Combining Weakly Supervised ML Techniques for Low-Resource NLU

Victor Soto and **Konstantine Arkoudas**
Amazon Alexa AI
New York, NY, USA
{nvmartin, arkoudk}@amazon.com

Abstract

Recent advances in transfer learning have improved the performance of virtual assistants considerably. Nevertheless, creating sophisticated voice-enabled applications for new domains remains a challenge, and meager training data is often a key bottleneck. Accordingly, unsupervised learning and SSL (semi-supervised learning) techniques continue to be of vital importance. While a number of such methods have been explored previously in isolation, in this paper we investigate the synergistic use of a number of weakly supervised techniques with a view to improving NLU (Natural Language Understanding) accuracy in low-resource settings. We explore three different approaches incorporating anonymized, unlabeled and automatically transcribed user utterances into the training process, two focused on data augmentation via SSL and another one focused on unsupervised and transfer learning. We show promising results, obtaining gains that range from 4.73% to 7.65% relative improvements on semantic error rate for each individual approach. Moreover, the combination of all three methods together yields a relative improvement of 11.77% over our current baseline model. Our methods are applicable to any new domain with minimal training data, and can be deployed over time into a cycle of continual learning.

1 Introduction

Virtual assistants are becoming ubiquitous, their expansion fueled by third-party developers who build voice-enabled applications for an increasingly diverse array of domains. However, oftentimes these independent developers don't have the wherewithal to produce sufficient amounts of labeled data, and consequently their applications suffer from poor NLU performance. Large pre-trained language models have mitigated but not eliminated the need for domain-specific training data, and therefore unsupervised and SSL techniques continue to form an important direction of work in the field. While a number of such methods have been previously tried in isolation, in this paper we explore the synergistic use of a number of SSL techniques with a view to improving NLU accuracy in low resource settings, specifically for third-party (3P) applications. NLU in our setting is understood as the joint task of Intent Classification (IC) and Named Entity Recognition (NER).

Alexa is an AI virtual assistant developed by Amazon. By default, Alexa is enabled to work across many Amazon-built domains, including news, music, calendar, weather, etc. But Alexa also includes a third-party *skill toolkit*, which enables external developers to implement new voice-enabled functionality in the form of external skills, such as quiz and trivia games, food-ordering skills, voice interfaces to a host of devices ranging from vacuum cleaners to automobiles, and so on. Developers can build skills by creating skill definitions, consisting of carrier phrases annotated with intent labels and slot labels (a.k.a named entities). For example, for a pizza ordering skill, a developer could provide the following carrier phrase: *I would like a* **Size** *pizza with* **Topping** *and* **Topping**, with an intent such as **OrderPizza**. The values of the **Size** and **Topping** slots are specified in the form of catalogs:

$$\begin{aligned}\text{Catalog}(\textbf{Size}) &= \{\textit{large}, \textit{medium}, \ldots\}\\ \text{Catalog}(\textbf{Topping}) &= \{\textit{bacon}, \textit{peppers}, \ldots\}\end{aligned}$$

A full utterance can be realized from the carrier phrase and the slot catalogs, e.g., *I would like a medium pizza with peppers and bacon*. Every token in this utterance would be labeled with an **Other** slot, except for the tokens corresponding to the slots in the carrier phrase, which would be labeled as **Size**, **Topping** and **Topping** respectively.

In most cases, skill definitions only contain a few carrier phrases per intent, resulting in very small training datasets that lack linguistic diversity and do not always resemble user queries. The main goal of this paper is to improve NLU model performance by incorporating unlabeled, anonymized and automatically transcribed user utterances from skills into the training pipeline, without human intervention or annotators. We explored three different methods: 1) Data augmentation via maximal FST-matching; 2) Data augmentation via tri-training ensembles; and 3) Injecting auto-encoder sentence embeddings into our baseline DNN (Deep Neural Network) model architecture.

The first two techniques automatically obtain labels for live user utterances, which are then used to augment the baseline training data. The third technique performs unsupervised pre-training of an auto-encoder (AE) on user traffic, and this AE is then used to inject sentence embeddings into our models as additional signals. The NLU task we focus on in this paper is the joint task of Intent Classification (IC) and Named Entity Recognition (NER), also referred to as Slot Labeling.

The rest of this paper is organized as follows: Section 2 gives an overview of data augmentation for NLU skills; Section 3 introduces the three methods; Section 4 details our experimental design; Section 5 summarizes the results for all skills, including cumulative results for the three methods. Finally, Section 6 presents our conclusions.

2 Related Work

Learning strategies focused on addressing the scarcity of labeled data within a specific domain can be grouped into two approaches: one focused on leveraging models and resources from other domains for which there is a wealth of resources; and approaches that aim to leverage unannotated data for the target domain.

In the first group, **Transfer Learning** strategies (McCann et al., 2017; Peters et al., 2018) focus on pre-training unsupervised models on large amounts of unlabeled data and then fine-tuning that model on a small quantity of labeled data. The most recent successful example of transfer learning are BERT models (Devlin et al., 2019), where a large transformer language model is pre-trained on either the task of masked language modeling or next sentence prediction, and whose encoder can be later fine-tuned as a feature extractor on other NLU tasks (Peshterliev et al., 2019).

In the second group, **Active Learning** algorithms use individual classifiers or ensembles of classifiers to select data points for human annotation based on different criteria: Least-confidence for examples that are assigned low confidence scores by the classifiers (Lewis and Catlett, 1994), query-by-committee for examples that are assigned a diverse set of labels by the individual classifiers in the ensemble (Freund et al., 1997), or, more recently, a Majority-CRF that relies on majority voting from an ensemble of binary classifiers (Peshterliev et al., 2019) to select data for new NLU domains. Similarly, **Semi-Supervised Learning** (SSL) (Chapelle et al., 2009) algorithms aim to use models trained on small amounts of annotated data to assign soft labels to unseen examples which can later be incorporated into the training set. Self-training (Scudder, 1965; Yarowsky, 1995; Lee, 2013) does this by using the same classifier or a teacher-student pair of classifiers to select and annotate data, whereas tri-training (Zhou and Li, 2005) creates an ensemble of three diverse classifiers that augments unlabeled datasets during an iterative process based on classifier agreement.

In this paper, we use (a) transfer learning to pre-train an auto-encoder that is later used to inject sentence embeddings into our IC-NER models (Section 3.2) and (b) SSL to augment data with IC-NER annotations obtained by maximal FST-matching and tri-training ensembles (Sections 3.1 and 3.3 respectively). On the topic of data augmentation via partial parses, similar to our maximal FST-matching approach, (Kim et al., 2015) proposes to extract slot tagging annotations from web logs using a weakly supervised approach based on Conditional Random Fields; and in (Augenstein et al., 2016) the authors use regular expressions to augment training data for the task of Stance Detection. More recently, (Karamanolakis et al., 2021) has proposed a semi-supervised framework to leverage unlabelled data and weak classification rules for improved text classification.

3 Methods

3.1 Maximal FST-Matching

The carrier phrases given by a developer are arranged into a finite-state transducer (FST), which can take an arbitrary utterance u and either (a) accept it, if u is an instance of a carrier phrase, while emitting the corresponding slot labeling and intent as its output; or (b) reject it, if u is not an exact match for any of the given carrier phrases. This FST is then sampled to generate the training data for the DNN. Depending on the number and linguistic diversity of the carrier phrases, the resulting training set can range from complete enough for a good NLU model (given smart use of transfer learning and model adaptation techniques) to severely underspecified.

With the SSL method described in this section, which we call *maximal FST matching*, we aim to augment the training set with automatically transcribed user utterances that are close in form and meaning to the ones provided by the skill developers, and which inject greater language diversity into the training. Specifically, while only a minority of user utterances are perfect FST matches (i.e., completely match developer-provided carrier phrases), many more of them are *partial* or imperfect matches. The intuition here is to compute the maximal part of a user utterance that matches a carrier phrase and then superimpose the semantics (intent and NER labeling) of that matched carrier phrase to the entire utterance. To take a simple example, the utterance *Hi, I would like a large pizza with peppers and mushrooms please* does not match the earlier carrier phrase *I would like a* **Size** *pizza with* **Topping** *and* **Topping**, due to the initial *Hi* and the trailing *please*. However, it is a partial match, since there is an internal segment of the utterance that is a perfect instance of the carrier phrase. Thus, the entire utterance can receive the intent and NER labeling of the internal segment (with tokens outside of the matching segment receiving the label **Other**) and become part of the training data.

We use a greedy approach to extract maximal internal FST matches from a user utterance: we first run the full span of the utterance through the FST, then every sub-utterance of length $n-1$, and so on down to length 1. The FST matching is stopped as soon as a match is found, at which point the intent and slot labels output from the FST are transferred to the full utterance as described above.

For each extracted match, we compute its span ratio, which is the fraction of the length of the FST-matched sub-utterance over the length of the full utterance. We treat this ratio as a tunable hyperparameter. In Section 5.1 we perform data augmentation by only keeping maximally FST-matched utterances whose span ratio exceeds a certain threshold, both globally and on a skill-specific basis.

Maximal FST-matching can introduce mislabeled utterances in several ways: a) it can ignore important semantic information if this is outside of the scope of the FST match (e.g. "(Do not) include pepperoni in the pizza."), which can potentially change the intent of the utterance (in our example, from ExcludeTopping to AddTopping); and b) it can miss slot entities that fall out of the scope of the match (e.g. "Add pepperoni (and green peppers)"). Both risks can be mitigated by filtering out utterances with low match span ratios.

This method has important advantages. It is easy to productize, since changes made to the NLU skill and its carrier phrases can be accommodated by rerunning the new FST on the already matched FST utterances, which is relatively inexpensive; and the resulting augmented training set will better reflect the true distribution of live user traffic.

3.2 Sentence Embeddings via Seq2Seq Auto-Encoder

Our baseline NLU model consists of a shared component that is pre-trained on the MLM task on Alexa traffic and other large NLP corpora, and a specific component that jointly models IC and NER, trained from scratch for every skill separately. We add to this architecture a component that is pre-trained on large amounts of automatically-transcribed traffic and optionally fine-tuned.

In particular, we experiment with seq-2-seq auto-encoder (s2s-AE) architectures. We pre-train several s2s-AE models with different features and data sources to obtain an encoder, and we plug the encoder's output into the baseline NLU model as a new component. The impact of this new feature on NLU performance (measured on a test set of 86 skills) is detailed in Section 5.2.

3.3 Tri-training

In an effort to obtain more realistic training data, we use SSL to compute high-quality labels for automatically-transcribed live user traffic. Specifically, we build a series of tri-training ensembles that we use to annotate live utterances. We show that by adding this newly annotated data to our regular training data, our baseline NLU models attain very significant improvements on SemER (Semantic Error Rate).

Tri-training (Zhou and Li, 2005) is a SSL technique that relies on three independently trained classifiers. It fine-tunes the three models by pulling examples from a pool of unlabeled data. On each iteration of the algorithm, if a pair of classifiers agree on an unlabeled example, that example is

Algorithm 1 Generalized Tri-Training

```
1:  L = grammar utterances
2:  U = live utterances
3:  Train M1, M2 on L
4:  **while** not criterion **do**
5:      U12 = Utts in U that M1 & M2 agree on
6:      M3 = Train(L + U12)
7:      U13 = Utts in U that M1 & M3 agree on.
8:      M2 = Train(L + U13)
9:      U23 = Utts in U that M2 & M3 agree on.
10:     M1 = Train(L + U23)
11: **end while**
12: Obtain labelled examples from U for examples
    that M1, M2 and M3 agree on.
```

then labeled and added to the training set of the third classifier. Algorithm 1 shows an implementation of the Generalized Tri-Training algorithm (Søgaard and Rishøj, 2010), that we also expanded to an arbitrary number of a classifiers.

Section 5.3 discusses our tri-training experiments.

4 Experiment Design

We compare our methods against our production 3P DNN model, which is depicted in Figure 1. It consists of a pre-trained shared component and a skill-specific component. The shared component is pre-trained on Alexa unsupervised traffic and external NLP corpora and computes BPE embeddings. The specific component consists of a large Bi-LSTM encoder whose input is the shared embeddings and a small BPE embedding followed by a short Bi-LSTM encoder. The IC layer output takes the summary vectors from both encoders and outputs an IC prediction. The NER layer takes as input a sequence of vectors, where each vector is composed of the concatenation of the two encoder representations and the gazetteer feature of each token. Gazetteer features are mappings from sequences of strings to Named Entities, e.g., the sequence *The Beatles* would be mapped to *Artist_Name*. This model is referred to as Base in what follows.

This study was carried out on 86 English skills from the top 100 Alexa skills with the highest amount of user traffic. Baseline training datasets for 3P skills consist of ten thousand carrier phrases sampled with repetition. For the 26 skills that include an out-of-domain (OOD) intent in their skill definition, we sample the training sets to contain

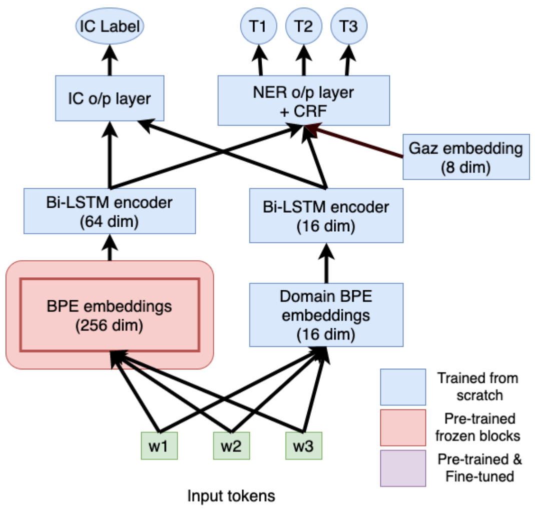

Figure 1: The Baseline NLU model.

50% of OOD intent examples. For the 86 skills included in this study, the average number of unique carrier phrases in a baseline dataset for a skill is 3,302. Our test sets for every skill are comprised of live user traffic annotated in-house with Intent and Slot labels. On average, the number of utterances in a skill test set is 816, with 425 of those being unique.

5 Results

We use two different metrics, SemER (Semantic Error Rate) and IRER (Interpretation Recognition Error Rate) (Su et al., 2018). SemER combines intent and slot classification accuracy into a single score. It computes a modified edit distance that takes into account the number of substitutions (S), incorrect predictions (I), and deletions (D) in the sequence of slots, and the intent prediction. For a sequence of L tokens, SemER is defined as (S + I + D) / (L + 1). The IRER of a single utterance is 1 if all the slots and intent are correctly recognized, and 0 otherwise. The IRER of a dataset is the fraction of utterances whose IRER is 1.

5.1 Data Augmentation via Maximal FST-Matching

We collected a dataset of live automatically-transcribed user traffic spanning four months across 86 skills, and split it in two: we use one set for selecting a Maximal Span Ratio (MSR) threshold, and the second one for testing the MSR threshold selection (and vice versa). All live user traffic used in this study was de-identified. When using a global threshold across all 86 skills, the best global threshold is 0.8. Under this scheme, a newly labeled live utterance is retained only if its maximal FST match spans at least 80% of the utterance. Using this policy, the relative improvement on SEMER with respect to our baseline DNN model (i.e., the SEMER improvement that we obtain by adding those utterances to our training data) is 1.14%.

When performing skill-specific thresholding (by using part of the skill's test data as a dev set for selecting a threshold, and the rest of the test set to compute metrics), we obtained a 10.14% relative improvement on SEMER and 6.2% improvement on IRER with respect to our baseline DNN. Another set of experiments where MSR threshold selection was performed on two months of data and tested on five months of data obtained relative improvements on SEMER and IRER of 5.01% and 1.44% respectively.

5.2 Sentence Embedding Injection

We collected automatically transcribed live traffic from the last six months of 2019. After deduping the utterances, we split them into an unsupervised training set of 60K hours of speech, and validation and test sets of 1.6K hours each.

We trained and evaluated s2s-AE models with different architectural features, as in LSTM and Transformer layers for their encoder or decoder, encoder and decoder depth (2 or 3 hidden layers), number of hidden units per layer (256, 512, or 1,024 units per layer), number of epochs for training, vocabulary token types (words or BPE), vocabulary size (20k or 50K) and size of training set (16.6K, 33K and 60K hours of transcribed speech). The best performing model, measured both on validation perplexity and on SEMER on an annotated test set that spanned September to December 2019, was a s2s-AE model with two transformer layers on both encoder and decoder, 512 units per layer, trained on 60K hours of automatically transcribed speech using a BPE vocabulary of 50k tokens.

The s2s-AE is hooked into our baseline NLU model by discarding the decoder and passing the latent code (sentence embedding) into the Intent Classification (IC) block; and by passing the sequence of hidden states into the Named Entity Recognition (NER) block. A block diagram for the resulting NLU model can be seen in Figure 2. Injecting the s2s-AE into the baseline model without any skill-specific fine-tuning yields SEMER relative improvements of 1.53% on our seven month anno-

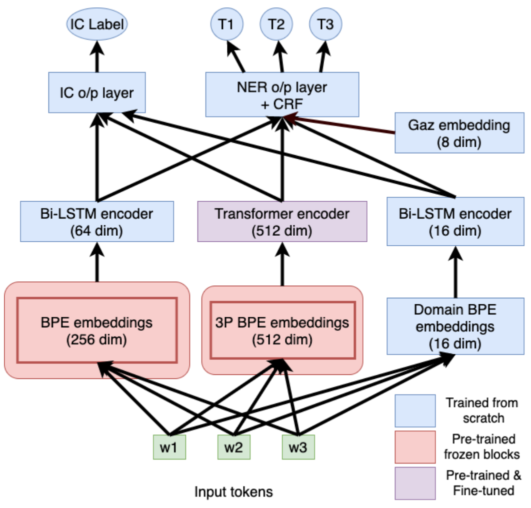

Figure 2: A Baseline 3P NLU model with a Sentence Embedding from the s2q-AE.

tated test set. Using an additional annotated test set that spanned September to December 2019, and by trying several learning rate multipliers (LRMs) for our seq2seq encoder, we found the overall best LRM was 0.0 (no fine-tuning), followed by 0.1 (which caused a -0.31% relative regression on SEMER). This indicates that for fine-tuning to work, it needs to be done in a skill-specific fashion.

Following the same steps for skill-specific fine-tuning as for maximal FST-matching and for the MSR threshold parameter, we use one annotated dataset from September to December 2019 for LRM selection and an annotated test set that spans January to July 2020 for evaluation. We choose the best LRM for each of the 86 skills on the first dataset, obtain SEMER values on the second one, and average them, observing a 4.73% relative improvement.

5.3 Data Augmentation via Tri-training Ensembles

We start by training a tri-training ensemble of three identical NLU models with different seeds to diversify the initial training/validation sets and the training algorithm. The tri-training stopping criterion was set to either reaching an average SEMER on the validation sets of 0.0 or run for a maximum of three iterations. Upon finishing, we obtained live utterances labeled with high-precision labels. These are utterances that all three models had complete agreement on IC and NER annotations (not majority vote agreement). On average, each skill had their training set size increased by 138.54%, ranging from 7.11% to 1089.57%.

We trained NLU models on new training sets formed by the baseline 10k training utterances and up to another 10k utterances from the live utterances with high-precision labels. The average training set went up to 16K utterances per skill and the average SEMER was improved with a 2.56% relative improvement. Without capping the amount of live data added to the training set, the SEMER is further improved with a 2.91% relative improvement on SEMER, and a 4.45% relative improvement on IRER. Despite the good initial results, 28 skills suffered SEMER degradation.

The second ensemble we tried consists of a baseline NLU model, the NLU model with an additional encoder pre-trained as an auto-encoder model on live utterances from Section 5.2, and an NLU BERT-based model. We use the same stopping criterion. By the end of tri-training, we obtained high-quality labels that increased the domain training sets for an average of 99.15% per skill.

Again, we trained NLU models without a maximum training size, with a validation set of 10% of these utterances. This time we obtained a relative improvement of 7.65% on SEMER. The IRER relative improvement is 9.67%. A total of 18 skills suffered SEMER degradation with this ensemble.

We used our trained tri-ensemble to infer high-precision labels on the test set and measure how accurate the high-precision labels really are. Using a setting where an utterance is only retrieved with complete agreement in the ensemble, the average coverage was 63.88%.

Tri-training theory assumes that the ensemble classifiers are independently trained. Here we explore the addition of an altogether different type of classifier to our collections of neural classifiers, namely SVMs for IC and NER. Both use BPE word embeddings as features. For IC, we extract the BPE embeddings and concatenate the max, min, sum and mean vector of the embedding sequence for a total of 1024 dimensions. For NER, we map every word to the sum of its BPE embeddings. Both models use linear kernels, since the size of our datasets and the need to perform grid search make non-linear kernels impractical.

We built a 4-classifier ensemble by adding the SVM classifiers to the previous ensemble composed of a NLU model, a NLU model with an additional encoder pre-trained as an auto-encoder model on live utterances, and an NLU BERT-based model. In this occasion the addition of the SVM model causes the test coverage ratio to be greatly reduced to 55%.Furthermore, the NLU DNN model trained on the additional data labeled by this ensemble achieves 7.27% and 8.32% relative improvements on SEMER and IRER, which is small reduction from the previous best ensemble.

Model	SEMER Δ%	IRER Δ%
+M-FST	5.01	1.44
+Tri	7.65	9.67
+M-FST&Tri	10.54	11.49

Table 1: NLU Performance for Base Model with three Data Augmentation options. Relative improvement values are computed with respect to the Base Model without data augmentation.

5.4 Cumulative Experiments

In this section we present some cumulative results obtained by applying combinations of the three techniques. We start by analyzing the performance of our baseline NLU model (Base) by itself; b) after adding fine-tuned maximally FST-matched data (FT-FST); c) after adding SSL data from tri-training (Tri); and d) after adding both FST-matched data and tri-training data (Table 1). Fine-tuned FST-matched data yields relative improvements of 5.01% and 1.44% on SEMER and IRER respectively, tri-training data yields 7.65% and 9.67% relative improvements on SEMER and IRER, and both combined deliver 10.54% and 11.49% relative improvements.

Next, we analyze the performance of the NLU model that incorporates a sentence embedding from an auto-encoder fine-tuned specifically for every skill (Base+SE). By adding fine-tuned maximally FST-matched data, SEMER and IRER improve by 8.23% and 7.73%, respectively. Adding tri-training data yields 8.33% and 10.81% relative improvements on SEMER and IRER, respectively, while both combined deliver relative improvements of 11.77% and 12.94% on SEMER and IRER. All these relative improvements are with respect to the baseline NLU model (without data augmentation). See Table 2.

6 Conclusions

We presented three different approaches to improving NLU performance for skills that have limited amounts of annotated data. The two data augmentation approaches, based on maximal fst-matching and tri-training ensembles, yield considerable relative improvements of 5.01 and 7.65%. A third

Model	SEMER Δ%	IRER Δ%
Base+SE	4.73	4.51
+M-FST	8.23	7.73
+Tri	8.33	10.81
+M-FST&Tri	11.77	12.94

Table 2: NLU Performance for Base+SE Model with three Data Augmentation options. Relative improvement values are computed with respect to the Base Model performance.

approach, based on injecting sentence embeddings obtained from an auto-encoder pre-trained on live traffic, gave 4.73%. The combination of the three techniques attained a total relative improvement of 11.77%. Overall, adding these three methods into our pipeline improves NLU performance, leverages automatically transcribed user traffic for all skills, lessens the need for developers to provide annotated data, and eliminates the need for internal human-based annotations for training better models.

Acknowledgments

The authors are thankful to Saleh Soltan and the rest of the DeepNLU team for some of the pre-trained models and components that have been used in this study, and to Lisheng Fu, Matthew Trager and Sandesh Swamy for their assistance.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20(3):542–542.

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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2-3):133–168.

Giannis Karamanolakis, Subhabrata (Subho) Mukherjee, Guoqing Zheng, and Ahmed H. Awadallah. 2021. Self-training with weak supervision. In *NAACL 2021*. NAACL 2021.

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 84–92.

Dong-Hyun Lee. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*.

David D Lewis and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in neural information processing systems*, pages 6294–6305.

Stanislav Peshterliev, John Kearney, Abhyuday Jagannatha, Imre Kiss, and Spyros Matsoukas. 2019. Active learning for new domains in natural language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 90–96, Minneapolis, Minnesota. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

H Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1065–1073, Beijing, China. Coling 2010 Organizing Committee.

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

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Trans. on Knowl. and Data Eng.*, 17(11):1529–1541.