Few-shot Pseudo-Labeling for Intent Detection

Thomas Dopierre[12] and **Christophe Gravier**[1] and
Julien Subercaze[1] and **Wilfried Logerais**[2]

[1]Laboratoire Hubert Curien
Saint-Étienne, France
firstname.lastname@
univ-st-etienne.fr

[2]Meetic
Paris, France
{t.dopierre,w.logerais}@
meetic-corp.com

Abstract

In this paper, we introduce a state-of-the-art pseudo-labeling technique for few-shot intent detection. We devise a folding/unfolding hierarchical clustering algorithm which assigns weighted pseudo-labels to unlabeled user utterances. We show that our two-step method yields significant improvement over existing solutions. This performance is achieved on multiple intent detection datasets, even in more challenging situations where the number of classes is large or when the dataset is highly imbalanced. Moreover, we confirm these results on the more general text classification task. We also demonstrate that our approach nicely complements existing solutions, thereby providing an even stronger state-of-the-art ensemble method. We make our code publicly available[1] for future research.

1 Introduction

Labeling user utterances as intents is of paramount importance for chatbots. Intent classes are usually engineered by a linguistic team, and these classes are likely to evolve with time. The most frequent situations include new unanticipated intents found by dialogue traces analysis, the need to introduce new intents that were previously handled using web forms, or the urge to split existing intents into several new ones for fine-grained user interactions. Regardless the rationale, each time the intent referential evolves, a significant amount of new labeled data is required to train a decent intent detection model. This calls for a time-consuming and error-prone labeling process – it is therefore expensive.

Consequently, intent detection systems need robust models in few-shot settings in order to avoid to repeatedly suffer from labeling user utterances into intents. A solution to this problem is to generate pseudo-labels for unlabeled utterances. A pseudo-label is a label (intent class) automatically assigned to an unlabeled utterance using the knowledge found in the set of labeled data. The resulting dataset composed of both labeled and pseudo-labeled utterances is then used to train a supervised intent detection model. Most pseudo-labeling methods combine advances in word representations (Peters et al., 2018; Bojanowski et al., 2017; Tissier et al., 2017) and then consider unsupervised clustering algorithms (von Luxburg, 2007) in order to propagate known labels to unlabeled data (see Section 2). Nonetheless, there are major drawbacks to existing pseudo-labeling algorithms. First, since clustering algorithms are used to partition user utterances representations, it leads to provide a pseudo-label to all unlabeled data. This is error-prone since cluster shapes for the entire dataset are unknown and hardly predictable in a few shot setting. Consequently, it is much harder to learn a discriminate hyperplane when training the downstream intent detection models since some data are incorrectly labeled. Worse still, this grows exponentially as the number of intents (classes) increases in practice since it is harder to predict correct pseudo-labels, and the hyperplane is more difficult to learn since it requires more discriminative power. A corollary is also that confidence values in pseudo-labels is hard to produce: since cluster shapes are unknown in advance, distances to centroids cannot reliably be used as a factor a confidence in a pseudo-label. In Computer Vision, the same issues exist for the image classification task and therefore end-to-end systems have been

[1]https://github.com/tdopierre/FewShotPseudoLabeling

Proceedings of the 28th International Conference on Computational Linguistics, pages 4993–5003
Barcelona, Spain (Online), December 8-13, 2020

proposed – Prototypical networks (Snell et al., 2017). Applied to intent detection, there are also part of related work and baselines in this work.

As contribution, we first present a new folding/unfolding hierarchical clustering algorithm for pseudo-labeling, which can be considered as a dynamic hierarchical clustering algorithm (Section 3). This original algorithm, by nature, is able to predict a pseudo-label only when it has sufficient clues (thereby reducing labeling noise) and to provide a pseudo-label normalized confidence score based on a temperature weighting mechanism – when training the downstream intent detection model using both labeled and pseudo-labeled utterances, those weights are used by penalizing the loss (Section 3.3). In order to validate our pseudo-label approach, we evaluate its impact on the downstream task of intent detection using four datasets and fine-tuned BERT-based language models (Section 4). Overall, in few-shot configurations and even when the number of classes is large, our proposed method is able to positively impact the intent or text classification tasks accuracies (Section 5), beyond the state-of-the-art. The experiments also exhibits a robustness of our methods to imbalance bias, which is common in practice. Finally, Section 6 concludes.

2 Related works

A growing amount of work has been done on the Conversational AI domain (Gao et al., 2018) over the recent years. There are different types of conversational agents: Question-Answering systems (Ren et al., 2018a), End-to-end Conversational Models (Vinyals and Le, 2015; Ren et al., 2018a), or Task-Oriented bots (Martin and Jurafsky, 2009). In the latter case, a central part of the process is the understanding of user utterances, a special case of text classification. Convincing approaches to this problem rely on neural networks (Lee and Dernoncourt, 2016), thus requiring large labeled datasets, which are costly to obtain. In practice, as more users interact with the conversational agent, it is easy to collect a large amount of raw data. Having access to a small labeled dataset along with a large unlabeled one calls for semi-supervised learning solutions (Chapelle et al., 2009; Zhu et al., 2005) for which label propagation is of paramount importance. One approach to this problem is self-training (Yarowsky, 1995; Mihalcea, 2004), where the model trains itself for multiple iterations, each time adding its most confident prediction to the training data. This approach has multiple weaknesses, the main one being the fact that the model cannot correct its own mistakes. One way of dealing with such a problem is to use tri-training (Zhi-Hua Zhou and Ming Li, 2005), a framework where three models are trained iteratively. At each step, common predictions of two models are added to the training set of a third. This continues until all models reach a stationary state. A substantial body of works focus on the few-shot classification problem, that is handling labeled data scarcity in the downstream training phase exclusively (Finn et al., 2017; Snell et al., 2017).

In our work, we tackle the problem differently: we propagate the knowledge we have on this small amount of data to unlabeled data points, before feeding both truly labeled and pseudo-labeled data to the intent detection system. This intermediate step is of the utmost practical interest: firstly, propagating knowledge on unlabeled points helps us understand how the data is structured. Secondly, in a labeling framework, it allows us to select interesting samples for our in-house linguists to label or correct given low pseudo-label confidence[2] as a human-in-the-loop approach, instead of selecting samples randomly. The more accurate the pseudo-labels, the more efficient it is for linguists.

From a broader perspective, it is worth noticing that semi-supervised learning (learning using both labeled and unlabeled data) has also received a lot of attention in computer vision (Vinyals et al., 2016), where annotations are time-consuming and error-prone as well. To overcome this problem, image-specific features – like coloration, pixel-level values – are used (Liu et al., 2018). Given a distance matrix between data points, labels from known samples are propagated to previously unlabeled data. Unfortunately this is hardly applicable in NLP as complex features are hardly intrinsic to words but rather based on distributional semantics. Indeed, in NLP we usually rely on pre-trained word embeddings (Mikolov et al., 2013) and build sentence representations by averaging word vectors (Joulin et al., 2017) or contextualized word embeddings (Peters et al., 2018; Conneau et al., 2017). From these sentence representations, labels from known utterances can be propagated mainly using clustering strategies.

[2]The pseudo-label confidence value is the alpha value calculated in section 3.3.

The first and most simple available method is the naive nearest neighbor algorithm – henceforth `nKNN` – that is for each unlabeled data point and each class, we compute its average similarity to the labeled samples of this class. Then, we assign to this point the label for which this average similarity is maximal, mapping all unlabeled data points to a pseudo-label.

The methods described so far are two-fold: unsupervised pseudo-labeling is followed by classification semi-supervised learning. Conversely, it is possible to learn end-to-end few shot classifiers, that are classifiers learning to classify using only few shots, without pseudo-labels. The state-of-the-art for end-to-end few-shot classification (Ren et al., 2018b) is an extension of Prototypical Networks (Snell et al., 2017). While originally designed for computer vision (Fort, 2017), such networks have recently gained traction in the field of few-shot setups in Natural Language Processing (Gao et al., 2019). Even though these methods are not meant to produce pseudo-labels directly, they excel at classifying utterances using only a few labeled samples. In this method, for each label, a prototype is built as the average embedding of support samples. Then, during training, query samples are classified using their distance to prototypes of each label. The optimizer's goal is to make this distance the lowest possible. It is therefore interesting to consider as related and as a later baseline, a system that is based on Prototypical Networks (Fort, 2017) where the input is the BERT (Devlin et al., 2018) sentence encoder. This approach will hereafter be referred to as `Proto`.

In our experiments (Section 5), we also include the state-of-the-art result for end-to-end few shot classification introduced by Ren (2018b). In this variant, unlabeled samples are used during an additional step, where the prototypes are refined using a soft K-Means technique. In our experiments, this baseline will be denoted as `Proto++`.

3 Two-fold Pseudo-Labeling

In this section, we will describe our folding/unfolding pseudo-labeling method (Section 3.1), as well as the aggregated approach (Section 3.2). We also introduce a loss weighting scheme (Section 3.3), which will be able to mitigate the impact of incorrect pseudo-labels on the intent detection downstream task. We advocate that this solution is state-of-the-art (Section 5) yet simple, therefore very valuable.

3.1 Folding/unfolding algorithm

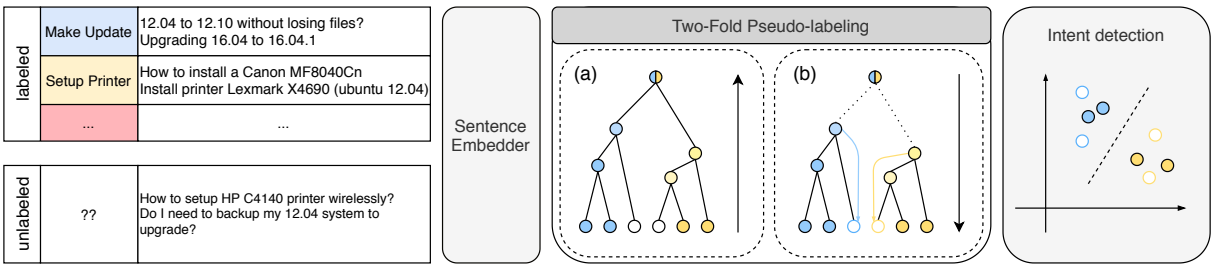

Figure 1: The two-fold Pseudo-Labeling process taking as an input embedded user utterances (Section 4.3). First, a hierarchical clustering method is applied from bottom to top, leading to a tree structure (*(a) – folding*). Second, from top to bottom, nodes are expanded iff they contain multiple labeled sentences with different labels ((b) – *unfolding*). Finally, retrieved pseudo-labels are used to train an intent detection model (Section 4.4).

Our pseudo-labeling method, illustrated in Figure 1, is a folding/unfolding algorithm, using hierarchical clustering. We decompose our approach in two main steps.

Step (a) – folding We start with all data points $X = X_l \cup X_u$, mixing together both labeled data points X_l as well as unlabeled data points X_u. At the beginning, each of those data points is considered as an individual cluster. Iteratively, we construct a tree in a hierarchical way: at each iteration, we select the two closest clusters, and bind them together into a new cluster. To choose which clusters to merge, we use Ward's method (Ward Jr, 1963). If we have N data points at the beginning, this process will take

$N - 1$ steps. In the end, this step produces a tree structure, where each non-leaf node has exactly two children, representing a merge between two nodes at some iteration.

Step (b) – unfolding After doing this bottom-to-top process, we now do the opposite. The step (a) yielded one tree structure containing all data points. Each node of this tree represents a merge of two clusters. Iteratively, we unfold the whole structure, by splitting clusters given a specific condition. More precisely, each cluster will be split into its two sub-clusters (which were merged at some point in step (a)) until all generated clusters contain either no labeled data points (in this case all its data points, which are unlabeled, are discarded), or some labeled data points with a unique label. In the latter case, we assign this unique label to all the unlabeled data points also belonging to the same cluster. This step is quite intuitive: if a cluster contains labeled data points with different labels, then it is not easy to choose which of those label will be assigned to unlabeled data points in this cluster. On the opposite, splitting clusters until only one label is represented in each cluster improves the confidence when assigning the pseudo-label to previously unlabeled data points.

There are several underlying motivations for this method. First, unlike most techniques, this approach does not require any hyper-parameter tuning, making it more robust to different datasets. In practice, when there is not much data to validate methods – this is the case in few-shot learning –, this is very important. Secondly, thanks to its iterative process, it is able to model unusual cluster shapes, which makes it more robust to sentence representations. Thirdly, the tree structure obtained via hierarchical clustering is richer than a more traditional cluster output. Indeed, it helps to understand the clustering process and how data points are linked to each other better. Lastly, unlike the `nKNN` or Prototypical-based approaches, this method does not give pseudo-labels for all the unlabeled data points. In a real-world scenario, this is very important, as we cannot ensure that all unlabeled data points – representing raw user utterances – belong to a known class. It is also better to consider not labeling a data point that is very hard to pseudo-label to avoid noise in the training data.

3.2 Aggregated pseudo-labeling approach

In our experiments, we also use an aggregated method using common pseudo-labels obtained from each of our single methods. Indeed, all pseudo-labeling schemes introduced earlier have a unique way to represent data and assign pseudo-labels. Following the unanimity vote for each given data point, we introduce an aggregated pseudo-labeling approach. To do so, after retrieving pseudo-labels from the various methods, we retain data points where all methods have assigned the same pseudo-label, and discard the rest. In order to measure the impact of each single method in the aggregation step, we will conduct additional ablation experiments, where we aggregate common predictions of all methods except one. By doing so for all methods, we will be able to measure the contribution of each method to the aggregation (Section 5.1).

3.3 Loss weighting mechanism

We will later demonstrate empirically that pseudo-labeling methods are able to recover most labels in one/few-shot configurations when the number of classes is relatively small (Section 5). When the number of labels is of one order of magnitude higher, generating correct pseudo-labels while refraining to introduce incorrect pseudo-labels alongside becomes harder. As the number of intent classes grows, the effect of providing correct labels issued by semi supervision is vastly reduced by incorrect pseudo-labels – even if they are provided in a lower amount than correct ones as it confuses the intent detection model and make the optimization task at hand unnecessarily harder/incorrect. In order to overcome this issue, we introduce a confidence score in our algorithms so that the loss function used to train the intent detection model penalizes mispredictions on pseudo-labels w.r.t. the confidence in the generated pseudo-label. More formally, given the similarity matrix S, an unlabeled data point $x_u \in X_u$ with pseudo-label $\hat{y}$, for each class c, we compute its logits $z_{u,c}$ as follows:

$$z_{u,c} = \frac{1}{n_{l,c}} \sum_{i=1}^{n_l} \mathbb{1}(y_i = 1) S_{i,u} \tag{1}$$

Then, using a softmax, we turn this z vector into a probability vector $P(z)$ (that is $P(z) = \sigma(z)$, where $\sigma(\cdot)$ is the softmax function). From here, the last steps are inspired from (Liu et al., 2018). We apply a final transformation in order to increase the differences between logits. The reason behind this is that all logits, being an average of similarity scores, are bounded between 0 and 1. Hence, if we just apply the softmax on those raw logits, the differences between classes will not be very significant. As a consequence, to increase sparsity and enforce extreme weights, we define the pseudo-label weight α as follows (we set the variable $\tau = 10$ in the experiments as in (Liu et al., 2018), and $\hat{y}$ still denotes the pseudo-label):

$$\alpha = \frac{\exp(\tau z_{u,\hat{y}})}{\sum_{j \neq \hat{y}} \exp(\tau z_{u,j})} \tag{2}$$

We will further discuss the impacts of this weighting technique in Section 5.2.

4 Experimental setup

Alongside with our contribution, the systems under evaluation are `nKNN`, `Proto/Proto++`, as introduced in Section 2. We implemented the `nKNN` baseline, and for the Prototypical-based approaches, we use the code associated to the seminal Prototypical Network paper[3], using BERT (Devlin et al., 2018) as our sentence encoder. For the `Proto++` baseline, to refine prototypes, we use 20 unlabeled samples per class both at training and testing time, as Ren (2018b) showed that it yielded better performances. As suggested by Chen (2019), we do not fine-tune Prototypical Networks at test time, as it is reported to decrease performances. Moreover, since Prototypical Networks are an end-to-end approach, it is first impossible to fine-tune such networks using a single shot for each class, and second, the embeddings of utterances for all labels are fine-tuned in the networks themselves.

The evaluation of the different systems is two-fold: we first assess the relevance of pseudo-labels retrieved from these methods in different few-shot situations, and then we evaluate the performances of the intent detection model (see Section 4.4) trained using those pseudo-labels alongside with the few labeled data. In order to measure the various performances, we use the weighted F1-score in all our experiments.

4.1 Few-shot and Cross-validation setup

In Few-Shot Learning, datasets are split into training and testing sets (Sung et al., 2017), where there are no overlapping classes between the two sets. Since few-shot regimes are highly dependent on the random initialisation of given shots, it is standard to repeat this process many times to create as many training/evaluation tasks and report average accuracy values. More formally, for each training task, we sample C classes from the training classes C_{train}. For each class $c \in C$, we sample K support samples $S_c = \{(x_i, y_i = c)\}_{i=1}^{K}$, as well as L query samples $Q_c = \{(x_i, y_i = c)\}_{i=1}^{L}$. Those samples will be used to train the model for this task. This setup is often named C-way K-shot settings. Test tasks are constructed in the same way, using the test set. At test time, we will evaluate the quality of our utterance encoder on those new classes C_{test}, disjoint from classes the model trained on. In our experiments, for each dataset, we use a third of classes for each training, validation, and testing stages. We also vary the number of shots K, ranging from 1 to 20, in order to assess the robustness of the different systems.

In order to get a sense of consistency in the results, each reported metric is averaged over 10 different cross-validation runs. For each run, the whole training set is used to train the pseudo-labeling model. Only 90% of the test set is selected to assign pseudo-labels to query samples using support samples. The remaining 10% of the test set is used to evaluate the performance of the text classification system which is trained on top of those pseudo-labels.

4.2 Datasets

To validate the effectiveness of our method, we explore 3 intent detection datasets. Additionally, because our method is not specific to intent detection, we also consider a text classification dataset. We summarize

[3]https://github.com/renmengye/few-shot-ssl-public

	#sentences	#classes (train/valid/test)	#sentences/class	#tokens/sentence
Snips	$14,484$	$2/2/3$	$2,069 \pm 21$	9.0 ± 3.2
OOS	$23,700$	$53/52/46$	157 ± 85	8.5 ± 3.3
Liu	$25,478$	$19/18/17$	472 ± 823	7.5 ± 3.4
R8	$7,685$	$3/2/3$	961 ± 1303	102 ± 117

Table 1: Datasets main statistics. The last two columns are the mean values and their standard deviation.

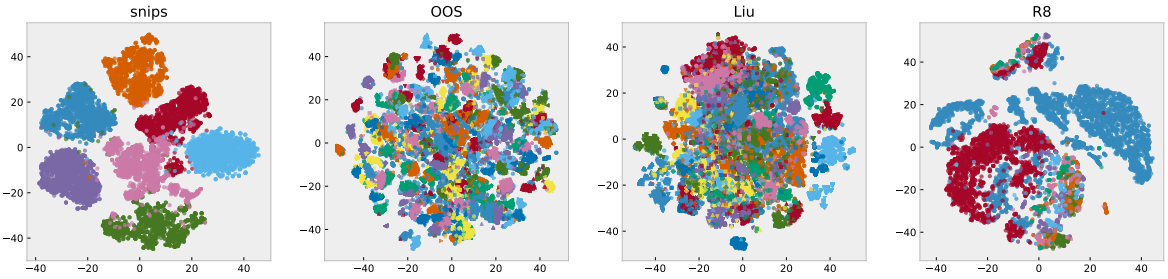

Figure 2: T-SNE visualization of `Snips`, `OOS`, `Liu`, and `R8` datasets. Each color/marker pair correspond to a given label. Utterance representations are derived from a BERT model, fine-tuned on the Masked Language Modeling task for each dataset.

the main statistics of the different datasets in Table 1. In order to visualize the data, after fine-tuning BERT models on each domain, we compute T-SNE embedding of utterances in Figure 2.

Snips This intent detection dataset is composed of 7 evenly distributed intent classes (Coucke et al., 2018). Class labels represent different voice commands, such as `BookRestaurant`, or `PlayMusic`. This dataset is the biggest one: it contains more than 2,000 sentences for each label, which is not often the case in practice.

OOS The OOS dataset, introduced by Larson (2019), was created to assess both intent detection and out-of-scope prediction models. In our experiments, the `out-of-scope` class – which has way more utterances as the others – is considered as any other. The dataset is recent, and the high number of classes makes the task challenging.

Liu Introduced by Liu (2019), this dataset is composed of user utterances collected on Amazon Mechanical Turk. It is used to assess the quality of various Natural Language Understanding Services, on both intent detection as well as entity recognition tasks. In our experiments, this dataset is one of the most unbalanced, ranging from 24 to $5,920$ samples per class.

R8 This dataset, mainly used for the more general case of text classification, contains Reuters Newswire articles. Following the work of authors who previously worked on this dataset (Zhao et al., 2018), we used the 8 most frequent classes. This process ensures that we have enough samples per class, and still yields a very imbalanced dataset. Unlike in the intent detection datasets, utterances in this text classification dataset are quite long – more than 10 times longer than the others, on average.

4.3 Sentence Encoder

For each dataset, we fine-tune the `bert-base` model (Wolf et al., 2019) on its former task, masked language modeling. Because this is an unsupervised task, we are able to use both labeled and unlabeled utterances during training. As shown by Sun (2019), this additional fine-tuning greatly increases the quality of embeddings. This fine-tuned model is then used as a checkpoint upon which prototypical-based approaches are built. Additionally, because prototypical networks benefit from a classification training step, it would be unfair to prevent the other approaches – `nKNN`, `Fold/Unfold` – from benefiting from such a fine-tuning. To account for this difference, we further fine-tune the BERT model obtained earlier on the classification task, using only few shots – `BERT-Fit`. The encoding part of this

model is then used as input for both nKNN and Fold/Unfold methods. This ensures that all methods benefit from a classification fine-tuning, only using the few shots which they are given.

4.4 Intent Detection model

Given the recent advances in NLP using transfer learning, we tried several methods for the intent detection model. We used pre-trained ELMo (Peters et al., 2018), InferSent (Conneau et al., 2017), FastText (Bojanowski et al., 2017) and BERT (Devlin et al., 2018) models. For ELMo (resp. BERT), we use the last hidden vector of the last (resp. first) token as the sentence embedding. All language models are fine-tuned with a last classification layer. We compare those methods on the intent detection task using a 10-fold validation, but do not report the results here due to space limitations. As in (Sun et al., 2019), we tried several variants for the BERT model – fine-tuning the model in the Masked Language Modeling (MLM) task as well as freezing or not freezing the transformer part. We find similar results as this paper: the variant which works best is obtained when fine-tuning on the MLM task, then further fine-tuning on the classification task, without freezing any layer. Hence, we chose this version as our intent detection model, in all our experiments.

5 Results and analysis

In Table 2, we report all results from pseudo-labeling as well as intent detection performance score. In order to measure the quality of pseudo-labels, we compute their F1-score. For the intent detection part, we report the F1-score on the held-out 10% of the dataset. Intuitively, aggregating pseudo-labels from different methods will discard a lot of pseudo-labels. Using 5 shots, aggregating pseudo-labels yields a discard rate of 10.2%, 25.1%, 47.8%, 45.5% for snips, OOS, Liu and R8 datasets respectively. To measure this trade-off between recall and precision of pseudo-labels, we will further discuss the impacts on the intent detection performances for each dataset.

Snips On the Snips dataset, even though our Folding/Unfolding approach outperforms all competitors, those results must be taken lightly. As we can see, with only 5 shots, we are able to recover pseudo-labels and build an intent detection model with a 0.988 F1-score. Those huge performances clearly hint that this dataset is not challenging at all: the t-SNE projection for this dataset in Figure 2 supports this claim. This means that we can hardly make concluding remarks on how the systems relatively perform on more realistic datasets. Overall, the massive usage of SNIPS to evaluate intent detection techniques in the literature (Xia et al., 2018; Goo et al., 2018) becomes open to question.

OOS On the OOS dataset, our method largely comes out on top, both on pseudo-label and intent detection results. On the intent detection part, our method is better than the aggregated ones, showing that other methods penalize the aggregation. This finding is very important, as it proves that using an ensemble of various methods do not always lead to better performances. Concerning the prototypical baselines, this is the dataset where the Proto++ is the further away from its former version, Proto. This comes from the fact that this dataset has the highest number of classes: when doing a soft-KMeans step, if the number of classes is high, then it is as much noise for the prototype refinement. Results on this dataset prove that our method is not handicapped when using a large number of classes, making it more robust to real-life scenarios. Additionally, we see that removing the BERT-Fit+nKNN from the aggregation yields better results. This poor performance was not to be seen in the pseudo-label results, as this baseline was performing second behind our approach. This raises the importance of using this double experiment, and confirms that better pseudo-labels do not directly imply a better model in the downstream task of intent detection.

Liu As explained in Section 4.2, this dataset is very imbalanced. Still, we get the same results as the two previous datasets, which were more balanced: our method is able to come out on top on both pseudo-labeling as well as intent detection results. Additionally, the ablation study where we withdraw our method from the aggregation yields the worst results in almost every situation, hinting that we contribute the most to this aggregation. This shows that our method is robust to various situations, including the case of imbalanced datasets.

		Pseudo-label F1			Intent Detection F1		
	shots	1	2	5	1	2	5
Snips	Proto	0.914	0.915	0.948	0.913	0.921	0.947
	Proto++	0.862	0.898	0.920	0.863	0.899	0.920
	BERT-Fit+nKNN	0.943	0.958	0.985	0.945	0.957	0.985
	BERT-Fit+Fold/Unfold	**0.986**	**0.982**	**0.992**	**0.988***	**0.983**	0.992
	BERT-Fit+Fold/Unfold+weights	-	-	-	0.987	0.983	**0.992**
	aggregated	**0.994**	**0.996**	**0.997**	0.983	0.983	0.992
	$\text{agg}_{-\text{BERT-Fit+nKNN}}$	0.994	0.996	0.997	**0.985**	**0.986**	**0.992**
	$\text{agg}_{-\text{Proto}}$	0.993	0.988	0.996	0.984	0.980	0.991
	$\text{agg}_{-\text{Proto++}}$	0.991	0.995	0.995	0.984	0.982	0.991
	$\text{agg}_{-\text{BERT-Fit+Fold/Unfold}}$	0.988	0.993	0.995	0.973	0.978	0.988
	shots-only	-	-	-	0.948	0.965	0.987
OOS	Proto	0.644	0.738	0.807	0.648	0.749	0.829
	Proto++	0.415	0.512	0.622	0.416	0.520	0.640
	BERT-Fit+nKNN	0.723	0.831	0.895	0.725	0.837	0.902
	BERT-Fit+Fold/Unfold	**0.883**	**0.930***	**0.939**	0.878	0.932	**0.945***
	BERT-Fit+Fold/Unfold+weights	-	-	-	**0.880***	**0.932***	0.944
	aggregated	**0.890**	**0.921**	**0.940**	0.839	0.890	0.927
	$\text{agg}_{-\text{BERT-Fit+nKNN}}$	0.880	0.912	0.933	**0.851**	**0.908**	**0.940**
	$\text{agg}_{-\text{Proto}}$	0.856	0.899	0.925	0.848	0.890	0.929
	$\text{agg}_{-\text{Proto++}}$	0.856	0.903	0.928	0.849	0.887	0.928
	$\text{agg}_{-\text{BERT-Fit+Fold/Unfold}}$	0.844	0.896	0.931	0.784	0.859	0.913
	shots-only	-	-	-	0.714	0.830	0.901
Liu	Proto	0.513	0.618	0.697	0.515	0.624	0.694
	Proto++	0.467	0.534	0.605	0.472	0.535	0.605
	BERT-Fit+nKNN	0.500	0.601	0.642	0.505	0.608	0.645
	BERT-Fit+Fold/Unfold	**0.587**	**0.668**	**0.763**	0.537	0.636	**0.728**
	BERT-Fit+Fold/Unfold+weights	-	-	-	**0.539**	**0.637**	0.726
	aggregated	**0.852**	**0.895**	0.902	0.585	0.686	0.747
	$\text{agg}_{-\text{BERT-Fit+nKNN}}$	0.829	0.887	**0.905**	**0.590**	**0.702**	**0.778**
	$\text{agg}_{-\text{Proto}}$	0.828	0.874	0.894	0.584	0.679	0.749
	$\text{agg}_{-\text{Proto++}}$	0.824	0.864	0.874	0.572	0.678	0.740
	$\text{agg}_{-\text{BERT-Fit+Fold/Unfold}}$	0.722	0.819	0.840	0.576	0.665	0.715
	shots-only	-	-	-	0.443	0.620	0.756
R8	Proto	0.575	0.629	0.709	0.598	0.650	0.725
	Proto++	0.556	0.620	0.647	0.566	0.607	0.628
	BERT-Fit+nKNN	0.737	0.789	0.890	0.758	0.799	0.885
	BERT-Fit+Fold/Unfold	**0.816**	**0.837**	**0.901**	0.817	**0.834***	0.901
	BERT-Fit+Fold/Unfold+weights	-	-	-	**0.817**	0.831	**0.902***
	aggregated	**0.890**	**0.921**	**0.940**	0.813	0.824	0.894
	$\text{agg}_{-\text{BERT-Fit+nKNN}}$	0.880	0.912	0.933	0.806	0.815	**0.900**
	$\text{agg}_{-\text{Proto}}$	0.856	0.899	0.925	**0.817**	**0.822**	0.900
	$\text{agg}_{-\text{Proto++}}$	0.856	0.903	0.928	0.808	0.822	0.893
	$\text{agg}_{-\text{BERT-Fit+Fold/Unfold}}$	0.844	0.896	0.931	0.767	0.790	0.895
	shots-only	-	-	-	0.738	0.780	0.892

Table 2: Evaluation of pseudo-labels for the **label recovery** and **intent detection** tasks. We separate results from single and aggregated methods, highlighting in **bold** the best method in each category. If a single method is better than the best aggregated one, it is highlighted with a wildcard (*). The $\text{agg}_{-\lambda}$ line represents the aggregation between all single methods **except method** λ. For the different aggregations, we underline the worst (i.e. stressing the method contributing the most to the aggregation).

R8 On the R8 dataset, even though we now deal with text classification and much longer sentences, we have similar results as on others intent detection datasets. Our method comes out on top, even achieving better performances than aggregated methods during the classification step. This shows that our Folding/Unfolding method is robust, and not only suited for intent detection tasks, but also for the more general subject of text classification.

5.1 Aggregated approach

On all datasets, aggregating pseudo-labels obtained from single methods always improves the pseudo-label quality, at the cost of retrieving less pseudo-labels. However, this increment is not fully reflected in the intent detection performance results: most of times, there is a single method which, when discarded from the aggregation, yields better text classification results. This is important to consider in practice, and to have in mind that better pseudo-labels does not necessarily imply better results for the intent detection downstream task.

Overall, in the ablation study where we withdraw each single method from the aggregation, the aggregation without our method is very often the worst. This additional study shows that our method contributes the most to the aggregation, as removing it is often the worst case.

5.2 Weighting mechanism

When training the intent detection model with pseudo-labels recovered by our method, we can use or ignore the confidence weights. The goal of those weights is to penalize pseudo-labels which might have been incorrectly assigned (i.e. with small confidence). We show that the weighting mechanism has a limited impact on the classification task. Remember that our method does not assign a pseudo-label to every unlabeled utterance: in step (b) 3.1, it can discard clusters which are far from one another. Through this, our method intrinsically already discards hard-to-predict utterances – we show that our framework is already optimal in this regard and therefore does not require a weighting hyperparameter, which makes it simpler to train.

5.3 Proto and Proto++

We discuss here the performances of prototypical baselines. In our experiments, we showed that the Proto++ variant performs less than its former version, Proto. On each dataset, we also found that fine-tuning BERT for the classification task tremendously improves the dataset separability – not reported here, due to space limitations. Using a BERT model only fine-tuned on the masked language modeling task directly for the Proto++ also introduces a lot of noise: because embeddings are not well separated, adding 20 unlabeled samples per class adds to much noise for the model to separate classes. This effect is amplified on OOS dataset, as the number of classes is high.

6 Conclusion

In this paper we introduced a new state-of-the-art, hierarchical clustering inspired method for pseudo-labeling intent detection datasets. This is of the utmost practical interest in order to help linguists to faster provide new intent detection model as the intent classes repository evolve with time. Our method, being hyper-parameter-free, is robust to various situations, even when the number of classes is high. Overall, it is straightforward to implement and still the best option available for pseudo-labeling user utterances in few-shot classification. Through an ablation study, we also show that our method is complementary to other baselines so that it contributes to aggregated approaches significantly. Moreover, we demonstrate that while Proto++ approach is known to out-perform its former version Proto on image classification, this does not apply for learning few-shot intent detection models. Finally, we showed that if we continue to evaluate intent detection using very easy – yet popular – datasets like Snips, we are ultimately not solving the actual downstream task in practice, so we encourage any intent detection researchers and practitioners to use it with caution.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

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

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. 2019. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loï c Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proc. of 25th EMNLP*, pages 670–680, Copenhagen, Denmark.

Alice Coucke, Alaa Saade, Adrien Ball, and Bluche et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Jacob Devlin, Ming - Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400.

Stanislav Fort. 2017. Gaussian prototypical networks for few-shot learning on omniglot. *preprint arXiv:1708.02735*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational AI. *CoRR*, abs/1809.08267.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6407–6414.

Chih-Wen Goo, Guang Gao, and et al. Hsu. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proc. of 14th NAACL, Volume 2*, pages 753–757.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proc. of 15th EACL, Volume 2*, pages 427–431. Association for Computational Linguistics, April.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *CoRR*, abs/1603.03827.

Bin Liu, Zhirong Wu, Han Hu, and Stephen Lin. 2018. Deep metric transfer for label propagation with limited annotated data. *arXiv preprint arXiv:1812.08781*.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. Benchmarking natural language understanding services for building conversational agents. *arXiv preprint arXiv:1903.05566*.

James H Martin and Daniel Jurafsky. 2009. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, Chapters 25-26*. Pearson/Prentice Hall Upper Saddle River.

Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In *Proc. of 8th CoNLL-2004 (HLT-NAACL 2004)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of 16th NAACL*.

Gary Ren, Xiaochuan Ni, Manish Malik, and Qifa Ke. 2018a. Conversational query understanding using sequence to sequence modeling. In *Proc. of 27th TheWebConf*, pages 1715–1724.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. 2018b. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. *CoRR*, abs/1703.05175.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2017. Learning to compare: Relation network for few-shot learning. *CoRR*, abs/1711.06025.

Julien Tissier, Christophe Gravier, and Amaury Habrard. 2017. Dict2vec : Learning word embeddings using lexical dictionaries. In *Proc. of 25th EMNLP*, pages 254–263.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638.

Ulrike von Luxburg. 2007. A tutorial on spectral clustering. *CoRR*, abs/0711.0189.

Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S Yu. 2018. Zero-shot user intent detection via capsule neural networks. *arXiv preprint arXiv:1809.00385*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of 33rd ACL*.

Taotao Zhao, Xiangfeng Luo, Wei Qin, Subin Huang, and Shaorong Xie. 2018. Topic detection model in a single-domain corpus inspired by the human memory cognitive process. *Concurrency and Computation: Practice and Experience*, 30:e4642, 08.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE TKDE*, 17(11):1529–1541, Nov.

Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. 2005. *Semi-supervised learning with graphs*. Ph.D. thesis, Carnegie Mellon University, language technologies institute.