Data selection and semi-supervised slot tagging for artificial intelligent agents

Minmin Shen
Amazon Alexa
{shenm

James Zhu
Amazon Alexa
jameszhu

Suranjit Adhikari
Amazon Alexa
suranjit

Angeliki Metallinou
Amazon Alexa
ametalli}@amazon.com

Abstract

We propose a semi-supervised learning framework to boost the performance of slot tagging in the low resource case, where we only have a small labeled target dataset available for model training, but we have access to a large unlabeled source dataset. Our framework consists of two components: first performing data selection to find a subset of the source data that is semantically similar to the target data, and second, training the model using both selected and target data through a combination of semi-supervised training techniques, inspired from self-training and self ensembling. We apply our techniques to a challenging slot tagging setup for a commercial artificial intelligent agent where unlabeled source data originates from largely different domains compared to the target data. We empirically show that our proposed techniques achieve up to 7% relative gain in low resource slot tagging compared to a strong pretrained model fine-tuned on target training data.

1 Introduction

Artificial intelligent agents have become widespread and popular among consumers as they facilitate practical everyday tasks, such as question answering, playing media, etc., through intuitive, voice-powered interfaces. Their popularity relies not only on correctly recognizing a user's request, but also on continuously expanding the range of functionality that they can service, so that they become increasingly useful. At the same time, there is a range of commercial services, such as the Alexa Skills Kit by Amazon (Kumar et al., 2017) or DialogFlow by Google (DialogFlow), that enable developers to build their own custom natural language understanding (NLU) functionality for artificial agents.

The NLU models that power this functionality typically require large amounts of labeled data to reach high accuracy. However, many practical NLU tasks in commercial settings are low resource in terms of labeled training data, especially when a new domain is being developed. At the same time, in commercial artificial agents, there are typically large amount of unlabeled text data from the output of the Automatic Speech Recognition (ASR) engine, obtained through interactions between the user and the agent. As a result, semi-supervised methods have been exploited to leverage large unlabeled or partially labeled datasets for improving performance in low resource settings (Ruder and Plank, 2018).

In this work, we focus on developing semi-supervised algorithms to improve slot tagging (ST) for a commercial artificial agent. The ST component of an NLU system is responsible for recognizing entities of interest, also called *slots*, in a user's request, such as people names, song titles, locations, dates etc. This is typically done through sequence tagging by employing neural network based architectures (Goyal et al., 2018; Liu and Lane, 2016; Huang et al., 2015). Correct recognition of slots is crucial for accurately servicing the user's request, therefore the ST component accuracy has a large impact on overall system accuracy.

Within the semi-supervised learning field, existing algorithms use the small labeled target dataset to train one or multiple models in a supervised way, and then apply those models to assign noisy labels to a large set of unlabeled source task data. Such techniques include self-training applied for parsing and word sense disambiguation (McClosky et al., 2006; Yarowsky, 1995), tri-training applied for text and web-page classification (Zhou and Li, 2005; Ruder and Plank, 2018), and recently proposed self ensembling applied for image classification (Laine and Aila, 2016). While such semi-supervised techniques have been successful for image classification and text classifica-

tion problems, they have shown mixed or smaller success for some sequence tagging tasks (Ruder and Plank, 2018). This indicates that it may be more challenging to incorporate noisily labeled training data for sequence tasks like slot tagging, compared to classification problems. Also, prior work on semi-supervised learning often assumes the source and target tasks are closely semantically related (McClosky et al., 2006), while for dissimilar tasks such techniques may not be beneficial (Asch and Daelemans, 2016). Nevertheless, in many practical settings this assumption of semantic relatedness does not hold. In our commercial NLU setup, we have large amounts of unlabeled ASR output text coming from a wide range of functionality domains, potentially developed by external developers for their custom NLU applications. User requested slots from such source functionality may be very different from the target domain slots. Therefore, it becomes important to select a subset of the source data that is *semantically similar* to the target data, such that the former can be used in a semi-supervised setting without introducing out-of-domain noise in the slot tagger.

We address these challenges by introducing a framework that combines semi-supervised learning and data selection to improve low resource ST. We evaluate our method in target ST tasks from a popular commercial artificial agent, where source data comes from a large pool of ASR text from hundreds of functionality domains. We show that our proposed method is able to boost ST accuracy, achieving up to 7.7% relative improvement (depending on the task and data setup), compared to a strong pre-trained baseline model, fine-tuned on the target training data. Overall, our contributions are three fold. We propose a semi-supervised learning framework that combines techniques from self-training (Yarowsky, 1995) and self ensembling (Laine and Aila, 2016) with data selection. We also present and compare three different method variations for effectively training a state-of-the-art ST model on a combination of labeled and unlabeled data. Finally, we evaluate our method on data from a commercial agent where the source data originate from largely different domains and data distributions compared to the target, and show the validity of our approach in this challenging practical setup.

2 Related work

The semi-supervised learning paradigm assumes access to a small labeled target training set, D_l, and a much larger unlabeled source dataset, D_u, and explores techniques for effectively leveraging the unlabeled data D_u when training models for the target task (Zhu, 2005). Typically, one of semi-supervised learning techniques use D_l to train one or multiple models in a supervised way, and then apply those models to assign noisy (*pseudo*) labels to the D_u. Self-training is a classical semi-supervised technique that follows this idea (Yarowsky, 1995; McClosky et al., 2006). Self-training is an iterative algorithm where target domain models are progressively trained on both clean and noisy labels, and then applied for obtaining updated noisy labels.

More recently, semi-supervised learning techniques that leverage deep learning models, have been successfully applied for tasks like image classification (Lee, 2013; Ding et al., 2018) and slot filling (Bapna et al., 2017). In (Lee, 2013), *pseudo labels* correspond to the class that has the maximum predicted probability as the ground truth label for unlabeled data. In this way, the pseudo-labeled data are trained with existing labeled data in a supervised learning fashion, i.e., adding a cross-entropy loss of pseudo-labeled data in the overall loss function. According to (Lee, 2013), this added loss is essentially *entropy regularization*, a scheme favors low density separation between classes by minimizing the conditional entropy of class probabilities of unlabeled data. In (Ding et al., 2018), *self ensembling* is applied to train a deep neural network in the semi-supervised learning manner. The main idea of this approach, also called Π-method, is to regularize the network to force it to generate approximately the same output for the same input which undergoes data augmentation and different dropout conditions. A *consistency loss* is used to penalize the difference of two model outputs obtained by evaluating the same input after different data augmentation with the same neural network under different dropout conditions.

Alternatively, recent work relies on pre-training robust neural network models on large amounts of existing data sources, and then fine-tuning those networks on the target training tasks (Peters et al., 2018; Howard and Ruder, 2018; Goyal et al., 2018). In (Goyal et al., 2018) authors pre-train

a model on existing labeled data for slot tagging from related domains, and fine-tune on the target task. Here, we use model pre-training techniques like (Goyal et al., 2018) as a strong baseline, and we explore whether classical semi-supervised ideas that incorporate noisy labels into training can further boost accuracy for low resource settings.

3 Methodology

We describe our methods in the context of a sequence tagging task. Given a small labeled target dataset D_l and a large unlabeled source dataset D_u, we first select a subset of D_u that is semantically similar to D_l, denoted as D_u^s. The input utterances we use for training our ST model are $\mathbf{x}_i, i = 1, ..., M + N$, where $M = |D_l|$ is the number of labeled utterances and $N = |D_u^s|$ is the number of selected unlabeled utterances, for which we need to obtain pseudo labels through semi-supervised techniques.

Our framework consists of four steps. For step 1, we train a state-of-the-art slot tagger model on the labeled data, here using a bi-LSTM-CRF model (Huang et al., 2015). This model will be used to generate pseudo labeled data. For step 2, from the source dataset we select the data that is most semantically similar to the target dataset (see Section 3.1). The selected utterances are fed to the trained bi-LSTM-CRF model, that generates the hypothesis labels and associated confidence scores. We also optionally filter-out utterances with low confidence scores. For Step 3, we explore three different semi-supervised learning methods to re-train the model with the target dataset and pseudo-labeled data, which will be elaborated in Section 3.2. Finally, we fine-tune our model over the target dataset with a very small learning rate. This was suggested in (Ding et al., 2018), and we empirically found it useful for further boosting performance.

3.1 Data selection

The source data comes from ASR output text from hundreds of source NLU functionality domains developed by external developers of our commercial agent. We also have noisy labels of the source domains obtained from tags generated by the external develpers about their application (the label space is different from target functionality domains). The aim of data selection is to select utterances from the source domains that tend to be most semantically similar to the target domains. For example, assume that a target low resource functionality is about requesting cooking recipes. We would like to select data from semantically related domains, such as requesting for food calories, food information, making meal plans, as opposed to unrelated domains such as airplane booking or playing music.

To extract the semantic representation of an utterance, we perform weighted average of pre-trained word embeddings using Term Frequency Inverse Document Frequency (TF-IDF) weights. The IDF weight is computed in a standard manner, as $log((1+U)/(1+n_w))$ where U is the total number of utterances in the corpus, n_w is the number of utterances that word w occurs. Similarly, the TF weight is computed as o_w/n_{token} where o_w is the occurrence of word w, and n_{token} is the total number of tokens in the corpus. We use pre-trained FastText embeddings (Bojanowski et al., 2016).

Our data selection technique follows the idea of similarity. Assuming that each target functionality utterance is a point in a semantic embedding space defined by the representation we described above, we select the source utterances, originating from source functionality domains, that are closest to the target utterance in the embedding space. For example, given a target utterance like 'find a recipe for apple pie', we would look for source utterances that are nearest neighbors in the embedding space. Detailed description could be found in Appendix A.

Our original ASR data pool contains millions of utterances from hundreds of domains. Using the selection technique described above, we end up selecting utterances in the order of tens of thousands, originating from around 50 domains.

3.2 Semi-supervised learning methods

As for Step 3, we experiment with three different approaches of training a sequence tagging model on both labeled data and pseudo labeled data in a semi-supervised fashion. Pseudo code of the three algorithms is presented in Appendix C.

Self-training Conventional self-training (Yarowsky, 1995; McClosky et al., 2006) is one of the techniques that we explore. In essence, it is an iterative training process that uses a trained model to produce pseudo labels for unlabeled data and incorporate the additional data in the next iteration of model training. The filtering mechanism and stopping criterion are the two critical steps in this conventional process. Here, we follow (Ab-

ney, 2007) and choose the average log-likelihood scores of predicted slots per utterance as a filtering criterion. We choose 10% of the unlabeled data set as the throttling level for the filtering process, i.e., we discard the bottom 90% of the utterances with the lowest average log-likelihood scores. As for the stopping criterion, we run the algorithm for a fixed number of R=20 rounds at first, and then examine the dev set performance and select the model that achieves the highest accuracy on the dev set. We notice that the performance (F1 score) on the dev data tends to converge after around 15 iterations. One disadvantage of this self-training algorithm is that it requires multiple iterations of full model training and therefore significantly increases the overall training time.

Semi-supervised learning with loss re-weighting (LW) Here, we describe a semi-supervised technique that does not require multiple model training iterations, therefore reducing model training time compared to self-training. We train the model only once with the combination of labeled and pseudo-labeled data, but update the pseudo-labels after each epoch by assigning the most probable label. Inspired by (Lee, 2013) that applied similar ideas for image classification, we also re-weight the loss of labeled and pseudo labeled data during training and assign a lesser weight to the pseudo labeled data. Deterministic annealing is applied to trade-off the benefits and the instability of introducing pseudo labeled data for training a better model: the weight of loss of pseudo labeled data progressively increases as the number of epochs increases.

Moreover, we divide the traning process into two stages: First, the whole bi-LSTM-CRF model is trained on labeled data for $T1$ epochs. Afterwards, we freeze the CRF layer, and only re-train the bi-LSTM layers on both labeled and pseudo labeled data following the loss function in Eq. (1):

$$L_1 = -\frac{2}{|B|}\sum_{i\in(B\cap D_l)} logz_i[y_i] - \alpha_t\frac{2}{|B|}\cdot \sum_{i\in(B\cap D_u^s)} logz_i[y_i'] \quad (1)$$

In this loss, the first term is the standard cross-entropy on the labeled data, where y_i is the annotated label and z_i is the predicted probability vector. $z_i[y_i]$ is the y_i-th element in z_i, corresponding to the annotated label. The second term is the cross-entropy on the pseudo-labeled data where y_i' is the pseudo label. $|B|$ is the batch size, e.g., we split each training batch equally between labeled and pseudo labeled utterances. α_t is the weight of the loss of pseudo labeled data, computed empirically as in (Lee, 2013). The strategy of including the pseudo labeled data only for the bi-LSTM layers and not in the CRF training empirically increases training stability. This could be because the CRF model is stronger at learning sequential information, thus easier to be misled by incorrect pseudo labels.

Self-ensembling based method (SE) In (Laine and Aila, 2016), authors introduce a self-ensembling algorithm for semi-supervised image classification where the model is encouraged to produce consistent network outputs across training epochs with perturbed input data. Their technique is motivated by the fact that typically an ensemble of multiple models performs better than a single model in the ensemble, and also by the idea that dropout regularization can be seen as a form of ensembling where the complete network can be seen as an implicit ensemble of sub-networks trained with dropout. In (Laine and Aila, 2016), image data augmentation (e.g. image flipping, scaling and shifting), along with dropout, is applied to the input x_i to get a perturbed input $\widetilde{x_i}$, which produces the two model predictions z_i and $\widetilde{z_i}$.

Here, we apply similar self-ensembling ideas for our slot tagging task. Specifically, we use dropout regularization of the input utterances as our form of data perturbation, leaving more advanced forms of text data augmentation as future work. In the loss function, we add an additional loss term as the L2 loss of the difference between z_i and $\widetilde{z_i}$, which is named *consistency loss* following (Ding et al., 2018). This loss penalizes different predictions for the same training input x_i by taking the mean square difference between the prediction vectors z_i and $\tilde{z}_i$, which are evaluated by the model twice with dropout. Thus the combined loss function is below, where the first two terms are as in Eq. (1), while the last term is enforcing consistency (β is a constant that upweights the *consistency loss*).

$$L_2 = -\frac{2}{|B|}\sum_{i\in(B\cap D_l)} logz_i[y_i] - \alpha_t\frac{2}{|B|}\cdot \sum_{i\in(B\cap D_u^s)} logz_i[y_i'] + \beta\frac{2}{|B|}\sum_{i\in(B\cap D_u^s)} \|z_i - \widetilde{z_i}\|^2 \quad (2)$$

3.3 Baseline pre-training method

We pre-train our baseline bi-LSTM-CRF slot tagger on a set of around 4 million labeled utterances originating from around 20 labeled NLU domains, and then fine tune our model on the set of target labeled utterances D_l. Again, we are inspired from computer vision literature, where typical neural baselines are pre-trained models on large image datasets (Donahue et al., 2014; Sharif Razavian et al., 2014). Here, we explore whether our proposed semi-supervised framework that incorporates noisily labeled data selected from ASR output across multiple functionality domains, can further improve the accuracy of this strong pre-trained baseline model for a low resource ST task.

4 Experimental Results

We evaluate the semi-supervised learning experiments on data sets from two NLU domains and compare with the baseline that does not use any semi-supervised learning techniques, but starts from a pre-trained model. These two domains are used as examples for showing the efficacy of our techniques in practice. Domain 1 relates to cooking and recipes, and contains 46 slots. Domain 2 is about asking information on businesses around the user's location, and contains 41 slots. The examples of utterances are shown in Table 2 of Appendix B. Each dataset contains core data ($\sim$ 300 or 500 utterances) and bootstrap data ($\sim$ 43k utterances). Core data refers to example utterances designed by user experience engineers that describe the main functionality that each domain supports. Bootstrap data refers to domain data collection and generation of synthetic utterances plus user data collected from user interactions with our agent. In addition to these labeled datasets D_l, we created unlabeled data pool D_u^s for each domain using the data selection process mentioned in 3.1. The size of the unlabeled data pool is about 30k.

4.1 Results and Analysis

The experiment results for the two data sets are summarized in Fig. 1, where we present F1 scores for slot labels. Bar plots represent the mean value and confidence interval (CI) of F1 scores for three algorithms: the baseline algorithm with a pre-trained model, described in Section 3.3, and the LW algorithm and SE algorithms with bootstrap data D_l plus unlabeled data D_u^s, described in Section 3.2. Star shape markers are F1 scores achieved from the conventional self-training method (Section 3.2). Following the conventional self-training process, we iteratively train the model and select the model with the best dev set performance, hence we do not perform confidence interval analysis for this method. Note that since the self-training model is selected as the best performing among multiple self-training iterations, it has a comparative advantage to the LW and SE algorithms that are trained only for one iteration. Meanwhile, self-training is also significantly slower in total model training time compared to the LW and SE algorithms.

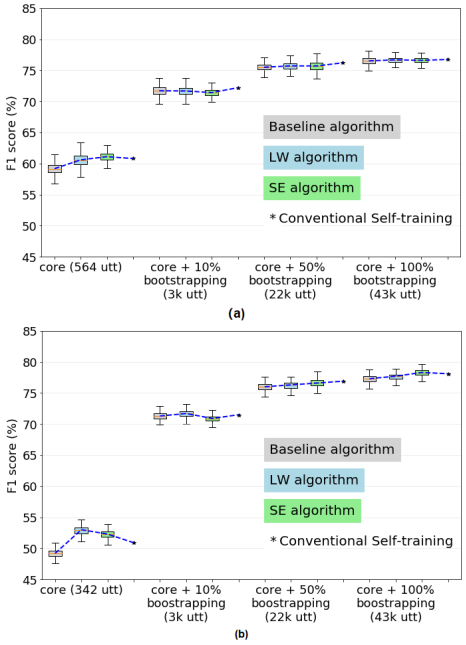

Figure 1: Slot tagging model accuracy (F1 score) between the three semi-supervised techniques and a strong pre-trained baseline model for the two examined domains.

Table 1 shows the detailed F1 scores for each method across the two domains and the different data settings. We ran t-tests for all experiments to check if the gain of F1 scores for LW and SE algorithms are statistically significant. Except the experiments using Domain 1 data with 10% bootstrapping data, the gain of F1 scores in rest experiments are all significant. As to compare between LW and SE methods, the t-test results confirm that it is difficult to conclude which method is more superior. Furthermore, we ran experiements using the LW and SE methods without the optional filtering step, as we empirically found that filtering does not bring much additional gain for these methods. Similar to findings in related

Table 1: Experimental Results for Domain 1 and 2 for the Baseline and the three examined semi-supervised methods w/ filtering data. F1 scores and rel % gains compared to the Baseline across data settings are denoted as: Core, +10%, corresponding to Core+10% boostrap data. The performance gain in bold are statistically significant.

	Domain 1							Domain 2						
	Base	LW		SE		Self-train.		Base	LW		SE		Self-train.	
	F1	F1	gain	F1	gain	F1	gain	F1	F1	gain	F1	gain	F1	gain
Core	59.1	60.5	**2.4**	61.2	**3.6**	60.8	**2.9**	49.2	50.5	**2.7**	52.2	**6.1**	50.9	**3.5**
+10%	71.6	71.7	0.01	71.3	-0.4	72.2	**0.8**	71.3	71.9	**0.8**	71.9	**0.8**	71.5	**0.3**
+50%	75.6	75.8	**0.3**	75.7	**0.1**	76.2	**0.8**	76.1	77.2	**1.4**	77.0	**1.2**	76.9	**1.1**
+100%	76.5	76.8	**0.4**	76.6	**0.1**	76.7	**0.3**	77.3	78.3	**1.3**	78.3	**1.3**	78.1	**1.0**

work (Ruder and Plank, 2018), for conventional self-training, filtering data is essential for performance improvement. However, for LW and SE algorithms the filtering step is not necessary. As a matter of fact, we observe the strongest improvement of 7.7% for the experiment using the LW method on domain 2 with only core data without this optional step. The advantage of performing the filtering step is that it provides more consistant results in the performance gain.

Overall, all three semi-supervised learning methods improve performance over the pretrained baseline model for almost all data settings. When only core data is used for training, the performance improvement from all semi-supervised learning algorithms is the largest. This indicates that even a strong pretrained baseline can benefit from additional pseudo labeled data, especially in very low resource settings with few hundreds of in-domain training utterances. From Fig. 1(b) we observe the strongest improvement for the experiment using the SE method on domain 2 with core data only. This corresponds to a gain of 6.1% relative in terms of F1 score, as seen in Table 1.

As more labeled data is added for training, the F1 scores from all methods including the baseline method increase as expected. The performance improvement from semi-supervised algorithms is not as pronounced as in the core data setting because the extra labeled data dilutes the effect from the pseudo labeled data during the training process. The size of just 10% labeled data is on the same order as the filtered pseudo labeled data, which is also about 3k. Among the three semi-supervised learning methods, in most cases, the F1 scores of algorithms LW and SE are higher than conventional self-training, with the added advantage of 10-20x faster total training time (because LW and SE are not trained iteratively). The domain specific characteristics of the labeled data affect the performance improvement of the semi-supervised methods. For example, we notice that domain 2 clearly benefits from semi-supervised learning compared to the baseline model, even for data settings where more bootstrap data is added for training, while for domain 1 the benefit is smaller. This could be a result of the quality of the initial core annotated data, where noisier data may harm the quality of assigned pseudo labels (e.g., we noticed larger annotation noise for domain 1), and also of how similar are the source domains in the available unlabeled data pool to each of the target domains. We leave further investigation as future work.

5 Conclusions

We described a semi-supervised learning framework for leveraging large unlabeled datasets in order to improve slot tagging accuracy (ST) in low resource settings. Our framework combines ideas from data selection along with techniques like self ensembling that were previously introduced in the context of image classification tasks. Our method is applied to improve ST accuracy for a commercial agent by first selecting utterances that are semantically similar to the target domain from a large data pool of diverse functionality domains, and then by assigning pseudo labels to the selected data using semi-supervised learning. We evaluate a variety of semi-supervised methods and compare them in terms of accuracy and model training efficiency. Overall, our methods improve ST performance up to 7.7% relative for low resource settings on top of a strong pre-trained baseline model.

References

Steven Abney. 2007. *Semisupervised learning for computational linguistics.* Chapman and Hall/CRC.

Vincent Van Asch and Walter Daelemans. 2016. Predicting the effectiveness of self-training: Application to sentiment classification. *arXiv preprint arXiv:1601.03288*.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. *arXiv preprint arXiv:1707.02363*.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

DialogFlow. https://dialogflow.com.

Yifan Ding, Liqiang Wang, Deliang Fan, and Boqing Gong. 2018. A semi-supervised two-stage approach to learning from noisy labels. *arXiv preprint arXiv:1802.02679*.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

A. Goyal, A. Metallinou, and S. Matsoukas. 2018. Fast and scalable expansion of natural language understanding functionality for intelligent agents. In *Proc. of NAACL*.

J. Howard and S. Ruder. 2018. Universal language model fine-tuning for text classification. In *Proc. of ACL*.

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

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*.

Samuli Laine and Timo Aila. 2016. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*.

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*.

B. Liu and I. Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *In Proc of Interspeech 2016*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proc. NAACL-HLT*.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. *arXiv preprint arXiv:1804.09530*.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc of ACL*.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *EEE Transactions on Data Engineering*, 17(1):1529–1541.

Xiaojin Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

A Appendices

Details of data selection procedure is as follows.

- For each target utterance x_t find the top-K nearest neighbor source utterances x_s from the unlabeled set D_u. For each source utterance x_s, we also know the noisy domain label d_s.
- Collect all selected utterances x_s to create a preliminary selected set D_u^{prelim}
- Select a subset of the utterances in D_u^{prelim} that correspond to the top-N most commonly occurring domains d_s in D_u^{prelim}. Those utterances make up our final set of D_u^s, containing data from N source domains that are are semantically related to the target domain.

B Appendices

Table 2: Examples of predictions from the baseline and the SE algorithm with only core data available as training data in Domain 1 and 2. Entity labels are next to the '|' symbol, while "Other" labels are not shown. Green labels are correct, while red are incorrect counterparts.

	Domain 1	Domain 2
Baseline algorithm	search for **oatmeal**|**DishName** **bread**|**DishName** recipe|InstructionType from king|DishName arthur's|DishName	is toys|PlaceName r.|PlaceName us|PlaceName opened right **now**|**Time**
Semi-supervised learning algorithm	search for **oatmeal**|**FoodItem** **bread**|**FoodItem** recipe|InstructionType from king|FoodIngredient arthur's|FoodIngredient	is toys|PlaceName r.|PlaceName us|PlaceName opened right **now**|**Date**

C Appendices

Algorithm 1 Conventional Self-training Algorithm

Required: x_i = training data
Required: D_l = set of training input indices with ground truth labels
Required: y_i = labels for labeled inputs
Required: $g_{\theta^+}(x)$ = two bi-LSTM layers + CRF layer of a pre-trained model with parameters θ^+
Required: $f_\theta(x)$ = two bi-LSTM layers of the model with parameters θ
Required: T = number of epochs for each iteration

for r in [1, R] **do**
    for t in [1, T] **do**
        for each minibatch B **do**
            $z_{i \in B \cap D_l} \leftarrow g^t_{\theta^+}(x_{i \in B \cap D_l})$ ▷ re-train the whole network
            loss $\leftarrow -\frac{1}{|B \cap D_l|} \sum_{i \in (B \cap D_l)} logz_i[y_i]$ ▷ supervised loss component
            $-\frac{1}{|B \cap D^s_u|} \sum_{i \in (B \cap D^s_u)} logz_i[y'_i]$ ▷ supervised loss of pseudo labeled data
            update θ^+ using optimization method e.g. Adam
        end for
    end for
    $D^s_u(r) \leftarrow D^s_u(r-1)$ ▷ filter top ranked confident data
    $y'_{i \in B \cap D^s_u}$ = softmax($z_{i \in B \cap D^s_u}$) ▷ update the reference labels by the predictions at r
end for
return $g^1_{\theta^+}, g^2_{\theta^+}, \cdots, g^r_{\theta^+}$ ▷ return r models

Algorithm 2 LW algorithm

Required: x_i = training data
Required: D_l = set of training input indices with ground truth labels
Required: y_i = labels for labeled inputs
Required: $g_{\theta^+}(x)$ = two bi-LSTM layers + CRF layer of a pre-trained model with parameters θ^+
Required: $f_\theta(x)$ = two bi-LSTM layers of the model with parameters θ
Required: T_1, T_2 = number of epochs for stage 1, 2

Stage 1:
for t in [1, T_1] **do**
    for each minibatch B **do**
        $z_{i \in B \cap D_l} \leftarrow g^t_{\theta^+}(x_{i \in B \cap D_l})$ ▷ re-train the whole network
        loss $\leftarrow -\frac{1}{|B|} \sum_{i \in (B \cap D_l)} log z_i[y_i]$ ▷ only re-train on labeled training data
        update θ^+ using optimization method e.g. Adam
    end for
end for
return $g^{T_1}_{\theta^+}$

Stage 2:
for t in [T_1+1, T_2] **do**
    for each minibatch B **do**
        $z_{i \in B \cap D^s_u} \leftarrow f^{t-1}_\theta(x_{i \in B \cap D^s_u})$ ▷ only re-train bi-LSTM layers
        $y'_{i \in B \cap D^s_u}$ = softmax($z_{i \in B \cap D^s_u}$) ▷ update the reference labels as the predictions at $t-1$
        loss $\leftarrow -\frac{2}{|B|} \sum_{i \in (B \cap D_l)} log z_i[y_i]$ ▷ supervised loss of labeled data
        $-\alpha_t \frac{2}{|B|} \sum_{i \in (B \cap D^s_u)} log z_i[y'_i]$
        update θ using optimization method e.g. Adam
    end for
end for
return $f^{T_2}_\theta$

Algorithm 3 SE algorithm

Required: x_i = training data
Required: D_l = set of training input indices with ground truth labels
Required: y_i = labels for labeled inputs
Required: $g_{\theta^+}(x)$ = two bi-LSTM layers + CRF layer of a pre-trained model with parameters θ^+
Required: $f_\theta(x)$ = two bi-LSTM layers of the model with parameters θ
Required: T_1, T_2 = number of epochs for stage 1, 2

Stage 1: Same as LW algorithm

Stage 2:
for t in [T_1+1, T_2] **do**
    for each minibatch B **do**
        $z_{i \in B \cap D_u^s} \leftarrow f_\theta^{t-1}(x_{i \in B \cap D_u^s})$ ▷ evaluate network outputs for the input
        $\widetilde{z}_{i \in B \cap D_u^s} \leftarrow f_\theta^{t-1}(x_{i \in B \cap D_u^s})$ ▷ evaluate network outputs for the same input
        $y'_{i \in B \cap D_u^s}$ = softmax($z_{i \in B \cap D_u^s}$) ▷ update the reference labels as the predictions at $t-1$
        loss $\leftarrow -\frac{2}{|B|} \sum_{i \in (B \cap D_l)} log z_i[y_i]$ ▷ supervised loss of labeled data
        $-\alpha(t) \frac{2}{|B|} \sum_{i \in (B \cap D_u^s)} log z_i[y'_i]$ ▷ supervised loss of pseudo labeled data
        $+\beta \frac{2}{|B|} \sum_{i \in (B \cap D_u^s)} \|z_i - \widetilde{z}_i\|^2$ ▷ **unsupervised loss component**
        update θ using optimization method e.g. Adam
    end for
end for
return $f_\theta^{T_2}$