Paraphrase Generation for Semi-Supervised Learning in NLU

Eunah Cho
Amazon, Alexa AI
eunahch@amazon.com

He Xie
Amazon, Alexa AI
hexie@amazon.com

William M. Campbell
Amazon, Alexa AI
cmpw@amazon.com

Abstract

Semi-supervised learning is an efficient way to improve performance for natural language processing systems. In this work, we propose Para-SSL, a scheme to generate candidate utterances using paraphrasing and methods from semi-supervised learning. In order to perform paraphrase generation in the context of a dialog system, we automatically extract paraphrase pairs to create a paraphrase corpus. Using this data, we build a paraphrase generation system and perform one-to-many generation, followed by a validation step to select only the utterances with good quality. The paraphrase-based semi-supervised learning is applied to five functionalities in a natural language understanding system.

Our proposed method for semi-supervised learning using paraphrase generation does not require user utterances and can be applied prior to releasing a new functionality to a system. Experiments show that we can achieve up to 19% of relative semantic error reduction without an access to user utterances, and up to 35% when leveraging live traffic utterances.

1 Introduction

Task-oriented dialog systems are used frequently, either providing mobile support (e.g. Siri, Bixby) or at-home service (e.g. Alexa, Google Home). Natural language understanding (NLU) technology is one of the components for dialog systems, producing interpretation for an input utterance. Namely, an NLU system takes recognized speech input and produces intents, domains, and slots for the utterance to support the user request (Tur and De Mori, 2011). For example, for a user request "turn off the lights in living room," the NLU system would generate domain *Device*, intent *LightControl*, and slot values of "off" for *OffTrigger* and "living room" for *Location*. In this work, we define *functionality* as a dialog system's capability given NLU output (e.g., turning off a light, playing a user's playlist).

It is crucial for applications to add support for new functionalities and improve them continuously. An efficient method for this is semi-supervised learning (SSL), where the model learns from both unlabeled as well as labeled data. One SSL method for NLU is to find functionality-relevant user utterances in live traffic and use them to augment the training data. In this work, we explore an alternative SSL approach "Para-SSL," where we generate functionality-relevant utterances and augment them by applying a conservative validation. To generate functionality-relevant utterances, we use paraphrasing, a task to generate an alternative surface form to express the same semantic content (Madnani and Dorr, 2010). Paraphrasing has been used for many natural language processing (NLP) tasks to additionally generate training data (Callison-Burch et al., 2006).

We view the generation work as a translation task (Quirk et al., 2004; Bannard and Callison-Burch, 2005), where we *translate* an utterance into its paraphrase that supports the same functionality. In our task, it is crucial to perform one-to-many generation so that we can obtain a bigger candidate pool for utterance augmentation. In this work, we use beam search to generate n-best list from paraphrase generation model. We then apply a validation step for utterances in the generated n-best list and augment the ones that could be successfully validated.

In order to model paraphrases that fit to the style of dialog system, we build a paraphrase corpus for NLU modeling by automatically extracting paraphrases in terms of NLU functionality. Experiments on five functionalities of our dialog system show that we can achieve up to 35% of relative error reduction by using generated paraphrases for semi-supervised learning.

Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation (NeuralGen), pages 45–54
Minneapolis, Minnesota, USA, June 6, 2019.

2 Related Work

SSL has been used in various tasks in NLP with self-training (Ma et al., 2006; Tur et al., 2005; McClosky et al., 2006; Reichart and Rappoport, 2007). Previous work also investigated learning representations from implicit information (Collobert and Weston, 2008; Peters et al., 2018). Oliver et al. (2018) showed that using SSL in a production setting poses a distinctive challenge for evaluation.

Paraphrase modeling has been viewed as a machine translation (MT) task in previous work. Approaches include ones based on statistical machine translation (SMT) (Quirk et al., 2004; Bannard and Callison-Burch, 2005) as well as syntax-based SMT (Callison-Burch, 2008). Mallinson et al. (2017) showed that neural machine translation (NMT) systems perform better than phrase-based MT systems in paraphrase generation tasks.

In Wang et al. (2018), authors show that paraphrase generation using the transformer leads to better performance compared to two other state-of-the-art techniques, a stacked residual LSTM (Prakash et al., 2016) and a nested variational LSTM (Gupta et al., 2018). Yu et al. (2016) showed that text generation task can be achieved using a generative network, where the generator is modeled as a stochastic policy. Later the model was explored and compared to maximum likelihood estimation, as well as scheduled sampling in Kawthekar et al. (2017). Authors noted that training generative adversarial networks (GANs) is a hard problem for textual input due to its discrete nature, which makes mini updates for models to learn difficult. Iyyer et al. (2018) proposed encoder-decoder model-based, syntactically controlled paraphrase networks to generate syntactically adversarial examples.

Paraphrase extraction using bilingual pivoting was proposed in Bannard and Callison-Burch (2005), where they assume that two English strings e_1 and e_2, whose translation in a foreign language f is the same, have the same meaning. Inspired by this, we apply monolingual pivoting based on NLU interpretations. If two strings e_1 and e_2 share the same set of NLU interpretations (represented by domain, intent and slot sets), they are considered to be paraphrases. Details will be given in Section 4.

The encoder-decoder based MT approach has been applied to generate paraphrases for additional training data for NLU (Sokolov and Filimonov, 2019). They trained the encoder on a traditional, bilingual MT task, fixed it and trained decoder for paraphrase task. Authors showed that using generated paraphrases can help to improve NLU performance for a given feature. Our work distinguishes itself from this work from two perspectives. First, we show that paraphrase generation for NLU can be modeled in a shared monolingual space by leveraging pivoting based on NLU interpretations. Second, we show that generating many variants of paraphrase and applying a validation step is an effective way to apply semi-supervised learning and improves model performance greatly.

3 Para-SSL

In this section, we describe two approaches for semi-supervised learning in NLU. The first one utilizes user utterances, while the second approach uses generated paraphrases.

3.1 Semi-supervised Learning for NLU

Our conventional semi-supervised learning approach has largely two steps: filtering and validation. We first find functionality-relevant utterances from live traffic (filtering) and augment them using the current NLU model (validation). In order to find the functionality-relevant utterances, we rely on a high-throughput, low complexity linear logistic regression classifier. To train the 1-vs-all classifier, we use available target functionality utterances as in-class examples, and the rest for out-of-class examples. As feature of the classifier, we use n-grams from the examples.

The filtered utterances are augmented and validated through NLU model. Utterances with confidence score above a threshold are added into training. Throughout the paper, we will call this approach *SSL*.

3.2 Paraphrase Generation for SSL

Another approach for SSL is to generate functionality-relevant utterances, instead of filtering them from live traffic. The SSL technique described in Section 3.1 has an advantage that the filtered utterances are indeed actual utterances from dialog system users. Thus, it ensures the quality of filtered utterances in terms of fluency and context fit for our dialog system. On the other hand, it requires live traffic utterances for the target function-

ality. Therefore, the above-mentioned SSL technique is not applicable when the functionality is not yet released.

In this work, we explore generation of functionality-relevant utterance for SSL. Generated utterances are validated in the same method as in conventional SSL, by running them through an NLU model and selecting utterances whose hypothesis confidence is higher than a threshold.

Inspired by its good performance in paraphrase generation task, we use the model constructed with self-attention encoders and decoders, known as the Transformer (Vaswani et al., 2017). Unlike other paraphrase tasks (Wang et al., 2018; Yu et al., 2016), our application requires one-to-many generation. Namely, when we input one in-class functionality utterance, we expect to have many paraphrases who are likely to invoke the same functionality. In order to generate multiple paraphrases for an input utterance, we use beam search and generate n-best lists (Tillmann and Ney, 2003), where we fix $n = 50$ in this work. Throughout this paper, we will call this approach *Para-SSL*.

3.3 Benchmarks

In order to evaluate the impact of generated paraphrases in NLU modeling we set up benchmarks on five functionalities. The details of the functionalities will be discussed in Section 6. In each benchmark, we simulate the NLU functionality development cycle by adding an increasing amount of training data on the target functionality.

The first version for each benchmark represents the *bootstrap phase*. On top of the training data for other functionalities that the dialog system supports, we have synthetically created training data for the target functionality. As the functionality is not yet launched, there is no training data coming from actual user utterances.

In the following *live phase*, we add 10%, 20%, 50%, 80%, or 100% of the annotated training data of the target functionality on top of the bootstrap phase version. We will refer to them as *annotation increments* in this paper. Using the annotation increments, we aim to simulate how support for the target functionality improves as we have more user utterances available for training.

The SSL algorithm on the benchmark is shown in Algorithm 1. The starting dialog system D is trained with the bootstrap data B for the functionality and other data T for other functionalities that it supports. As we have more annotation data available, we find and validate candidate utterances C_A. We then update D using the additional training data for the functionality. Note that the bootstrap data B is continuously used throughout the live phase in order to secure a broad support for the functionality.

Algorithm 1 Algorithm for SSL

Require: Bootstrap data B
Require: Annotated $A_i, i = \{10, 20, 50, 80, 100\}$
Require: Training data for other functionalities T
Require: Dialog system D, trained on $d = T \cup B$
1: **for** each increment A_i **do**
2: train D with $d = T \cup B \cup A_i$
3: find candidate utterances C_{Ai} from user traffic
4: hypotheses from dialog system $H \leftarrow D(C_{Ai})$ with model confidence score for each hypothesis c_{hi}
5: $\mathcal{S} \leftarrow \emptyset$
6: **for** $h_i \in H$ **do**
7: **if** $c_{hi} > \theta_{cSSL}$ **then**
8: $\mathcal{S} \leftarrow \mathcal{S} \cup h_i$
9: train D with $d = T \cup B \cup A_i \cup \mathcal{S}$, evaluate

Algorithm 2 Algorithm for Para-SSL

Require: Bootstrap data B
Require: Annotated $A_t, t = \{10, 20, 50, 80, 100\}$
Require: Training data for other functionalities T
Require: Dialog system D, trained on $d = T \cup B$
1: $\mathcal{S}_\mathcal{B} \leftarrow \emptyset$
2: **for** each input in $U = \{B, A\}$ **do**
3: **if** $U_i = B$ **then**
4: train D with $d = T \cup B$
5: **else**
6: train D with $d = T \cup B \cup U_i$
7: generate paraphrases $P \leftarrow para(U_i)$
8: hypotheses from dialog system $H \leftarrow D(P)$ with model confidence score for each hypothesis c_{hi}
9: $\mathcal{S} \leftarrow \mathcal{S}_\mathcal{B}$
10: **for** $h_i \in H$ **do**
11: **if** $c_{hi} > \theta_{cPara-SSL}$ **then**
12: $\mathcal{S} \leftarrow \mathcal{S} \cup h_i$
13: **if** $U_i = B$ **then**
14: $\mathcal{S}_\mathcal{B} \leftarrow \mathcal{S}_\mathcal{B} \cup h_i$
15: **if** $U_i = B$ **then**
16: train D with $d = T \cup B \cup \mathcal{S}$, evaluate
17: **else**
18: train D with $d = T \cup B \cup U_i \cup \mathcal{S}$, evaluate

We perform Para-SSL as shown in Algorithm 2. As Para-SSL does not require live traffic utterances, we can start augmenting more utterances using bootstrap data only. Note that during the live phase ($U_i \in A$), we continue to use the bootstrap data B (line 6). Instead of the step to find candidate utterances C_A in SSL (line 3 in Algorithm 1), Para-SSL enables generation of utterances given input group (line 7 in Algorithm 2). Both algorithms have a validation step to threshold NLU interpretations based on model score. For each an-

notation increment in Para-SSL, we can also leverage the validated data from bootstrap phase $\mathcal{S}_B$ by setting $\mathcal{S}$ to always include $\mathcal{S}_B$ (see line 9).

Based upon preliminary experiments to set $\theta_{cPara-SSL}$, we fixed $\theta_{cPara-SSL}$ at 0.9 in this work. For θ_{cSSL}, we took the model's reject threshold of each functionality. When an NLU interpretation has a confidence score lower than the reject threshold, it will not be accepted by the downstream process in the dialog system. The reject threshold is set differently for each functionality to minimize false rejects as described in Su et al. (2018) and varies from 0.13 to 0.35.

4 Dialog Paraphrase Corpus

How users interact with a spoken dialog system is very distinguishable in terms of style of the speech. Thus, it is crucial to use a corpus that contains such style of utterances. Since we do not have a hand-annotated paraphrase corpus for our dialog system, we automatically created a paraphrase corpus from user interaction with the dialog system.

4.1 Definition

In order to pair up existing utterances with NLU interpretation with their paraphrases, we first have to define what makes **paraphrase** in this work. We define utterances that invoke the same functionality from our spoken dialog system with same entities are paraphrase of each other. For example, an utterance *Play Adele in my living room* is a paraphrase of *I would like to listen to Adele in my living room*. However, such paraphrases that share the same entities in granularity would be sparse throughout the corpus. Thus, we propose a concept of **para-carrier phrase**, which groups utterances that invoke the same functionality of the dialog system but not necessarily share the entities. For example, *Play Adele in my living room* can be a para-carrier phrase of an utterance *I would like to listen to Lady Gaga in my kitchen*.

4.2 Paraphrase Pairs

For paraphrase pair extraction, we used NLU training data that was available before any of the five functionalities we consider in this work were designed or launched. Thereby, we aim to simulate scenarios where a new functionality does not have similar or related utterances in the training data of the paraphrase model. In order to avoid potential annotation errors, we first applied frequency-based de-noising to the data, removing annotated utterances whose frequency is lower than m times throughout the corpus. For annotated utterances from live traffic, we apply $m = 3$ and for synthetic utterances we apply $m = 6$. Given de-noised utterances, we pair up utterances that are para-carrier phrase of each other. Once they are paired, we masked their entities with their slot type. Our previous example will become a para-carrier phrase pair *Play Artist in HomeLocation - I would like to listen to Artist in HomeLocation* in this step. We then randomly sample entities from an internal catalog for each slot type in order to make them into a paraphrase pair that shares the same entities. In this way, we obtained around 1M paraphrase pairs. This data is used as an in-domain data for paraphrase generation system.

5 System Description

5.1 Neural Machine Translation

For training data of the paraphrase generation system, we use both general and in-domain paraphrase corpora. The in-domain paraphrase corpus, as described in Section 4, contains 1M paraphrase pairs that fit the style and genre of the dialog system. For the general domain data, we use a back-translated English paraphrase corpus (Wieting and Gimpel, 2017). Out of a 50M pair parallel corpus, we first selected 30M pairs whose score are the highest. We then randomly selected 10M parallel sentences. The general and in-domain corpora are shuffled so that each batch can be exposed to both of them. For development data, we randomly chose 3K sentences from the in-domain data. Prior to training, we apply BPE (Sennrich et al., 2015) at operation size 40K for both source and target side concatenated.

We use a transformer (Vaswani et al., 2017) for the task, using the implementation in Klein et al. (2017). Our hyper-parameters follow the *Base* configuration of the original work, with several alterations. We use 512 as the hidden layer size, and 2048 for the inner size of the feed-forward network. We added sinusoidal position encoding to each embedding. The model is trained for $200,000$ steps, with the Adam optimizer (Kingma and Ba, 2014). We set 0.998 for β_2 in Adam optimizer and $8,000$ for warm-up steps. As our source and target languages are the same, we shared the embeddings between encoder and decoder.

Funct.	Domain	#Intent	#Slot (new)	Test
Announce	Comms.	1	17 (1)	1.3K
Quotes	Info	1	12 (5)	1.4K
Playlist	Music	2	32 (0)	1.9K
Donate	General	1	7 (3)	1.3K
Chat	General	1	1 (1)	2.7K

Table 1: Five functionalities considered in this work

5.2 Natural Language Understanding

Our NLU model consists of a domain classifier (DC), an intent classifier (IC), and a named entity classifier (NER). For this experiment, we used statistical models for the three components. A DC model outputs whether a given input utterance is intended for the target domain (e.g. Book). We trained our DC with a maximum entropy (ME) classifier, using n-grams extracted from the training data as input features. An intent of the input utterance is classified in IC. Trained with a multi-class ME classifier, the IC outputs the intent for each utterance (e.g. ReadBook). The model uses n-grams as features. The NER is used to identify named entities in the utterance (e.g. "Harry Potter" for BookTitle). We used conditional random fields for NER tagging, using n-grams extracted from training data.

Each component outputs labels and corresponding confidence scores. The overall model confidence is obtained by multiplying the three confidence scores. We also applied a reranker scheme to integrate outputs from the components and provide a list of hypotheses. A detailed description of the reranker scheme as well as the NLU system can be found in Su et al. (2018).

6 Experimental Setup

We apply paraphrase generation to five functionalities of our spoken dialog system, where each functionality consists of one to two intents (e.g. PlayMusic, PlayVideo, etc.). Five functionalities come from four different domains, as shown in Table 1. By applying paraphrase generation to various functionalities across multiple domains, we show the applicability of the technique.

Table 1 shows the number of intents and slots covered by each functionality. Additionally, the number of new slots introduced by modeling this functionality is shown in parentheses. Each functionality has a designated test set, which contains 1k to 3k functionality-specific utterances of live traffic data annotated. Table 1 shows test set size for each functionality.

In this work, we evaluate the impact of generated paraphrases in terms of the NLU model performance, measured in Semantic Error Rate (SemER) (Makhoul et al., 1999). There are three types of slot errors in a hypothesis with respect to reference interpretation: substitution error (S), insertion error (I), and deletion error (D). We treat intent of NLU interpretation as one of slots using the metric, where an intent error is considered as a substitution. SemER is calculated as follows:

$$SemER = \frac{S + I + D}{S + D + C} \quad (1)$$

where C denotes the number of correct slots/intents. Numbers we report in this work are the relative performance in terms of SemER.

6.1 Bootstrap Phase

We apply the paraphrase generation technique to two phases of spoken dialog system. The first phase is bootstrap phase where the functionality is in development. Thus, we do not have any actual user utterances in the training data but only synthetically-created training data. In our experiment, we rely on FST-generated synthetic data for a new functionality. We will call this data *bootstrap data*. We use the bootstrap data as an input to the paraphrase generation system.

Note that we can only apply Para-SSL for the bootstrap phase, as SSL requires live traffic utterances.

6.2 Live Phase

The second phase we consider in this work is the live phase, where we have user utterances annotated for training. We apply our SSL approaches on the benchmarks, as described in Section 3.3.

For live phase experiments, we compare three methods against the baseline where no additional data was used. In Para-SSL, we use live annotation data as an input for paraphrase generation and validate the output using the NLU model. In SSL, we show the results of conventional SSL where we use an n-gram based filter to find functionality-related utterances and validate them using NLU model. In Combined, we use the validated utterances from SSL as an additional input for paraphrase generation model. The validated paraphrases from both SSL and Para-SSL are added for the system with SSL for each annotation increment.

Funct.	Bootstrap	Para	Valid.	Ratio
Announce	90.0K	2.0M	460.6K	22.5%
Quotes	50.0K	1.6M	538.6K	32.8%
Playlist	21.6K	928.8K	79.2K	8.5%
Donate	5.0K	202.0K	74.8K	37.0%
Chat	150.0K	2.0M	811.8K	41.4%

Table 2: Data statistics for paraphrase generation in the bootstrap phase, including the number of utterances and validation ratio.

Funct.	Para-SSL	ω Boot.
Announce	-18.99%	-7.85%
Quotes	-3.49%	+0.56%
Playlist	-5.33%	-8.72%
Donate	-5.09%	-6.16%
Chat	-17.39%	-9.49%

Table 3: Relative SemER reduction for target functionality when adding generated paraphrases in bootstrap phase, compared to the model with bootstrap data only.

7 Results

7.1 Bootstrap Phase

Table 2 shows data statistics of the generated paraphrases for the bootstrap phase. In the second column, we show how many utterances we used for the bootstrap data. Note that this data contains duplicate utterances. Para column in the table shows how many unique paraphrases are generated when inputting the bootstrap data into paraphrase model. The next two columns show the number of validated utterances and the corresponding ratio.

We then added the validated paraphrases into the NLU training. In second column of Table 3, we show how much relative improvement in SemER we can achieve by adding the validated paraphrases. Relative performance is evaluated against the baseline where no validated paraphrases were used. We can see that all functionalities' performances is improved greatly, with relative SemER reduction ranges from -3.5% up to -18.99% .

Additionally, we investigated whether we can achieve comparable performance by up-weighting the existing bootstrap training data. For this experiment, we randomly sample existing bootstrap data to the same amount as the validated paraphrases. Instead of the validated utterances, we then used the up-weighted bootstrap data (the ωBoot. column in Table 3). Especially for the functionalities where we obtained a big improvement using paraphrases (Announce, Chat), up-weighting bootstrap data did not lead to comparable result. This result shows the potential of Para-SSL in bootstrap phase to improve functionality performance without using user interactions.

7.2 Live Phase

Table 4 shows the number of validated paraphrases, for each functionality and annotation increment. As expected, we obtain more validated utterances as annotation increment increases. We can see that for most of the functionalities SSL obtains a bigger pool of validated utterances, compared to Para-SSL. It is noticeable that Combined sometimes obtains a smaller number of utterances validated, compared to SSL (e.g. Live10 for Quotes). Note that SSL and Combined rely on two different NLU models to augment and validate the utterances. For validation, Combined uses the model trained with validated paraphrases of bootstrap data. Adding generated paraphrases from bootstrap data changes decision boundary for the model, shifting confidence score ranges as well.

Table 5 shows the impact when we generate paraphrases given live annotation data and add the validated ones into training data. For each functionality, we present three systems' performance against the baseline where only annotated training data is available. We can see that using Para-SSL can effectively improve NLU performance for most of the functionalities. Note that Para-SSL benefits from its capability of utilizing bootstrap data, in live phase as well. Even when the amount of validated utterances from Para-SSL is much smaller than the ones from SSL, we often observe comparative results.

On the other hand, Para-SSL did not bring a great improvement for Playlist, possibly due to the low validation rate throughout bootstrap and live phase, compared to other functionalities. We believe that the model is less prone to provide a high confidence score for a complex functionality such as Playlist. As shown in Table 1, Playlist involves the highest number of slots and intents. Also, there is no new slot involved for the modeling of this functionality. Thus, the model has to learn existing slots in a different context, which may lead to a generally lower confidence score range for the functionality.

In Combined, we observe that Para-SSL and SSL bring complementary improvements. It is also noticeable that even when we have less amount of utterances validated, Combined outperforms SSL. Figure 1 depicts utterances from vari-

Functionality	System	Live10	Live20	Live50	Live80	Live100
Announce	Para-SSL	1.0K	1.8K	1.9K	6.0K	27.8K
	SSL	6.1K	6.5K	9.9K	9.5K	29.6K
	Combined	10.9K	21.1K	34.8K	44.6K	111.5K
Quotes	Para-SSL	0.2K	0.6K	2.0K	5.5K	9.5K
	SSL	45.4K	55.6K	68.9K	120.6K	176.7K
	Combined	15.2K	98.9K	198.7K	378.7K	298.7K
Playlist	Para-SSL	40	0.2K	0.5K	1.6K	3.2K
	SSL	32.7K	96.3K	260.3K	398.9K	725.1K
	Combined	10.0K	69.3K	173.3K	664.4K	1.4M
Donate	Para-SSL	0.2K	0.3K	0.7K	1.8K	3.0K
	SSL	0.2K	0.2K	0.4K	0.5K	1.2K
	Combined	1.1K	1.6K	2.6K	4.8K	10.0K
Chat	Para-SSL	0.5K	0.8K	1.5K	2.5K	3.8K
	SSL	30.7K	22.4K	36.1K	52.8K	103.6K
	Combined	137.1K	147.7K	243.1K	330.4K	579.2K

Table 4: Data statistics on the number of validated utterances in live phase, per annotation increment

Functionality	System	Live10	Live20	Live50	Live80	Live100
Announce	Para-SSL	-19.11%	-16.49%	-14.40%	-15.96%	-22.71%
	SSL	-20.27%	-19.12%	-16.78%	-10.74%	-17.73%
	Combined	-27.44%	-31.13%	-29.40%	-29.55%	-35.73%
Quotes	Para-SSL	-11.08%	-9.65%	-2.42%	-4.92%	-5.01%
	SSL	-16.71%	-15.51%	-12.46%	-13.50%	-18.45%
	Combined	-21.38%	-28.26%	-21.90%	-18.62%	-22.19%
Playlist	Para-SSL	-1.55%	-2.98%	-1.63%	-0.89%	-0.61%
	SSL	-18.43%	-13.50%	-18.19%	-13.20%	-15.45%
	Combined	-19.40%	-16.24%	-19.85%	-14.02%	-15.68%
Donate	Para-SSL	-4.92%	-2.24%	-5.22%	-6.85%	-13.52%
	SSL	-4.92%	-4.36%	-6.04%	-6.69%	-12.78%
	Combined	-8.03%	-11.9%	-15.82%	-19.17%	-29.92%
Chat	Para-SSL	-9.55%	-15.03%	-12.59%	-13.75%	-16.76%
	SSL	-25.14%	-16.50%	-17.90%	-20.50%	-24.39%
	Combined	-30.95%	-30.90%	-26.72%	-32.27%	-35.27%

Table 5: Relative SemER reduction for target functionality when adding generated paraphrases in live phase.

ous sources in an embedding space[1]. We can see that generated paraphrases from Combined fill the gap between data points for bootstrap, live annotation, and SSL.

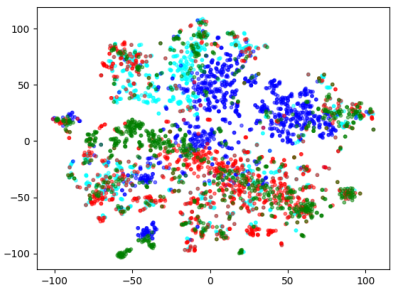

Figure 1: Embedding depiction of utterances for Announce functionality. Blue = bootstrap data, red = live annotation data, green = SSL, cyan = Combined

[1]1K utterances are randomly sampled from each source. We train embeddings using 89 million utterances in production (Pagliardini et al., 2017). For visualization we used t-SNE (Maaten and Hinton, 2008).

8 Analysis

First, we quantified the quality of generated utterances in n-best list in terms of their validation yield. Figure 2 shows the validation rate for each functionality. Validation rate is calculated for each n in the n-best list, by dividing the number of validated paraphrases by the number of generated ones, given all input utterances. Solid line represents semantic fidelity trend as n increases, showing how many of the generated utterances for each n are validated through. The dashed line shows the diversity trend in the n-best list. It represents how many of the generated utterances are unique utterances within the generated data assuming that we are adding new utterances starting from top to bottom in n-best lists for all utterances.

As expected, the general trend of the yield decreases (thus semantic fidelity likely decreases as well) as n grows. However, note that it does not drastically drop, but instead it reaches a plateau. The varying level of yield rate for different func-

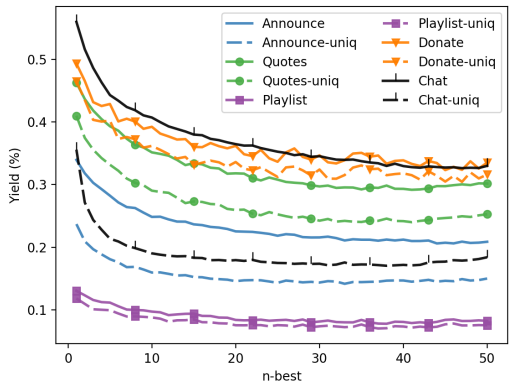

Figure 2: Validation rate for n in n-best list. Dashed line represents how many of the validated utterances are unique utterances. Paraphrases are generated by inputting bootstrap training data for each functionality.

Validated	Filtered-out
Announce that supper is ready	Declare that supper is ready
Get me a famous quote from Obama	I'd like to order a famous quote from Obama
Put on a dance pop song from the nineties to the playlist	Put on a dance pop song from the nineties

Table 6: Examples of validated and filtered-out paraphrases.

tionalities also indicates that validation is a necessary step in order to use generated paraphrases into training data. We believe a further analysis will be beneficial to understand the trade-off between computational complexity and diversity of validated utterances when increasing size of n.

Additionally, we share examples of validated and filtered-out utterances in Table 6. We can see that validation step can successfully filter out paraphrases that do not conform well to the context of dialog system. A vague paraphrase in terms of NLU functionality (e.g. add a song to playlist vs. play a song) could also be filtered out.

For the second analysis, we looked into the necessity of keeping paraphrases of bootstrap data, especially when the model is trained with more live annotation data. As shown in Algorithm 2, we kept using the validated paraphrases of bootstrap data in live phase, in order to benefit from Para-SSL's applicability in bootstrap phase. As bootstrap data is often relatively larger than the annotated live data, we would keep the big corpus throughout the cycle of functionality development, potentially increasing the computational cost. For this analysis, we remove the validated paraphrases of bootstrap data and retrained the model.

Funct.	Live80	Live100
Announce	-29.05%	-35.31%
Quotes	-15.35%	-18.96%
Playlist	-13.17%	-15.18%
Donate	-17.92%	-26.27%
Chat	-32.34%	-37.96%

Table 7: Retiring paraphrases from bootstrap data, from Combined experiments. Numbers are reported in relative SemER reduction.

The analysis is applied for Combined experiments. Table 7 shows the result for annotation increments 80 and 100. Comparison to the numbers in the same increments shown in Table 5 shows that there is no substantial degradation caused by retiring the generated paraphrases from bootstrap data, when model is trained with sufficient live annotation data.

Experiment showed that we still benefit from augmenting and validating paraphrases using the Combined system, without retiring the validated paraphrases of bootstrap data. When we use the system with data retirement, we reached a worse performance in live phase. This indicates that we can use a better-performing but potentially computationally expensive model for utterance augmentation and validation, and for production we can use a lighter system with a comparable performance.

9 Conclusion

In this work, we investigated the impact of paraphrase generation for semi-supervised learning in NLU. The proposed method has an advantage over the conventional SSL that it does not require actual user utterances. Using Para-SSL, thus, we can improve the support for a new functionality effectively prior to launching it.

We applied Para-SSL on five functionalities in an NLU system. In addition to compare the results with the conventional SSL, we also combined the two SSL methods to achieve even better performance. Experiments show that Para-SSL leads up to 19% of relative error reduction without an access to user utterances, and up to 35% when combined with SSL method, leveraging live traffic utterances.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 196–205. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 17–24. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *North American Association for Computational Linguistics*.

Prasad Kawthekar, Raunaq Rewari, and Suvrat Bhooshan. 2017. Evaluating generative models for text generation.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Jeff Ma, Spyros Matsoukas, Owen Kimball, and Richard Schwartz. 2006. Unsupervised training on large amounts of broadcast news data. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 3, pages III–III. IEEE.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

John Makhoul, Francis Kubala, Richard Schwartz, Ralph Weischedel, et al. 1999. Performance measures for information extraction.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 881–893.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics.

Avital Oliver, Augustus Odena, Colin Raffel, Ekin D Cubuk, and Ian J Goodfellow. 2018. Realistic evaluation of deep semi-supervised learning algorithms. *arXiv preprint arXiv:1804.09170*.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*.

Chris Quirk, Chris Brockett, and Bill Dolan. 2004. Monolingual machine translation for paraphrase generation.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 616–623.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Alex Sokolov and Denis Filimonov. 2019. Neural machine translation for paraphrase generation. *2nd Conversational AI*.

Chengwei Su, Rahul Gupta, Shankar Ananthakrishnan, and Spyros Matsoukas. 2018. A re-ranker scheme for integrating large scale nlu models. *arXiv preprint arXiv:1809.09605*.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational linguistics*, 29(1):97–133.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Gokhan Tur, Dilek Hakkani-Tür, and Robert E Schapire. 2005. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2018. A task in a suit and a tie: paraphrase generation with semantic augmentation. *arXiv preprint arXiv:1811.00119*.

John Wieting and Kevin Gimpel. 2017. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu Seqgan. 2016. sequence generative adversarial nets with policy gradient. arxiv preprint. *arXiv preprint arXiv:1609.05473*, 2(3):5.