and the tail entity can be partly correct ("old"/"retiring" instead of "60"). This suggests the need to refine the generated outputs of the Generator using a Reranker. The maximum possible recall on the EXTRACTION and INFERENCE tasks increases from 41.0% to 59.9% and 22.3% to 41.0% respectively when considering top-10 instead of only top-1 generated candidate. While the

Dataset	True Relation (n)	P	R	F1	Top 3 Most Frequent (n) Predicted Relations	True Tail Entities	Predicted Tail Entities
EXTRACTION	[has_profession] (274)	83.8	62.0	71.3	[has_profession] (189) [employed_by_general] (30) [want_job] (17)	teacher (29) nurse (28) real estate agent (25)	nurse (27) real estate (25) teacher (19)
	[have_pet] (149)	97.3	55.0	70.3	[have_pet] (88) [have_family] (18) [like_animal] (12)	dog (55) cat (45) pets (22)	cat (32) pets (23) dog (18)
INFERENCE	[like_food] (77)	46.7	41.6	44.0	[like_food] (62) [like_activity] (5) [like_animal] (4)	pizza (18) onion (9) italian (7)	pizza (19) italian cuisine (10) onion (8)
	[like_music] (71)	40.8	23.9	30.2	[like_music] (40) [favorite_music_artist] (9) [like_activity] (7)	jazz (10) country (9) rap (6)	the story so far (12) country (8) jazz (7)

Table 8: Some common relations in EXTRACTION and INFERENCE datasets

achieved recall (52.4% and 35.4% respectively) are still a distance from the maximum possible recall, the achieved recall is much higher than using the Generator alone.

Utterance	How old are you? I am 60, retiring in a few years.		
Elements	head entity	relation	tail entity
Ground truth	I	[has_age]	60
Predicted	I	[has_age]	old
Most likely	I	[has_age]	60
↓	I	[has_age]	retiring
	I	[other]	old
Less likely	I	[want_do]	retiring

Table 9: Example of Generator output (Extraction dataset) to demonstrate role of the Reranker.

6.2 Misclassification of Relations

Major sources of error on the EXTRACTION dataset came from relation tokens that have close semantic meanings. They either were related to one another (*e.g.*, [has_profession] vs [want_job]) or could be correlated with one another (*e.g.*, [like_animal] vs [have_pet] or [like_music] vs [favorite_music_artist]) , as illustrated in Table 8. Such errors likely arose due to the way that the DialogNLI dataset (Welleck et al. 2019) was annotated. Specifically, annotators were asked to label a single possible triple given a sentence instead of all applicable triples. Because of this, our evaluation metrics are likely to over-penalize models when they generate reasonable triples that did not match the ground truth. Future work can avoid this problem by labelling all possible triples and framing the task as multilabel learning.

7 Related Work

Personal Attribute Extraction: Most work on extracting personal attributes from natural language (Pappu and Rudnicky 2014; Mazaré et al. 2018; Wu et al. 2019; Tigunova et al. 2019, 2020) employed distant supervision approaches using heuristics and hand-crafted templates, which are known to have poor recall. In contrast, we use a strong supervision approach in which triples were manually annotated. Li et al. (2014) and Yu et al. (2020) also attempted to extract personal information from dialogue using a strongly supervised paradigm. However, they focused on demographic attributes as well as interpersonal relationships, which contrast with our focus on what people own and like. Li et al. (2014) used SVMs to classify relations and CRFs to perform slot filling of entities while Yu et al. (2020) used BERT to identify relations between given entities.

Generating KG triple-like Data using Language Models: Autoregressive language models have been applied to a wide range of tasks involving the generation of data with similar structures as personal attribute KG triples, including dialogue state tracking (Hosseini-Asl et al. 2020) and commonsense KG completion (Bosselut et al. 2019). The most similar application is Alt, Hübner, and Hennig (2019), which used the original GPT model (Radford and Narasimhan 2018) for relation classification. Their task formulation involves identifying a specific relation (out of around 30 possible options) for two given entities. On the other hand, our tasks seek to identify not only the relation, but also the head and tail entities, which have potentially open vocabulary requirements. Thus, the search space is significantly increased.

8 Conclusion

In conclusion, we demonstrate that personal attributes can support the personalization of open-domain social chit-chat agents and multi-task, task-oriented dialogue agents. Thus motivated, we propose the novel tasks of extracting and inferring personal attributes from dialogue and carefully analyze the linguistic demands of these tasks. To meet the challenges of our tasks, we present GenRe, a model which combines constrained attribute generation and re-ranking on top of pre-trained language models. GenRe achieves the best performance vs. established Relation Extraction baselines on the Extraction task (≥ 3.7 F1 points) as well as the more challenging INFERENCE task that involve lexical and commonsense inferences (≥ 5.1 F1 points). Together, our work contributes an important step towards realizing the potential of personal attributes in personalization of dialogue agents.

References

Alt, C.; Hübner, M.; and Hennig, L. 2019. Improving Relation Extraction by Pre-trained Language Representations. In *Automated Knowledge Base Construction (AKBC)*.

Bosselut, A.; Rashkin, H.; Sap, M.; Malaviya, C.; Çelikyilmaz, A.; and Choi, Y. 2019. COMET: Commonsense Transformers for Automatic Knowledge Graph Construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Byrne, B.; Krishnamoorthi, K.; Sankar, C.; Neelakantan, A.; Duckworth, D.; Yavuz, S.; Goodrich, B.; Dubey, A.; Kim, K.-Y.; and Cedilnik, A. 2019. Taskmaster-1: Toward a Realistic and Diverse Dialog Dataset.

Dinan, E.; Logacheva, V.; Malykh, V.; Miller, A.; Shuster, K.; Urbanek, J.; Kiela, D.; Szlam, A.; Serban, I.; Lowe, R.; Prabhumoye, S.; Black, A. W.; Rudnicky, A.; Williams, J.; Pineau, J.; Burtsev, M.; and Weston, J. 2019. The Second Conversational Intelligence Challenge (ConvAI2). arXiv:1902.00098.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Hogan, A.; Blomqvist, E.; Cochez, M.; d'Amato, C.; de Melo, G.; Gutierrez, C.; Gayo, J. E. L.; Kirrane, S.; Neumaier, S.; Polleres, A.; Navigli, R.; Ngomo, A.-C. N.; Rashid, S. M.; Rula, A.; Schmelzeisen, L.; Sequeda, J.; Staab, S.; and Zimmermann, A. 2021. Knowledge Graphs. arXiv:2003.02320.

Hosseini-Asl, E.; McCann, B.; Wu, C.-S.; Yavuz, S.; and Socher, R. 2020. A Simple Language Model for Task-Oriented Dialogue. arXiv:2005.00796.

Joshi, C. K.; Mi, F.; and Faltings, B. 2017. Personalization in Goal-Oriented Dialog. arXiv:1706.07503.

Koncel-Kedziorski, R.; Bekal, D.; Luan, Y.; Lapata, M.; and Hajishirzi, H. 2019. Text Generation from Knowledge Graphs with Graph Transformers. 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)*, 2284–2293. Minneapolis, Minnesota: Association for Computational Linguistics.

Li, A. W.; Jiang, V.; Feng, S. Y.; Sprague, J.; Zhou, W.; and Hoey, J. 2020. ALOHA: Artificial Learning of Human Attributes for Dialogue Agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 8155–8163.

Li, J.; Galley, M.; Brockett, C.; Spithourakis, G.; Gao, J.; and Dolan, B. 2016. A Persona-Based Neural Conversation Model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 994–1003. Berlin, Germany: Association for Computational Linguistics.

Li, X.; Tur, G.; Hakkani-Tür, D.; and Li, Q. 2014. Personal knowledge graph population from user utterances in conversational understanding. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, 224–229.

Lu, Y.; Srivastava, M.; Kramer, J.; Elfardy, H.; Kahn, A.; Wang, S.; and Bhardwaj, V. 2019. Goal-Oriented End-to-End Conversational Models with Profile Features in a Real-World Setting. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, 48–55. Minneapolis, Minnesota: Association for Computational Linguistics.

Luo, L.; Huang, W.; Zeng, Q.; Nie, Z.; and Sun, X. 2019. Learning Personalized End-to-End Goal-Oriented Dialog. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 6794–6801.

Majumder, B. P.; Jhamtani, H.; Berg-Kirkpatrick, T.; and McAuley, J. 2020. Like hiking? You probably enjoy nature: Persona-grounded Dialog with Commonsense Expansions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 9194–9206. Online: Association for Computational Linguistics.

Mazaré, P.-E.; Humeau, S.; Raison, M.; and Bordes, A. 2018. Training Millions of Personalized Dialogue Agents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2775–2779. Brussels, Belgium: Association for Computational Linguistics.

Miller, A. H.; Feng, W.; Fisch, A.; Lu, J.; Batra, D.; Bordes, A.; Parikh, D.; and Weston, J. 2017. ParlAI: A Dialog Research Software Platform. *arXiv preprint arXiv:1705.06476*.

Mo, K.; Li, S.; Zhang, Y.; Li, J.; and Yang, Q. 2017. Personalizing a Dialogue System with Transfer Reinforcement Learning. arXiv:1610.02891.

Moon, S.; Shah, P.; Kumar, A.; and Subba, R. 2019. OpenDialKG: Explainable Conversational Reasoning with Attention-based Walks over Knowledge Graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 845–854. Florence, Italy: Association for Computational Linguistics.

Nayak, T.; and Ng, H. T. 2020. Effective Modeling of Encoder-Decoder Architecture for Joint Entity and Relation Extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 8528–8535.

Nivre, J.; de Marneffe, M.-C.; Ginter, F.; Goldberg, Y.; Hajič, J.; Manning, C. D.; McDonald, R.; Petrov, S.; Pyysalo, S.; Silveira, N.; Tsarfaty, R.; and Zeman, D. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 1659–1666. Portorož, Slovenia: European Language Resources Association (ELRA).

Pappu, A.; and Rudnicky, A. 2014. Knowledge Acquisition Strategies for Goal-Oriented Dialog Systems. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 194–198. Philadelphia, PA, U.S.A.: Association for Computational Linguistics.

Pei, J.; Ren, P.; and de Rijke, M. 2021. A Cooperative Memory Network for Personalized Task-oriented Dialogue Systems with Incomplete User Profiles. *arXiv preprint arXiv:2102.08322*.

Petroni, F.; Rocktäschel, T.; Riedel, S.; Lewis, P.; Bakhtin, A.; Wu, Y.; and Miller, A. 2019. Language Models as Knowledge Bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2463–2473. Hong Kong, China: Association for Computational Linguistics.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program*, 14(3): 130–137.

Qian, Q.; Huang, M.; Zhao, H.; Xu, J.; and Zhu, X. 2018. Assigning Personality/Profile to a Chatting Machine for Coherent Conversation Generation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 4279–4285. International Joint Conferences on Artificial Intelligence Organization.

Radford, A.; and Narasimhan, K. 2018. Improving Language Understanding by Generative Pre-Training.

Rastogi, A.; Zang, X.; Sunkara, S.; Gupta, R.; and Khaitan, P. 2019. Towards Scalable Multi-domain Conversational Agents: The Schema-Guided Dialogue Dataset. *arXiv preprint arXiv:1909.05855*.

Roller, S.; Dinan, E.; Goyal, N.; Ju, D.; Williamson, M.; Liu, Y.; Xu, J.; Ott, M.; Shuster, K.; Smith, E. M.; Boureau, Y.-L.; and Weston, J. 2020. Recipes for building an open-domain chatbot. arXiv:2004.13637.

Shalyminov, I.; Sordoni, A.; Atkinson, A.; and Schulz, H. 2020. Fast Domain Adaptation For Goal-Oriented Dialogue Using A Hybrid Generative-Retrieval Transformer. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Speer, R.; Chin, J.; and Havasi, C. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, 4444–4451. AAAI Press.

Tigunova, A.; Yates, A.; Mirza, P.; and Weikum, G. 2019. Listening between the Lines: Learning Personal Attributes from Conversations. arXiv:1904.10887.

Tigunova, A.; Yates, A.; Mirza, P.; and Weikum, G. 2020. CHARM: Inferring Personal Attributes from Conversations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 5391–5404. Online: Association for Computational Linguistics.

Wadden, D.; Wennberg, U.; Luan, Y.; and Hajishirzi, H. 2019. Entity, Relation, and Event Extraction with Contextualized Span Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 5784–5789. Hong Kong, China: Association for Computational Linguistics.

Welleck, S.; Weston, J.; Szlam, A.; and Cho, K. 2019. Dialogue Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3731–3741. Florence, Italy: Association for Computational Linguistics.

Wu, C.-S.; Madotto, A.; Lin, Z.; Xu, P.; and Fung, P. 2019. Getting To Know You: User Attribute Extraction from Dialogues. *arXiv preprint arXiv:1908.04621*.

Ye, H.; Zhang, N.; Deng, S.; Chen, M.; Tan, C.; Huang, F.; and Chen, H. 2021. Contrastive Triple Extraction with Generative Transformer. arXiv:2009.06207.

Yu, D.; Sun, K.; Cardie, C.; and Yu, D. 2020. Dialogue-Based Relation Extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4927–4940. Online: Association for Computational Linguistics.

Zang, X.; Rastogi, A.; Sunkara, S.; Gupta, R.; Zhang, J.; and Chen, J. 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*, 109–117.

Zhang, S.; Dinan, E.; Urbanek, J.; Szlam, A.; Kiela, D.; and Weston, J. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2204–2213. Melbourne, Australia: Association for Computational Linguistics.

Zhang, Y.; Zhong, V.; Chen, D.; Angeli, G.; and Manning, C. D. 2017. Position-aware Attention and Supervised Data Improve Slot Filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, 35–45.

Zheng, Y.; Chen, G.; Huang, M.; Liu, S.; and Zhu, X. 2020a. Personalized Dialogue Generation with Diversified Traits. arXiv:1901.09672.

Zheng, Y.; Zhang, R.; Huang, M.; and Mao, X. 2020b. A Pre-Training Based Personalized Dialogue Generation Model with Persona-Sparse Data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 9693–9700.

Zhou, X.; Zhang, Y.; Cui, L.; and Huang, D. 2020. Evaluating Commonsense in Pre-Trained Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 9733–9740.

A Appendix

A.1 Blender Fine-tuning Details

Finetuning hyperparameters are taken from https://parl.ai/projects/recipes/, with the exception of validation metric changed to Hits@1. Each fine-tuning epoch takes 1.5 hours on a Nvidia V100 GPU. We only prepend personal attributes before system utterances but not user utterances. Metrics are for the validation set because test set was not available. All experiments were conducted using ParlAI (Miller et al. 2017).

A.2 Task Analysis Details

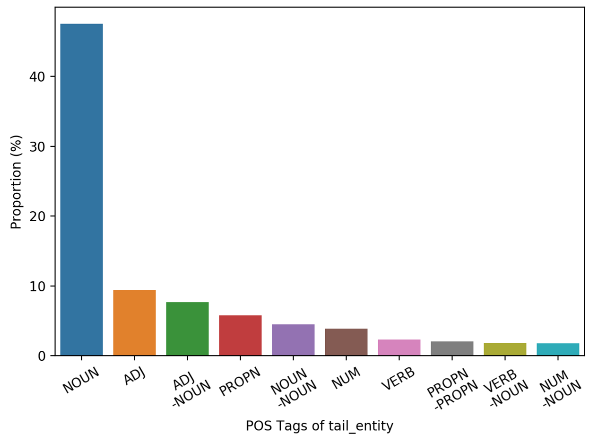

Figure 4: Bar plot for 10 most common POS tags of tail entities.

A.3 Details of Transformations to Link Tail Entity to Sentence

ConceptNet_related: All words in the tail entity can be found in the 100 most related words to each sentence word based on embedding distance on ConceptNet

ConceptNet_connect: All words in the tail entity can be found in the 100 words that have the highest-weighted edge with each sentence word on ConceptNet.

WordNet_synonym: All words in the tail entity can be found in the synonyms of every synset of each sentence word on WordNet.

WordNet_hypernym: All words in the tail entity can be found in the hypernyms of every synset of each sentence word on WordNet

WordNet_hyponym: All words in the tail entity can be found in the hyponyms of every synset of each sentence word on WordNet

Same_stem: All words in the sentence and tail entity are stemmed using a Porter Stemmer (Porter 1980) before searching for the tail entity in the sentence

A.4 Generator Details

GPT-2-small was used. Additional special tokens including the control tokens ([HEAD], [RELN], [TAIL]) as well as relation tokens were added into the tokenizer. Beam search decoding (beam size = 10) was used at inference time. GPT2-small was accessed from HuggingFace Transformers library with 125M parameters, context window 1024, 768-hidden, 768-hidden, 12-heads, dropout = 0.1. AdamW optimizer was used with $\alpha = 7.5 * 10^{-4}$ for the EXTRACTION dataset and $\alpha = 2.5 * 10^{-3}$ for the INFERENCE dataset, following a uniform search using F1 as the criterion at intervals of $\{2.5, 5, 7.5, 10\} * 10^n; -5 \leq n \leq -3$. Learning rate was linearly decayed (over a max epoch of 8) with 100 warm-up steps. Each training epoch took around 0.5 hour on an Nvidia V100 GPU with a batch size of 16. Validation was done every 0.25 epochs during training. 5 different seeds (40-44) were set for 5 separate runs.

A.5 Reranker Details

BERT-base-uncased was used. Additional special tokens including the control tokens ([HEAD], [RELN], [TAIL]) as well as relation tokens were added into the tokenizer. BERT-base-uncased was accessed from HuggingFace Transformers library (with 12-layer, 768-hidden, 12-heads, 110M parameters, dropout = 0.1). The choice of the base model was made to have fairness of comparison with baseline models in terms of model size. AdamW optimizer was used with $\alpha = 5 * 10^{-6}$, following a uniform search using F1 as the criterion at intervals of $\{2.5, 5, 7.5, 10\} * 10^n; -6 \leq n \leq -3$. Learning rate was linearly decayed (over a max epoch of 8) with 100 warm-up steps. Each training epoch took around 1 hour on an Nvidia V100 GPU with a batch size of 10.Validation was done every 0.25 epochs during training. 5 different seeds (40-44) were set for 5 separate runs.