Improving Natural Language Understanding Accuracy by Pre-Adapting to Live Traffic

Lisheng Fu, Konstantine Arkoudas
Amazon Alexa AI
New York, NY, USA
{lishef,arkoudk}@amazon.com

Abstract

Virtual assistants enable users to interact with a large number of services in natural language. Third-party developers building new applications for virtual assistants often have limited annotation resources and find it challenging to procure large amounts of suitable training data, opting instead for limited collections of sample utterance templates, annotated with their semantics. We can enrich such collections by synthesizing more examples based on the given templates, but the resulting utterance distribution will still be quite different from the distribution of actual user utterances in the wild. We treat this as a domain adaptation problem from developer-provided sample utterances to live utterances, and we apply adversarial training between them to mitigate their gap. In addition, we show that we can achieve this in the pre-training stage as pre-adaptation. We demonstrate consistent improvements across different test sets in two different languages.

CCS Concepts

• **Computing methodologies** → **Natural language processing**; **Neural networks**.

Keywords

domain adaptation, adversarial training, spoken language understanding

ACM Reference Format:
Lisheng Fu, Konstantine Arkoudas. 2021. Improving Natural Language Understanding Accuracy by Pre-Adapting to Live Traffic. In *Proceedings of KDD '21: Data-Efficient Machine Learning workshop (DeMaL '21)*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Virtual assistants such as Amazon's Alexa and Google Assistant have become increasingly popular in recent years. These assistants accept spoken utterances from users to complete tasks such as setting an alarm, looking up the weather, playing music, and so on. They use speech recognition to convert voice to text and then use natural language understanding (NLU) to identify the intents and slots in the utterances (Table 1). NLU is typically formulated as the joint task of intent classification (IC) and named entity recognition (NER), also known as slot labeling. With sufficient volumes of annotated training data, such NLU problems can be solved fairly well.

Utterance	play	taylor	swift
Intent	PlayMusicIntent		
Slot	Other	ArtistName	ArtistName

Table 1: Example of intent classification and slot labeling for utterances.

Most modern virtual assistants are extensible, allowing third-party developers to build new voice-powered applications for new domains. In Alexa, external developers build new applications (so-called *skills*) with the *Alexa Skill Kit*, or ASK for short, which allows them to define an interactive model including an intent schema, slot types, and sample annotated utterances [10] that later serve as the training data for a statistical NLU model (see listing 1).

Listing 1: Skill definition which contains sample utterances.

```
{"skill_name": "play music",
 "sample_utterances": [
  {"id": 0, "intent": "PlayMusicIntent", "text": "play {SongName}"
       },
  {"id": 1, "intent": "PlayMusicIntent", "text": "listen to {
       SongName}"},
  {"id": 1, "intent": "PlayMusicIntent", "text": "can you play {
       SongName}"},
  {"id": 3, "intent": "StopMusicIntent", "text": "stop the music"}
       ,
  {"id": 4, "intent": "ResumeMusicIntent", "text": "continue the
       song"}
  ],
  "slots": [{"name": SongName, "values": ["BLINDING LIGHTS", "
       DANCE MONKEY", "ROSES"]}]
 }
```

Depending on the resources available to the developers, these sample utterances can be quite limited in their scope. Consequently, users in real life end up speaking in markedly different ways that were not anticipated by the developer-provided samples, resulting in poor NLU performance. There are two serious challenges for NLU here: the low volume of sample utterances serving as training data, and their distribution drift from actual user utterances. An additional practical constraint is that these challenges must be addressed in a cost-efficient and scalable way.

We tackle these challenges by introducing live traffic data into model building and employing adversarial training [5] between sample utterances and live utterances to implement the desired adaptation. We use small models (BiLSTMs) to be able to train and run inference on low-cost machines in production. Moreover, we try to adapt the models to live traffic in pre-training as pre-adaptation (Figure 1). In this way, we reduce training cost and

DeMaL '21, Aug 14–18, 2021, Virtual, CA, United States
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

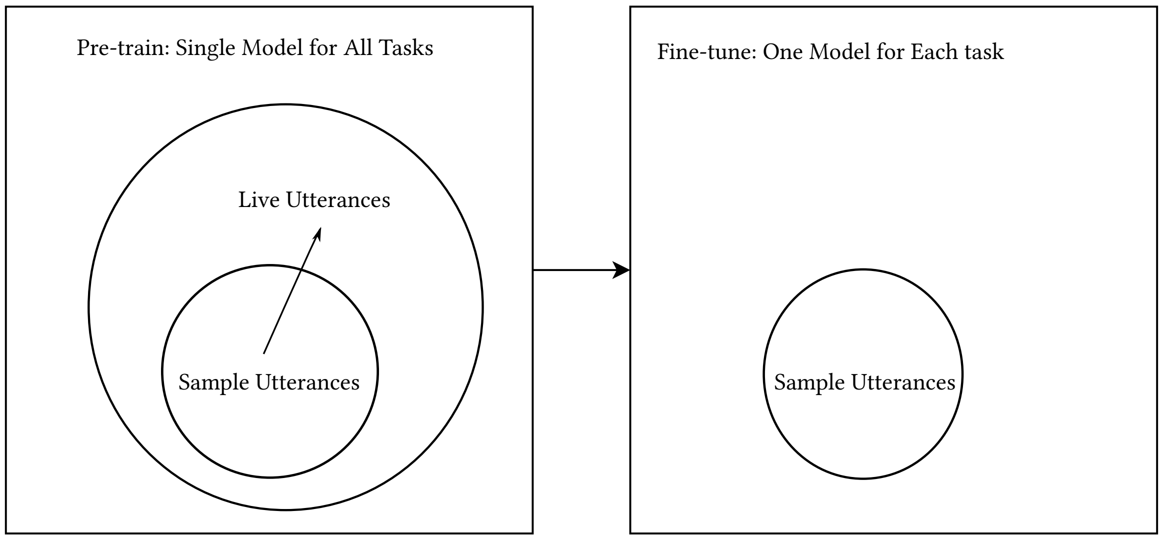
Pre-train: Single Model for All Tasks
Live Utterances
Sample Utterances
Fine-tune: One Model for Each task
Sample Utterances

Figure 1: Pre-adaptation to Live Traffic

latency while maintaining accuracy improvements. In particular, our contributions are as follows:

- We show that adversarial training can leverage live traffic to improve NLU (IC and NER) accuracy.
- We demonstrate that this kind of domain adaptation can be achieved in pre-training as pre-adaptation.
- We demonstrate consistent improvements across different test sets in two languages.
- Our solution is cost-efficient and can be deployed in production with no additional infrastructure.

The rest of the paper is organized as follows. In Section 2 we review related technologies. In Section 3 we introduce our model components, live traffic selection, and how to train the model as pre-adaptation. We present our experimental results in Section 4 and conclude in Section 5.

2 Related Work

Spoken Language Understanding (SLU): This is an important component of conversational AI systems, which typically includes domain detection, intent classification, and slot labeling. Currently, state-of-the-art SLU systems are mostly based on deep learning models [12, 13, 17]. Although these models show good performance in a variety of datasets, they usually require large amounts of training data and make strong i.i.d. (independent and identically distributed) assumptions. It is still a challenging research problem to make these models work with small training datasets, particularly when these are based on a distribution that is different from that of the test data. This is the problem we tackle in this paper.

Adversarial Training: There have been different adversarial training strategies for augmenting training data, such as Generative Adversarial Training [6] and Virtual Adversarial Training [14]. Both add a noise function to the input data as regularization. This can make the model more robust to test data coming from a different distribution. In our setting we have a similar problem but the distribution gap is often larger than a small permutation. We also have access to the distribution difference from the live traffic. Thus, we can use domain adversarial training [5] to take advantage of this prior knowledge. This technique has also been successful for adaptation in a variety of NLP tasks such as natural language inference [1, 8], natural langauge grounding [16], and information extraction [4]. Here, we apply it on SLU with joint IC and NER training. More distinctly, we conduct the adversarial training in the pre-training stage, and show that the adaptation still helps after fine-tuning. In terms of data (domain), we adapt the models to the user-generated live traffic from sample utterances. The distribution change is different from the academic datasets which often have domains as different genres or contain manually designed difference.

Pre-training: This has been a popular technique for models with insufficient training data, which can also be helpful in dealing with domain shifts. Large pre-trained Transformer-based language models and their variations [2, 3, 11] have delivered very strong results on many different NLP tasks. They have shown fairly good adaptation ability on language expressions that do not appear in the training set. However, a downside of these models is that they are extremely large, which inhibits training and inference in production. The cost of using these models is also much higher, and this may not always be affordable. Our approach is based on lightweight BiLSTM models [7], so it is more cost-efficient. Our results demonstrate good adaptation even with these small models.

3 Method

3.1 Intent Classification and Slot Labeling Baseline

The systems are built based on skill definitions (Listing 1). A popular hybrid approach to NLU is to use a symbolic pattern matching algorithm based on the sample utterances in the skill definition,

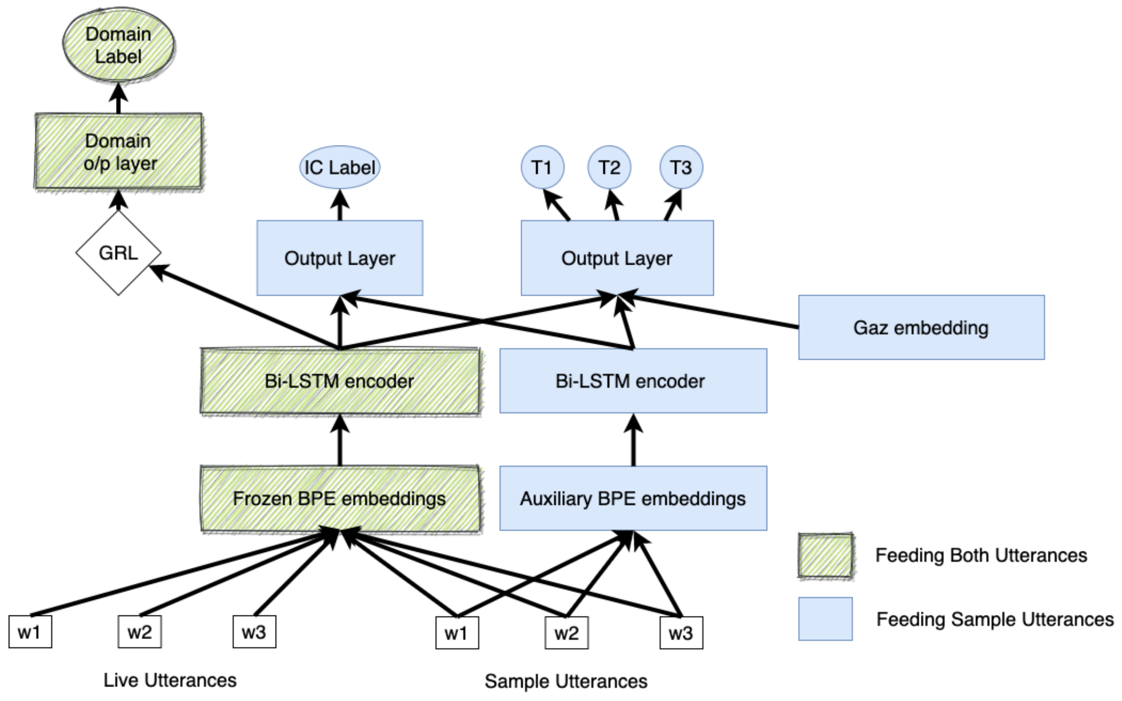

Figure 2: Model architecture with the adversarial training.

and a statistical model for utterances that cannot be handled by such pattern matching. In a cost-efficient setting, our base statistical model is a small BiLSTM with joint intent classification and slot decoding. Both training and inference can be done on low-cost machines. The intent classification (IC) component is a multi-class classification task and the slot labeling (SL) is a sequence labeling task. For the input layer, we convert utterance tokens to subword Byte Pair Encoding (BPE) embeddings [15]. The main encoder is a BiLSTM on top of the frozen BPE embeddings. The BPE embeddings are frozen to be shared between a set of models to reduce the cost of storage and computation. Then, we have an lightweight auxiliary BiLSTM encoder with a smaller vocab size to compensate the limitation of the frozen embeddings for each model. As the results, the trainable parameters that require storage for the models are more than 10 times smaller than the frozen embeddings. Finally, we have auxiliary gazetteer features encoded in the gazetteer embeddings. Gazetteer features are indicator features of whether the token stays in a span that matches an entry of the dictionary. Then, the indicator features are converted to vectors for training. In the output layers, we have a softmax for IC and a conditional random field (CRF) layer for SL. Both have one hidden fully connected layer before the last layer. The design rationale behind frozen BPE embeddings is to be able to serve a large number of models at the same time. Each additional model will only incur a fairly small cost in space and time. The model is trained with the sum of the cross-entropy losses of IC and SL. The formulation is the following:

$$x_{main} = BiLSTM(W_{frozen}), \tag{1}$$

$$x_{aux} = BiLSTM(W_{aux}), \tag{2}$$

$$x_{ic} = max\_pool([x_{main}; x_{aux}]), \tag{3}$$

$$x_{sl} = [x_{main}; x_{aux}; W_{gaz}], \tag{4}$$

$$y_{ic} = softmax(dense(x_{ic})), \tag{5}$$

$$y_{sl} = CRF(positionwise\_dense(x_{sl})), \tag{6}$$

$$L_{ic\_sl} = L_{ic} + L_{sl}, \tag{7}$$

where W_{frozen} are the frozen BPE embeddings shared by all skills, W_{aux} are the auxiliary BPE embeddings for each skill with a smaller vocab size, and the gazeetter embeddings W_{gaz} are only used for SL. The model architecture is shown in Figure 2. This includes the adversarial classifier, which we describe in the next section.

3.2 Adaptation by Adversarial Training

To adapt to live traffic, we take unlabeled user utterances as input and use the source of these utterances (live traffic) as prior knowledge in order to perform adversarial training between sample and live utterances. This can be viewed as domain adversarial training [5], where $domain = \{live, sample\}$. The adversarial training is independent of the main underlying tasks, as it does supervised training on its own labels. Accordingly, it can be applied to other problems as well. In our setting, we add a new domain classifier. The labels are the source of the utterances (i.e., live or sample). This is a binary classifier which determines whether an utterance comes from live traffic or is a sample utterance coming from the domain definition. The input consists of the representations that are used as signals into the main tasks (i.e., IC and SL). These representations would learn the domain difference when optimizing the loss of the domain classifier. We then apply a Gradient Reversal Layer (GRL) [5] . The representations will learn in a reverse direction, which achieves domain invariance. This means the input representations will tend to become the same for live utterances and sample utterances.

More formally, the input representation for the domain classifier will go though a GRL first:

$$GRL(x) = x, \tag{8}$$

$$\frac{d_{GRL(x)}}{d_x} = -I, \tag{9}$$

where x is the input representations and I is the identity matrix. The parameters of the domain classifier and the input representations are optimized in reverse directions, which act as a min-max game in the adversarial training. This classifier is trained with the main tasks (IC and SL) in a multi-task setting. The loss is also a cross entropy loss:

$$L = L_{ic\_sl} + \lambda L_{adv}, \tag{10}$$

where λ controls the min-max game. When λ is small, the input representations are still mainly learned from the IC and SL tasks, and thus distinctive for source (sample or live). When λ is sufficiently large the input representations are more domain-invariant, which should therefore generalize better on the target domain (live traffic). If λ is too large, the adversarial loss will dominate the main tasks, leading to poor performance in both domains. We conduct grid search for the hyper-parameter tuning of λ as explained in Section 4.2 and Section 4.3.

Since we are doing joint IC classification and SL decoding, we use the concatenation of the two representations to take account of both tasks:

$$x = [x_1; x_2], \tag{11}$$

$$x1 = max\_pool(BiLSTM(W_{BPE})), \tag{12}$$

$$x2 = last\_pool(BiLSTM(W_{BPE})), \tag{13}$$

where x_1 is also used to predict intent as x_{ic} and x_2 is the sequence representation as last pooling of tokens x_{sl} that predicts slots.

3.3 Live Traffic Selection

Unlike sample utterances provided by developers, live traffic may contain several orders of magnitude more utterances. Not only will the training cost be much higher if we take all live traffic, it will also be hard to maintain balance in the multi-task training. We thus experiment with different criteria for selecting utterances from live traffic:

- **Top/Random N** This extracts a subset of size N by sampling. By making N close to the number of sample utterances, we reduce training cost while simplifying multi-task training. We consider the frequency of unique utterances in live traffic, in order to either select the top N or else to randomly sample according to utterance frequency.
- **ASR Confidence** This considers automatic speech recognition (ASR) errors in live traffic, which may result in training noise. We experimented with this threshold to remove poor quality utterances.
- **Pattern Match** Our system is a mixture of pattern matching on the basis of the sample utterances and the statistical model. User utterances that are already recognized by pattern matching (i.e., which match sample utterances) do not add value, so this criterion aims to filter out such utterances and focus instead on utterances that only exist in live traffic.
- **Minimum Occurrence** We observe that there are a lot of long-tail utterances which occur very rarely in live traffic. We hypothesize that these utterances have a higher chance of being out-of-domain utterances or created due to various errors. Valid utterances with proper semantics are more likely to be uttered multiple times by one or more users.

3.4 Pre-adaptation

Although selecting a small subset of live traffic can significantly reduce training costs, these are still much higher than they would be without any live traffic. Introducing live traffic into model building may also introduce privacy concerns, as it directly incorporates customer data, which could potentially be exploited by malicious developers. Thus, putting this adaptation process in the pre-training stage may be preferable. It is not clear how to achieve this, especially when we have a large number of models instead of just one model. We experiment with a simple variation that turns out to work fairly well in practice.

We manually create a new task that combines all intents and slots together. The label space is the combination of all tasks. This is like doing multi-task training within a single task. We combine the live traffic as well for the domain classifier. Then the adversarial training is performed on this single model as pre-training. The adaptation for the actual models in production is achieved by fine-tuning the encoder obtained from this combined dataset. As the pre-trained encoder is the same size as our original model, the training cost and latency in production do not change at all. Our experiments show that it can still perform adaptation fairly well, even though it is done only in pre-training on a combined dataset.

4 Experiments

4.1 Settings

We evaluate our methods on our internal datasets, which consist of dev and test data for a fixed set of skills. To verify the generality of our models, we tested them in two different language/locales. We have four test sets in the following experiments (en_US, en_US held-out, de_DE, de_DE held-out), where en_US and de_DE were used for dev (tuning) purposes. All four sets were sampled from live traffic. The number of skills in each test set is around 80. The model sizes are the same for all skills.

We use Semantic Error Rate (SemER) as our evaluation metric for IC and SL. SemER is a metric that combines both tasks by treating the intent as one of the slots. It allows for partial credit, as it computes a sort of edit distance between a prediction and its reference. More formally,

$$SemER = ((S + I + D)/T) * 100$$
$$S = number\ of\ substitution\ errors$$
$$I = number\ of\ insertion\ errors$$
$$D = number\ of\ deletion\ errors$$
$$T = number\ of\ total\ slots$$

Method	SemER Reduction (%)
Baseline	0.0
+ Adversarial Training	1.85
+ Skill-specific Weight	3.80

Table 2: Adversarial training with live traffic.

Method	en_US	held-out	de_DE	held-out
Baseline	0.0	0.0	0.0	0.0
+ Pre-adaptation	2.39	1.88	2.28	2.10

Table 3: Relative SemER Reduction (%) with pre-adaptation.

where the errors and T are computed based on the reference semantic representations. Relative SemER reductions are reported in the tables.

In order to reduce training costs in production, we use small dimensions for our models. Our main BiLSTM encoder has 64 units and the auxiliary encoder has 16 units. The embedding sizes for the frozen BPE, auxiliary BPE, and Gazetteers are 256, 16, and 8, respectively. The vocab size is around 55K for the frozen BPE and less than 10K for the auxiliary BPEs. We use the Adam Optimizer [9] with a learning rate of 0.002. In order to obtain more stable results, every score is computed as the average of 5 runs with different random seeds.

4.2 Adaptation in Production

In this section we present results when we have access to live traffic during model building. This means we can have domain adversarial training along with the IC and SL tasks in production. We first experiment with different criteria for filtering live traffic, as discussed in Section 3.3. We obtain the best results with the top 20K unique utterances, each having at least two occurrences. Random sampling does not work very well in our setting, most likely because of too many long-tail unique utterances. Thus, we use the most frequent utterances, which appear to better characterize live traffic. 20K utterances for each skill result in a good trade-off between training cost and accuracy. Accuracy does not improve much farther if we add more live data. ASR confidence levels and filtering out pattern-matched utterances do not make much difference in our results. Even though we can observe some poor quality utterances, our models seem to be robust in tolerating that level of noise. Live utterances have been de-identified for experiments. With these live-traffic settings, we experiment on the en_US dataset, which contains 88 skills. We tune the weight of the adversarial loss over the list $\lambda = [0.01, 0.03, 0.05, 0.08]$, obtaining the best results with 0.08. We obtain 1.85% relative SemER reduction in this setting. If we allow using different weights for different skills, the improvement goes up to 3.8% within these four weights (Table 2). Thus, if we do enable live traffic data in production, allowing the skill-specific weight for the adversarial loss will maximize the gains obtained from live traffic data.

4.3 Pre-adaptation

This section presents results when we apply adversarial training in pre-training (Section 3.4), which reduces production costs for training and inference and may better safeguard privacy. We apply pre-training on the main BiLSTM (64-unit) encoder and fine-tune during training on the sample utterances. We then evaluate the models on the test sets. Notice that there could be different ways to pre-train an BiLSTM encoder. We have two more pre-training baselines here:

- **Masked Language Model:** This is to perform language model pre-training on a more recent BERT-style MLM (masked language model) task [3], aimed at pretraining the BiLSTM encoder.
- **Target Task Pre-training:** This is to pretrain on similar tasks (IC and SL) when much more annotated data is available.

However, neither of these two baselines yields an improvement on these test sets. This might be counterintuitive. Although the language model has access to live traffic data, it may not capture language expressions that are more specific to individual skills and, more importantly, their labels (which are not seen during MLM training). Our small encoder also makes this harder. For IC and SL pre-training, the pre-training tasks contain very different sets of intents and slots. When we apply the technique to a set of diverse skills, as in our test sets, it is common for it to improve some skills while hurting others. We need more precise pre-training in order to make the technique viable.

Our adversarial pre-training combines the targeted sample utterances and live utterances and focuses on bridging the gap between training and test data. We obtain consistent improvements on all test sets, including two held-out test sets in two different languages (Table 3). The weight of the adversarial loss is tuned at $\lambda = (0.1, 1.0, step = 0.1)$. The best λ is 0.3 for en_US and 0.5 for de_DE. On the held-out test set, the pre-trained encoder is directly applied without tuning. The generalization on a held-out test set depends on the distribution shift between live and test data, changes in skill definitions, and changes in the underlying skill set. This is easier if we just split the test set rather than divide according to date ranges. This might be the main reason that we see slightly better consistency in de_DE than in en_US, as we use the former splitting scheme for de_DE and the latter for en_US. In de_DE we just split the test set because we do not have enough annotations on different date ranges for the same set of skills.

5 Conclusion

We have presented a simple and cost-efficient method for adapting synthetic sample utterances to live traffic. This technique is largely unsupervised, which makes it easier to deploy. If live traffic data is enabled, we have shown that using skill-specific weights for the adversarial loss will maximize the accuracy gains to be derived from that data (3.8% relative improvement). Moreover, we achieve adaptation in pre-training with 1.88%-2.39% relative improvement in two different languages. This is an interesting result, as it has not been clear that adaptation would be at all possible in pre-training stages.

References

[1] Yonatan Belinkov, Adam Poliak, Stuart Shieber, Benjamin Van Durme, and Alexander Rush. 2019. Don't Take the Premise for Granted: Mitigating Artifacts in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 877–891. https://doi.org/10.18653/v1/P19-1084

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[3] 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, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[4] Lisheng Fu, Thien Huu Nguyen, Bonan Min, and Ralph Grishman. 2017. Domain Adaptation for Relation Extraction with Domain Adversarial Neural Network. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, 425–429. https://www.aclweb.org/anthology/I17-2072

[5] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*. PMLR, 1180–1189.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[7] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[8] Anush Kamath, Sparsh Gupta, and Vitor Carvalho. 2019. Reversing Gradients in Adversarial Domain Adaptation for Question Deduplication and Textual Entailment Tasks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 5545–5550. https://doi.org/10.18653/v1/P19-1556

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Bjorn Hoffmeister, Markus Dreyer, Stanislav Peshterliev, Ankur Gandhe, Denis Filiminov, Ariya Rastrow, et al. 2017. Just ASK: building an architecture for extensible self-service spoken language understanding. *arXiv preprint arXiv:1711.00549* (2017).

[11] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

[12] Ryo Masumura, Yusuke Ijima, Taichi Asami, Hirokazu Masataki, and Ryuichiro Higashinaka. 2018. Neural confnet classification: Fully neural network based spoken utterance classification using word confusion networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6039–6043.

[13] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig. 2015. Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23, 3 (2015), 530–539. https://doi.org/10.1109/TASLP.2014.2383614

[14] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 1979–1993.

[15] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1715–1725. https://doi.org/10.18653/v1/P16-1162

[16] Xin Eric Wang, Vihan Jain, Eugene Ie, William Yang Wang, Zornitsa Kozareva, and Sujith Ravi. 2020. Environment-agnostic multitask learning for natural language grounded navigation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*. Springer, 413–430.

[17] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*. 189–194. https://doi.org/10.1109/SLT.2014.7078572