Controlled Data Generation via Insertion Operations for NLU

Manoj Kumar[1], **Haidar Khan**[1], **Yuval Merhav**[1], **Wael Hamza**[1]
Anna Rumshisky[1,2] **Rahul Gupta**[1]
[1]Alexa AI, Amazon
[2]Department of Computer Science, University of Massachusetts Lowell
{abithm,khhaida,merhavy,waelhamz,gupra}@amazon.com
arum@cs.uml.edu

Abstract

Use of synthetic data is rapidly emerging as a realistic alternative to manually annotating real data for industry-scale model building. Manual data annotation is slow, expensive and not preferred for meeting customer privacy expectations. Further, commercial natural language applications are required to support continuously evolving features as well as newly added experiences. To address these requirements, we propose a targeted synthetic data generation technique by inserting tokens into a given semantic signature. The generated data are used as additional training samples in the tasks of intent classification and named entity recognition. We evaluate on a real-world voice assistant dataset, and using only 33% of the available training set, we achieve the same accuracy as training with all available data. Further, we analyze the effects of data generation across varied real-world applications and propose heuristics that improve the task performance further.

1 Introduction

One of the common challenges to deploying natural language understanding (NLU) techniques at scale in commercial applications is the necessity for continuous annotation of user data. Models can then be re-trained and updated to capture new usage patterns. This process is expensive, labor intensive, and time-consuming.

At an age when user privacy is becoming the focus of increased concern in all AI applications, manual review of user data normally required for such annotation becomes highly undesirable. Consequently, multiple initiatives are undertaken towards minimizing the amount of human annotations needed for training NLU models.

Data augmentation (DA) refers to strategies that aim at increasing the diversity of training samples without explicitly collecting new data. In this work, we present a generative model that is used to generate labeled synthetic data. Given a set of utterance templates [1] that we construct from a limited amount of labeled data, our goal is to generate synthetic utterances and augment the original (reduced) training data, with the objective of improving the model robustness and performance.

We focus on the special case where the synthetic data must retain a specific fine-grained interpretation of the original utterance, such as token-level annotation. For example, we would like to control the composition of entities (and their combinations) in the training data when expanding to new features while maintaining NLU model performance. In our proposed approach, we control the desired annotation by re-framing the generation process as insertion rather than left-to-right generation. We preserve the desired entities in the synthetic example by including them in the model's input during generation and introduce methods to explicitly prevent entity corruption during the generation process.

Our contributions are as follows: (i) We propose a novel synthetic data generation technique using insertion transformers that allows for token-level control over the generated synthetic utterance. (ii) We demonstrate the usefulness of the proposed approach for NLU model building. Our model which is trained using a limited fraction of user data combined with synthetic data matches the performance of a model trained with the entire real data. (iii) We apply domain-specific heuristics to improve the quality of synthetic data, which would further improving task performance.

2 Background

Our NLU models are responsible for interpreting the corresponding domain, intent and actionable slots of customer utterances. These categories are modularized, i.e., utterances belonging to a partic-

[1]For the purposes of this work, we define a *template* as the sequence of intent label, slot labels and slot values.

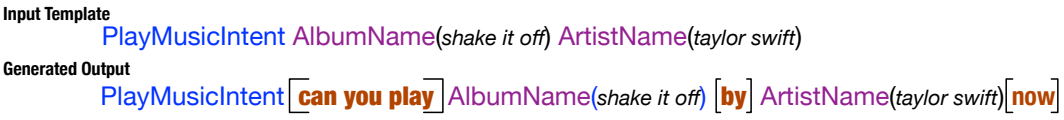

Figure 1: An input template to GIT with its generated labeled utterance. The output maintains the original template but inserts new phrases (shown within brackets) between the slots.

ular domain (e.g., Books) are supported by a specific set of intents (e.g., PlayBook) and actionable slots (e.g., BookName), and served by the domain-specific intent classification (IC) and named entity recognition (NER) models. In this work we experiment and evaluate IC and NER tasks across multiple domains. We explore the use of synthetic data as an additional source for training the models of these domains.

While a number of data augmentation techniques for natural language have been proposed, ranging from token-level perturbations (Wei and Zou, 2019) to paraphrase generation (Chen et al., 2020; Jolly et al., 2020) and auto-regressive models (Ding et al., 2020; Malandrakis et al., 2019; Anaby-Tavor et al., 2020; Kumar et al., 2020), these techniques can not be directly applied to token labeling tasks such as NER. Specifically, synthetic data generation for NER involves two additional challenges: (1) **Label preservation**: producing correct token-level annotation in the generated utterances, e.g., in Figure 1 "shake it" may be incorrectly labeled as *AlbumName* instead of "shake it off" (2) **Entity control**: controlling slot-type and slot-values in the synthetic data. e.g., we would like to generate requests for other artists and albums. The first challenge is typically addressed by a label projection approach (Ehrmann et al., 2011) or semi-supervised learning, however this is known to introduce errors in the resulting annotation. To handle the second challenge, methods such as (Jolly et al., 2020; Malandrakis et al., 2019) input the desired slot types and values to the model but cannot force the generator to include these slots in the synthetic example.

3 Methodology

3.1 Synthetic Data Augmentation with GIT

Our approach, dubbed generative insertion transformer (GIT) is based on non-autoregressive insertion transformer model introduced in (Stern et al., 2019). Previously, it has been shown that these models can be used effectively for generating annotations; given an utterance generate the correct NLU interpretation (intent and slots) using insertion operations (Zhu et al., 2020). In this work, we extend the idea to solve the inverse NLU problem; given a template produce a valid labeled utterance that matches the annotation (Figure 1).

The insertion transformer is a generative model in which the decoder generates a sequence by inserting tokens between previously generated tokens. We adopt this idea to insert carrier tokens (token without an entity label) between the labels in the template in an iterative manner. (An example of template is provided in Figure 1, and the insertion process is illustrated in Figure 2). The insertion process at each position in the utterance is independent of every other position and stops when the EOS token is generated at all positions, resulting in a fully annotated synthetic utterance that can be directly augmented with real data for model building purpose. We now describe the stages involved in building and deploying GIT.

Pre-Training: We pre-train GIT using BERT encoder (Devlin et al., 2019) and KERMIT (Chan et al., 2019) objective on an unsupervised LM task: given a sentence with masked tokens, GIT is trained to insert the masked tokens. We test two configurations (1) Pre-training using only English Wikipedia[2] (**wiki**), and (2) Pre-training using an internal corpus of 800M unlabeled utterances randomly sampled from de-identified Alexa requests, using English Wikipedia pre-trained models as initialization (**wiki+in-domain**).

Fine-Tuning: We fine-tune the pre-trained GIT models for each domain (e.g., Books) using annotated real data (reduced). Table 1 shows a few data samples and derived templates. For each utterance, we provide the template as model input and the complete (annotated) utterance as output. During training, at each insertion slot, we have multiple candidate tokens from the ground truth unlike autoregressive generation which entails a single token per generation step. For example, in Figure 2 the tokens "can", "you" and "play" can be inserted between "PlayMusicIntent" and "Album(". Hence, we cannot use the traditional cross-entropy loss

[2] https://en.wikipedia.org

Table 1: Representative examples for labeled utterances and derived templates from 3 domains. Carrier tokens (slot label "|Other") are removed from the labeled utterance and intent names added to create a template

ID	Domain	Intent	Labeled Utterance	Derived Template
1	Recipes	SearchRecipe	find|Other breakfast|Meal recipe|InstructionType please|Other	breakfast|Meal recipe|InstructionType
2	Books	NavigateBooks	skip|Other to|Other chapter|SectionName one|SectionNumber	chapter|SectionName one|SectionNumber
3	Home	GetSettingsDetails	what's|Other the|Other heat|Setting set|Other at|Other	heat|Setting

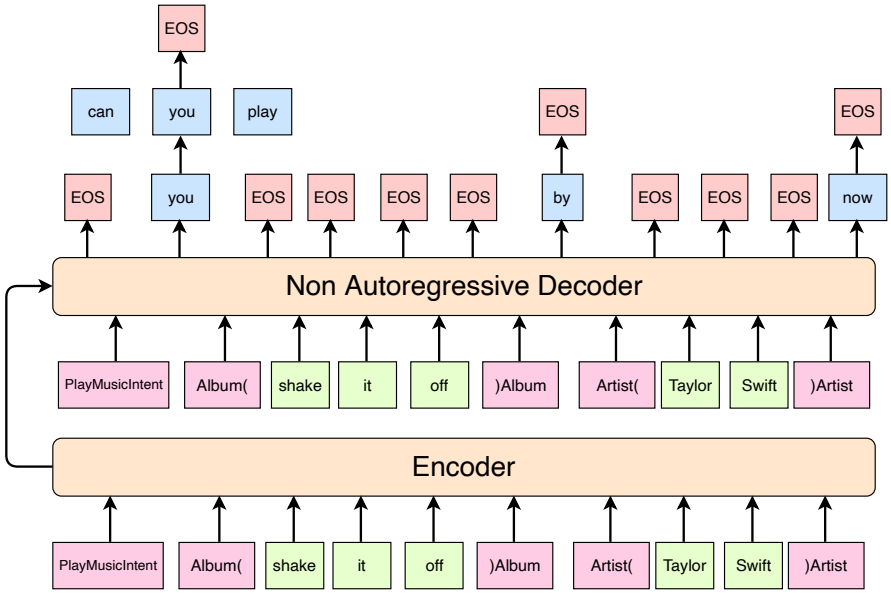

Figure 2: A generation example with GIT. An utterance template is provided as input to the decoder. The decoder generates one or more (carrier) tokens between every two input tokens and stops the generation process when the End of Sequence (EOS) token is generated (we set maximum number of non-EOS generated tokens to three). The model learns to only generate EOS tokens within entity tokens (e.g., "shake it off") but this is not enforced. We discard generated examples when it is not the case (<0.01%).

and instead compute KL divergence between the predicted token distribution and the ground truth distribution at each position, and use the mean divergence over all insertion slots as the training loss (Zhu et al., 2020). The ground truth distribution sets non-candidate token probabilities to 0 and uniformly weighs all candidate token probabilities.

Generation: To generate synthetic data for NLU, we first construct a template that contains the desired intent, slot types, and slot values for the synthetic example. This priming sequence is provided as an input to the decoder, which inserts carrier tokens in an iterative manner to form a coherent utterance. The generation process is shown in Figure 2 and addresses both the label projection and entity control challenges. Templates used in inference are constructed from the reduced real data.

4 Experimental Setup

In order to study the effectiveness of synthetically generated data, we evaluate NLU model performance in reduced data regime. For each domain, we build multiple IC-NER models using all real data, a reduced set of real data and combination of real and synthetic data. All models within a domain share the same training hyper-parameters, including architecture and encoder. They differ only in training data composition. Similar to (Ding et al., 2020), we limit the focus of this work to synthetic data generation and defer hyper-parameter optimization to future work. We use Apache MXNet (Chen et al., 2015) to build both GIT and IC-NER models in this work.

Full: This baseline is trained using all real data and default training hyper-parameters for each domain. This setup reflects the current performance of NLU models in production and serves as an estimate for lower bound in error metric for all other models.

Reduced: We train another baseline using **one-third** of real data. Our reduction of two-thirds of the data is motivated by a privacy control feature allowing customers to delete their data. Given the trends, we estimated a worst case drop of 67% in our annotated data before it can be refilled with

more human annotations. To simulate this worst case scenario, we randomly downsample across all utterances.

Duplicate: To reduce the impact of hyper-parameters we also train a model with the **Reduced** set duplicated to reach the full training size. We refer to it as **Duplicate**. We note that duplication has been used as a baseline for data augmentation in (Estabrooks et al., 2004; Kumar et al., 2019; Wei and Zou, 2019)

EDA: Easy Data Augmentation (EDA) consists of four simple operations: synonym replacement, random insertion, random swap, and random deletion. EDA has shown to be a strong baseline, outperforming complex model-based baselines particularly for small datasets (Wei and Zou, 2019). Similar to GIT, EDA can provide control and flexibility over the type of data generated, which is a key requirement from our users.

GIT: (ours). We use the **Reduced** set to fine-tune domain-specific GIT models and also as input templates during inference, with fixed hyper-parameters. During inference, we control the number of generated synthetic utterances which is augmented with **Reduced** set. We test two configurations: in **GIT_50**, the fraction of synthetic to real data is set to 50% while with **GIT_200**, the fraction of synthetic to real data is set to 200%. In the former, synthetic data size is kept smaller than real data while in latter, we add enough synthetic data to compensate for removed data.

4.1 Confidence filtering of synthetic data selection

Not all synthetic utterances may be useful for model training, such as duplicates of real utterances, semantically incorrect samples ("play album" instead of "buy album" for BuyAlbumIntent), etc. A handful of previous approaches have investigated filtering synthetic utterances before augmentation: using influence functions (Yang et al., 2020), reinforcement learning (Bhattarai et al., 2020), etc. In this work, we use the confidence score obtained using **Reduced** models to filter synthetic utterances. Assuming M represents **Reduced** model, we predict labels $\hat{y}$ for a synthetic utterance **x** using M, i.e

$$\hat{y}, c = M(\mathbf{x}) \quad (1)$$

Here, c is a confidence score derived as the unweighted mean of IC and NER scores and scaled to (0,1). We select **x** for augmentation if (i) $\hat{y} = y$ and (ii) $c \in (t_{low}, t_{high})$, where y is the ground truth label of **x** available from its template, and t_{low} and t_{high} are threshold hyper-parameters >0.5. Hence, we select those synthetic samples which are correctly labeled by M and avoid incorrect utterances (t_{low}) and duplicates (t_{high}). We consider $y = \hat{y}$ if the predicted intent label and all slot labels exactly match with the ground truth.

4.2 Evaluation

We evaluate the models on each domain's test set. For each model, we use weighted semantic error rate (SemER$_w$ Su et al. (2018)) to jointly evaluate IC-NER performance. SemER is defined as the ratio of Leveshtein distance between reference and hypothesis labels, and total count of reference labels. We concatenate the intent and slot labels to arrive at an utterance-level label. We weigh each domain's SemER by its test utterance count and report the mean SemER (SemER$_w$) for each model. We report relative performance gains with respect to **Full** baseline: we only report relative performance as we are not allowed to publish absolute performance numbers.

5 Results

Table 2 shows relative SemER$_w$ across different methods (lower is better). We can see that SemER$_w$ for **Reduced** model increases 2.42%. Interestingly, Sports domain improves in SemER (>5%) when reducing real data (**Reduced** vs **Full**; Figure 3): We found that Sports is a relatively smaller domain and tends to have noisier training data (Section 6.3). While **Duplicate** and **EDA** do not improve over **Reduced**, **GIT_50 (wiki+in-domain)** achieves the same error rate as training with all available data. Not surprisingly, using in-domain data during pre-training **GIT_N (wiki+in-domain)** improves results significantly over pre-training only on Wikipedia **GIT_N (wiki)**.

6 Discussion

While the overall regression appears modest, there exists significant variation among domains (Figure 3). We can see that GIT improves SemER only among certain domains when compared to **Reduced** (e.g., Music but not Sports). In general, domains with relatively higher traffic exhibit moderate regression (<5%). Recall that for simplicity we use the same hyper-parameters across all domains.

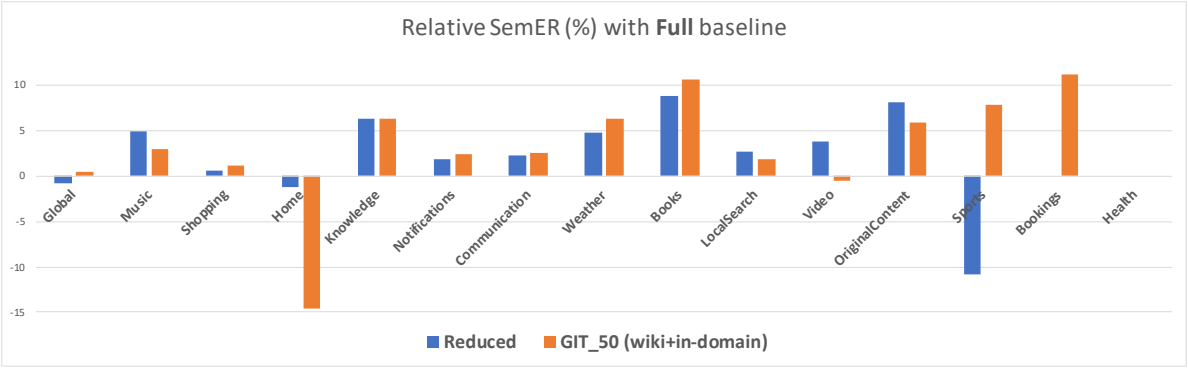

Figure 3: Relative change in per-domain SemER comparing **Reduced** and **GIT_50_wiki+in-domain** to **Full**. Domains are sorted according to decreasing traffic volume

Table 2: Relative SemER_w (weighted mean, by traffic volume) for baselines and GIT models for different pre-training corpora and synthetic data sizes. All results are reported relative to **Full** baseline

Full	Reduced	Duplicate	EDA	GIT_200 (wiki)	GIT_50 (wiki)	GIT_200 (wiki+in-domain)	GIT_50 (wiki+in-domain)
0%	+2.38%	+2.57%	+3.07%	+9.94%	+1.27%	+2.66%	**-0.05%**

6.1 Value of Synthetic Data

While we observe that **GIT_50 (wiki)** and **GIT_50 (wiki+in-domain)** configurations provide overall improvements over **Reduced**, we investigate whether data reduction effects are related to improvements with synthetic data addition. Specifically, using the null hypothesis that the relative SemER (%) between data reduction (**Full** $\rightarrow$ **Reduced**) and data addition (**Reduced** $\rightarrow$ **GIT**) are not related, we estimate the Pearson correlation between them using two-tailed t-distribution. In Table 3, we present the correlation coefficients (**r**) along with significance information. We notice in all configurations that a domain's SemER improvement with added synthetic data is inversely proportional to the regression with data reduction. In other words, domains which are most affected by data reduction benefit from adding synthetic data and vice versa, irrespective of the source and quantity of synthetic data.

Table 3: Pearson correlation coefficient ($p < 0.01$**) for domain-level relative SemER between (Full $\rightarrow$ Reduced) and (Reduced $\rightarrow$ GIT)

Method	**r**
GIT_200 (wiki)	-0.711**
GIT_50 (wiki)	-0.751**
GIT_200 (wiki+in-domain)	-0.234
GIT_50 (wiki+in-domain)	-0.700**

6.2 Confidence Filtering

Among domains where **GIT_50 (wiki+in-domain)** performance is worse than **Reduced**, we notice that there exist real utterances which lack the appropriate context necessary for GIT inference and are more error-prone, such as those without an entity slot (E.g., "stop|Other", "turn|Other off|Other"). As described in Section 4.1, we implement confidence filtering for the top 5 domains with highest SemER degradations for GIT (Figure 3) and present results in Table 5. Based on empirical observations, we choose (t_{low}, t_{high}) = (0.5, 0.85). We find that confidence-filtering results in consistent SemER improvements across domains compared to **GIT**, with upto 12.85% relative improvement for Bookings. When combined with confidence filtering, **GIT** marginally improves over the **Reduced** baseline for these 5 domains.

6.3 Synthetic Data Diversity

In this section, we analyze the generated synthetic data using quantitative metrics and qualitative examples. We use the distinct-n metric (introduced by Li et al. (2016)), which measures the fraction of unique n-grams to the n-gram count (higher metric indicates more diverse utterances). We compare distinct-2 and distinct-3 metrics between real and synthetic utterances for domains with highest (Bookings, Books, Sports) and lowest (Home, Video, Health) relative SemER in Table 6. We notice a clear decrease in token diversity in synthetic data among former domains and increase in token diversity among latter domains. This hints at the usefulness of distinct-n as a measure for predicting value of synthetic data for IC-NER model building.

Table 4: Some representative utterance templates and generated synthetic utterances. Tokens in orange represent carrier tokens which are replaced by tokens in blue during synthetic data generation by GIT

Domain	ID	Real utterance	Synthetic utterance
Video	1	youtube|AppName denis|VideoName daily|VideoName half|VideoName hour|VideoName song|MediaType	search|Other youtube|AppName for|Other denis|VideoName daily|VideoName half|VideoName hour|VideoName song|MediaType
	2	youtube|AppName baby|VideoName car|VideoName	search|Other on|Other youtube|AppName for|Other baby|VideoName car|VideoName
	3	find|Other pineola|VideoName lucinda|ArtistName william|ArtistName	search|Other for|Other pineola|VideoName by|Other lucinda|ArtistName william|ArtistName
	4	show|VisualModeTrigger the|Other video|MediaType nursery|VideoName rhymes|VideoName	show|VisualModeTrigger me|Other a|Other video|MediaType of|Other nursery|VideoName rhymes|VideoName
Sports	1	can|Other you|Other give|Other me|Other the|Other sports|Other news|Other of|Other the|Other day|Other	tell|Other me|Other a|Other sports|Other updates|Other
	2	can|Other you|Other give|Other me|Other the|Other latest|SortType sports|Other headlines|Other	what's|Other the|Other latest|SortType in|Other sports|Other update|Other
	3	what's|Other the|Other latest|SortType in|Other the|Other sports|Other program|Other	what's|Other the|Other latest|SortType in|Other sports|Other update|Other

Table 5: Relative SemER (compared with **Full**) results using confidence-filtered synthetic utterances for 5 domains with highest regressions

Domain	Reduced	GIT	+ Conf. filtering
Bookings	0%	11.1%	-3.1%
Books	8.8%	10.5%	7.7%
Sports	-10.8%	7.8%	-1.3%
Weather	4.8%	6.3%	3.8%
Knowledge	6.3%	6.3%	6.3%
Total (Weighted)	6.3%	8.4%	5.6%

Table 6: Quantitative estimate of n-gram diversity of real and synthetic utterances as measured with distinct-2 and distinct-3 metrics for each domain. Relative diversity is provided for comparison purposes.

Domain	Distinct-2			Distinct-3		
	Real	Syn	Rel(%)	Real	Syn	Rel(%)
Bookings	0.119	0.108	-9.1	0.194	0.174	-10.4
Books	0.097	0.055	-43.9	0.197	0.116	-40.81
Sports	0.024	0.006	-77.26	0.047	0.010	-78.27
Health	0.076	0.086	13.23	0.139	0.154	10.95
Video	0.072	0.095	31.42	0.158	0.188	18.88
Home	0.018	0.021	18.47	0.045	0.050	10.33

We further discuss two domains which show the highest magnitude of diversity change.

Sports: Similar to typical real-world tasks, Sports domain contains class-imbalanced training data (ranging from $\mathcal{O}(10^2)$ to $\mathcal{O}(10^4)$ samples per intent), ambiguous short utterances ($\sim$65% of utterances in a minority intent contain a single-token and repeat in a majority intent) and 95.3% of utterances do not contain any tokens with slot labels. In addition to reduced token diversity, these factors contribute to shorter synthetic utterances on average (Mean utterance length: real = 3.73 vs syn = 2.32). Representative examples are provided in Table 4: utterances 2 and 3 result in the same synthetic utterance even though their tokens are different.

Video: From Table 4, we observe that GIT enhances the semantics of real utterances by appropriate carrier token insertions, specifically for utterances that search for video titles. In example 1, GIT inserts the tokens "search" and "for" which convey the meaning of the utterance more clearly and disambiguate tokens representing the application and video title. Similarly, in example 3 GIT inserts the correct preposition "by" between "peneola" and "lucinda william" using their slot label information. We hypothesize that such synthetic utterances are a better representation of token-level labels when compared to corresponding real utterances, and better assist NLU model building.

7 Limitations

Since our primary focus in this work was developing insertion transformers for DA, we did not explore extensive hyper-parameter optimization while building IC-NER models using combination of real and synthetic data. For example, we observed that adding the same fraction of synthetic data results in significant performance variations across domains, suggesting that per-domain parameter optimization may be yield improved performance.

8 Conclusion

We demonstrated DA using GIT as a feasible data generation technique to mitigate reduced annotation volumes for IC and NER tasks. We showed that NLU models trained on 33% real data and synthetic data perform similar to models trained on full real data. Further, on domains with highest SemER regressions we improved the quality of synthetic data by filtering them with model confidence scores. Among domains which benefit from synthetic data, we showed that appropriate carrier token insertion enhances utterances' semantics and their value as

training samples. In the future, we would like to explore data generation with entities replaced through knowledge base sampling. Such finer control over entities better supports new feature expansion and enhances customer privacy.

9 Ethical Considerations

Risk: In this work, we have not controlled the entities in utterance templates during generation. This presents a risk of private information propagating into the synthetic data. We note that the entities themselves are not introduced during generation, but carried over from real data. As mentioned in Section 8, entity control methods such as considered in the present work with GIT can prevent such identifiable information from being exposed to model training.

Data Protection: There are multiple guardrails to safeguard customer data in our organization. In addition, we remove all metadata and personal identifiable information (PII) from utterances before using them for NU model building and synthetic data generation with GIT in this work.

References

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? Deep Learning to the rescue! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7383–7390.

Binod Bhattarai, Seungryul Baek, Rumeysa Bodur, and Tae-Kyun Kim. 2020. Sampling strategies for GAN synthetic data. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2303–2307.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. Kermit: Generative insertion-based modeling for sequences. *ArXiv*, abs/1906.01604.

Hannah Chen, Yangfeng Ji, and David Evans. 2020. Finding Friends and flipping frenemies: Automatic paraphrase dataset augmentation using graph theory. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4741–4751. Association for Computational Linguistics.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. DAGA: Data augmentation with a generation approach for low-resource tagging tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6045–6057. Association for Computational Linguistics.

Maud Ehrmann, Marco Turchi, and Ralf Steinberger. 2011. Building a multilingual named entity-annotated corpus using annotation projection. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 118–124.

Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. 2004. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36.

Shailza Jolly, Tobias Falke, Caglar Tirkaz, and Daniil Sorokin. 2020. Data-efficient paraphrase generation to bootstrap intent classification and slot labeling for new features in task-oriented dialog systems. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 10–20, Online. International Committee on Computational Linguistics.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26. Association for Computational Linguistics.

Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and Wlliam Campbell. 2019. A closer look at feature space data augmentation for few-shot intent classification. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 1–10. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Nikolaos Malandrakis, Minmin Shen, Anuj Goyal, Shuyang Gao, Abhishek Sethi, and Angeliki Metallinou. 2019. Controlled text generation for data augmentation in intelligent artificial agents. In *Proceedings of the 3rd Workshop on Neural Generation*

and Translation, pages 90–98. Association for Computational Linguistics.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985. PMLR.

Chengwei Su, Rahul Gupta, Shankar Ananthakrishnan, and Spyros Matsoukas. 2018. A re-ranker scheme for integrating large scale nlu models. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 670–676. IEEE.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. *CoRR*, abs/1901.11196.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025. Association for Computational Linguistics.

Qile Zhu, Haidar Khan, Saleh Soltan, Stephen Rawls, and Wael Hamza. 2020. Don't parse, insert: Multilingual semantic parsing with insertion based decoding. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 496–506, Online. Association for Computational Linguistics.