EFFICIENT SEMI-SUPERVISED LEARNING FOR NATURAL LANGUAGE UNDERSTANDING BY OPTIMIZING DIVERSITY

Eunah Cho[1], *He Xie*[1], *John P. Lalor*[2*], *Varun Kumar*[1], *William M. Campbell*[1]

[1]Amazon Alexa AI Natural Language Understanding
[2]University of Notre Dame

ABSTRACT

Expanding new functionalities efficiently is an ongoing challenge for single-turn task-oriented dialogue systems. In this work, we explore *functionality-specific* semi-supervised learning via self-training. We consider methods that augment training data automatically from unlabeled data sets in a functionality-targeted manner. In addition, we examine multiple techniques for efficient selection of augmented utterances to reduce training time and increase diversity. First, we consider paraphrase detection methods that attempt to find utterance variants of labeled training data with good coverage. Second, we explore sub-modular optimization based on n-grams features for utterance selection. Experiments show that functionality-specific self-training is very effective for improving system performance. In addition, methods optimizing diversity can reduce training data in many cases to 50% with little impact on performance.

Index Terms— Dialog system, Data efficient learning, Paraphrase learning

1. INTRODUCTION

Single-turn task-oriented dialogue systems have become commonplace. These systems are based on spoken language understanding which takes speech input and processes it with natural language understanding (NLU) to produce intents, domains, and slots [1]. As an example, a dialog system may have the input "set an alarm for five p.m. tomorrow" and produce domain *alarm*, intent *set alarm*, and slot values of "five p.m." for *time* and "tomorrow" for *date*. In this work we define a functionality as one or two NLU intents with their associated slots, that our dialog system can group into one model capability (e.g. "SettingAlarm").

One of the ongoing challenges with production applications is to quickly design and add new functionalities. Building accurate NLU models for new functionalities requires collection and manual annotation of new data which is an expensive process. Semi-supervised learning (SSL), learning from both unlabeled and existing labeled data, potentially provides a low-cost yet efficient method to improve NLU models performance.

Maintaining training data so that it is relevant with current usage pattern as well as to achieve efficient training is another challenge in production applications. As new functionalities are being introduced, it is expected to have constant shift in usage pattern. At the same time, continuously adding data for each functionality increases complexity for training.

For this work, we focus on self-training for semi-supervised NLU that provides performance improvement for a specific functionality. In self-training, the system improves its performance by applying the current NLU model to unlabeled utterances, selecting functionality-relevant data, and then augmenting this data back into the system training set. As new system capabilities emerge, self-training can be focused on the functionality of interest and improve system performance quickly with no human annotation effort. In contrast, other semi-supervised methods may provide overall system performance improvements, but yield no performance improvements for a particular functionality. In the following discussion, we explore a method with diversity measures to sub-sample data in order to achieve efficient training with SSL and make the model agile to shifts in usage pattern.

Popular SSL models include self-training, mixture models, graph-based methods, co-training and multiview learning [2, 3, 4]. Particularly, in speech and language processing, self-training [5, 6, 7, 8] and methods that learn representations from implicit information are common [9, 10]. In [11], authors shared their insights on how evaluation of SSL approaches should be made when using production settings. Diversity in SSL was considered for text categorization tasks [12]. In this work, diversity is used to better exploit unlabeled data by using feature subspace classifiers. In our work, we use diversity as a criterion to decrease the amount of training data necessary for maintaining the NLU performance. The authors in [13] investigated six models for paraphrase embedding, varying in terms of expressiveness and complexity including deep averaging network [14] and LSTM networks [15]. Inspired by this work, we examined the six models for embedding and selected the best performing one for our experiments.

Prior approaches to self-training for NLU [16, 17] have

* Work done during an internship at Amazon Alexa

primarily focused on model performance. In this paper, we focus on optimizing both NLU model performance and augmented training data diversity. As the unlabeled data pool grows and functionality usage increases, the amount of functionality-specific data can grow arbitrarily. Much of the variation in this data is minimal and does not improve system performance. Thus, the challenge with functionality-specific SSL is to find the functionality in the unlabeled pool, minimizing false positives given limited training data.

In addition, incorporation of large amounts of training data can linearly increase training times with neural network and embedding-based techniques. To address these concerns, we optimize augmented training set size with a selection criterion. Efficiently learning new functionalities with less data is a common problem across many modern voice assistants. Methods for optimizing measures similar to diversity are seen in multiple areas including general deep learning, e.g. active learning in NLU [18, 19]. In order to ensure external validity of our experiments, we selected NLU functionalities with varying number of slots and intent combination.

Our contribution in this paper is two-fold. First, we show that *functionality-specific* semi-supervised learning substantially improves production system performance for new capabilities, reaching up to 25% in relative performance difference. Second, we explore different methods for optimizing the size of the augmented training data set. Using a paraphrase detection model, we propose a greedy algorithm for optimizing diversity in the augmented data set. In addition, we compare our proposed paraphrase model with sub-modular data selection method. We demonstrate that applying these techniques to the unlabeled training data yields substantially smaller training data sets (down to 50%) with comparable performance to unrestricted augmentation.

2. SEMI-SUPERVISED LEARNING FOR NLU

Our approach to SSL is shown in Figure 1. For the first step, we take a pool of unlabeled utterances and apply a functionality filter. In order to simulate the SSL process, we gathered live traffic utterances from 2017 January to 2018 June, and used these utterances as the pool for SSL utterances.

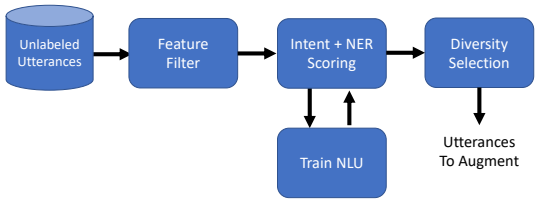

Fig. 1. SSL using self-training and diversity

The functionality filter is a 1-vs-rest classifier trained with in-class examples of the functionality and all other functionalities as out-of-class. A low complexity n-gram based linear logistic regression classifier is used for our implementation for high-throughput and to reduce volume. As features, we used unigram, bigram and trigram of the token sequences. Only utterances above a threshold are examined for subsequent stages. In this experiment, we used 0.5 as a fixed threshold.

For the next step in the figure, we iteratively refine our selection by applying both intent classification (IC) and named entity recognition (NER) to the filtered utterances. A fused system score is obtained by multiplying confidence score from each NLU component (e.g. IC) and used for selection of candidate utterances above a threshold. Given that the confidence score is normalized between 0 and 1, we experimented with a varying range of confidence score thresholds, $\{0.2, 0.3, 0.4, ..., 0.9\}$. The selected utterances are augmented back into training, where the data is weighted same as the threshold score. Thus, the more the model is confident, the more important the data is for training. In our preliminary experiments, we learned that we can achieve the best performance in SSL when we aggregate all data over different thresholds and add the aggregated data with a constant weight 1.0. Since highly confident utterances are included in the data set with a lower threshold, this data is already weighted towards more confident data. We will be using this aggregating method for SSL in this work.

As a final step, we take the results of the first two stages, namely the aggregated data, and apply a diversity selection process. Since the initial stages are confidence based, the output size can be arbitrary based on the size of the unlabeled data pool, the popularity and newness of the functionality, and the precision/recall of the filter. The diversity process ensures we obtain a parsimonious final set of utterances to augment into the final training process. Detailed description on the diversity based selection will be given in Section 4.4.

3. DIVERSITY IN SSL

Our intuition of the SSL process is that we can train the NLU model more efficiently by prioritizing utterances with diversity. Utterances with diversity will introduce lexical and syntactic variety into the training data. For example, let us assume that the model has already seen an utterance *play Adele* and thus can support the feature for the given carrier phrase. Compared to prioritizing the same or very similar utterances, prioritizing utterances with more variety (e.g. *I want to listen to Lady Gaga*) into training would help enhance the model to support the target feature.

Figure 2 depicts utterance embeddings for an Alexa functionality[1]. The figure shows that augmented data is scattered and fills in the gaps between annotated data. What we aim in

[1]For the figure, we randomly sampled 1K utterances from both annotated and augmented data. Sentence embeddings are obtained from 88.5 million utterances in production as described in [20]. For visualization we used t-SNE [21].

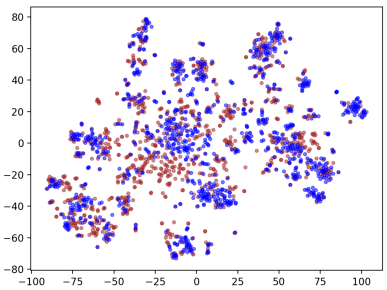

Fig. 2. Embeddings of functionality-specific utterances of different origin. Brown points are from human-annotated data, and blue points are from SSL augmented data.

this work is to prioritize augmentation samples given already existing training data, so that we can fill in the distribution more efficiently.

4. PARAPHRASE DETECTION

4.1. Paraphrase Embedding

To select our paraphrase embedding models, we experimented with the set of models proposed in [13]. In that work, six models were proposed, varying in terms of expressiveness and complexity. The first model only learns a word embedding matrix W_w, and sequences are embedded by averaging over the learned word embeddings:

$$g_{para-phrase}(x) = \frac{1}{n}\sum_{i}^{n} W_w^{x_i} \tag{1}$$

The second model adds a projection layer to the first model and learns a second weight matrix in addition.

$$g_{para-proj}(x) = W_c(\frac{1}{n}\sum_{i}^{n} W_w^{x_i}) + b_i \tag{2}$$

The third model tested is the Deep Averaging Network (DAN) of [14], which generalizes the second model above to variable depth as well as nonlinear activation functions. The fourth model is a standard RNN.

$$h_t = f(W_x W_w^{x_i} + W_h h_{t-1} + b) \tag{3}$$

$$g_{RNN}(x) = h_{-1} \tag{4}$$

The fifth proposed model is an identity-RNN proposed by the authors [13]. In an identity-RNN, the weights are all initialized to the identity matrix, and the activation function is the identity function. Before initialization, the identity-RNN reduces to the first model above and averages the word embeddings, but the expectation is that during learning the model learns semantic information to improve upon the word averaging baseline. The final model is an LSTM network [15].

Learning the parameters for each model involves minimizing an objective function with a margin [13].

$$\min_{W_c, W_w} \frac{1}{|X|}\Big(\sum_{\langle x_1, x_2\rangle \in X} \max(0, \delta - \cos(g(x_1), g(x_2)) \tag{5}$$

$$+ \cos(g(x_1), g(t_1))) + \max(0, \delta - \cos(g(x_1), g(x_2)) \tag{6}$$

$$+ \cos(g(x_2), g(t_2))) + \lambda_c ||W_c||^2 + \lambda_w ||W_{w_0} - W_w||^2\Big) \tag{7}$$

Where δ is the margin, $g()$ is the embedding function, γ_c and γ_w are regularization parameters, W_{w_0} is the initial word embedding matrix, and t_1 and t_2 are negative examples.

To construct negative pairs for training (t_1 and t_2), we selected the most similar non-paraphrase to an utterance, in paraphrase pairs x_1 and x_2 respectively, from a training mini-batch. For each model we learned paraphrase embeddings using the PPDB-S dataset [22], which includes 1.5 million paraphrase pairs. We trained each of the above models and evaluated them on the PPDB human-ranking task [22] and found that the word-averaging method performed best in our setup. Therefore, we selected the word averaging model as the embedding layer for our downstream paraphrase detection task.

4.2. Paraphrase Data

In this work, a group of utterances that share the same functionality in our dialog system are defined as paraphrase. In order to group such utterances, we rely on their NLU annotation.

We trained the paraphrase detection model on a set of internally-collected utterance pairs. In order to find and manually annotated paraphrase pairs, we look into the difference in time between the two utterances. If a request fails, users may immediately attempt to rephrase the original request so that the model can correctly process the request the second time around. As this fail-success pair may also include other errors (e.g. ASR error), our human annotators annotated whether it is indeed a paraphrase pair. This data consisted of 9.1K utterance pairs, with roughly one third of the pairs annotated as paraphrases.

Because the human annotated data is relatively small, we build another corpus using utterances in NLU training data, as shown in [23]. We collected utterance pairs if they had the same domain, intent, and number and types of slots in their NLU annotation. We then removed the slot entities so that we could compare the slot types, not specific slot entities. For example, we masked a paraphrase pair, as in *play Artist - I want to listen to Artist*. After the pairs were collected, we added back in slot entities so that they would match between the two utterances by randomly selecting entities from an internal catalog for each slot type. After this step, each utterance pair

was considered a positive example for the paraphrase detection task. To include generated negative examples into training, we randomly selected two utterances from our entire set of utterances. In addition to this, we added negative examples where carrier-phrases are the same but different entities are randomly sampled (e.g. *play Adele - play Ed Sheeran*). As a result, we added an additional 2.7 million examples to our training set.

4.3. Paraphrase Detection Model

We built a paraphrase model to output a score indicative of the likelihood that a pair of utterances are paraphrases. The model is defined as follows:

$$e_i = g(x_i) \tag{8}$$

$$h = [e_1, e_2, |e_1 - e_2|, e_1 \times e_2] \tag{9}$$

$$p(\text{para}(x_1, x_2)) = \sigma(h) \tag{10}$$

In this model, we embed both utterances using the selected paraphrase embedding model (Section 4.1). We then combine the embeddings to form an input representation by concatenating the embedding for each utterance, the element-wise difference between the two utterances, and the element-wise product between the two. This representation is then passed through a fully-connected neural network with a single output, representing the probability that the two utterances are paraphrases. We experimented with a number of hidden layers and activation functions and found that using two 100-dimension hidden layers with ReLU activation [24] gave the best performance.

4.4. Data Selection using Paraphrase Model

Our method to select utterances from the augmentation data we obtained from SSL is shown in Algorithm 1.

Algorithm 1 Greedy algorithm for diversity

Require: Annotated $A = (a_1, a_2, ..., a_m)$
Require: Augmented $U = (u_1, u_2, ..., u_l)$
Require: Batch size $s = |U| \times 5/100$
Require: Augmentation budget $k = p \times |U|, 0 < p \leq 1$
1: $S \leftarrow \emptyset$
2: **while** $|S| < k$ **do**
3: $\quad \mathcal{D} \leftarrow \emptyset$
4: $\quad$ **for** $u_i \in U$ **do**
5: $\quad\quad pr(u_i) \leftarrow \max_j \text{para}(u_i, a_j)$ for $a_j \in A$
6: $\quad$ **while** $|\mathcal{D}| < s$ **do**
7: $\quad\quad i^* = \text{argmin}_i pr(u_i)$
8: $\quad\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{u_{i^*}\}$
9: $\quad\quad U \leftarrow U - \{u_{i^*}\}$
10: $\quad A \leftarrow A \cup \mathcal{D}$
11: $\quad S \leftarrow S \cup \mathcal{D}$
12: use S for training

In order to select utterances from the augmented data U, we measured similarity between U and the annotated data set A. In steps 4 and 5, we find the best matching paraphrase for each utterance in U to utterances in A. In each iteration of selection, we choose 5% of the utterances in U, that are most dissimilar to A to prioritize diverse utterances for training. We split the augmentation budget into smaller batches to successively expand the support of the augmented data. For instance, after the first iteration, we augment data $\mathcal{D}$, from U that is least similar to A, into A. In subsequent iterations, we search for data to augment that is both dissimilar to the original A as well as the augmented $\mathcal{D}$.

In our preliminary experiment, we compared multiple sampling schemes. First, we considered prioritizing utterances for selection by using either the argmax or argmin operator in line 7 of the Algorithm 1. As expected, increasing diversity with argmin was most effective. Second, we evaluated how to handle duplicate utterances which can occur since both sets A and U are sampled from production traffic. In one approach we refrained from selecting duplicate utterances as long as there are other candidates in the batch. In another approach, we allowed duplicate occurrences while sampling. Results showed that the best performance was achieved when we sampled unique utterance first, prioritizing diversity.

5. BENCHMARKS SETUP

To simulate the effect of SSL as well as the efficiency improvement afforded by data selection using diversity, we established benchmarks. For each benchmark, we created different versions of the NLU model with varying amount of live training data on the target functionality in order to simulate the development cycle of it. Functionalities with varying complexity are chosen in order to better represent the diversity within our NLU model.

The baseline for each benchmark only includes synthetically created training data from FST for the target functionality. On top of that, 10%, 20%, 50%, 80% and 100% of the annotated live training data for target functionality is added to simulate the development cycle. In this paper, we will refer to this as *annotation increment*. On each annotation increment, SSL is carried out and a training data augmentation set is selected, as described in Section 2. Finally, paraphrase detection is employed to sub-select the SSL augmentation data.

Funct.	Domain	#Int.	#Slot	Annt.	Test
Announce	Comms.	1	17 (1)	4.4K	1.3K
Quotes	Info	1	12 (5)	4.1K	1.4K
Playlist	Music	2	32 (0)	5.2K	1.9K
Alarms	Music	2	51 (4)	10.6K	2.7K
Chat	General	1	1 (1)	1.3K	2.7K

Table 1. Data statistics for NLU functionalities.

Increment	System	Announce	Quotes	Playlist	Alarms	Chat
0%	Baseline	41.05	107.44	51.33	22.75	30.86
10%	Annotation	34.05	77.02	40.61	16.01	34.03
	+ SSL	**27.16**	**67.45**	**33.32**	**15.12**	**26.73**
20%	Annotation	31.78	70.02	37.89	14.28	34.72
	+ SSL	**24.75**	**59.98**	**32.38**	**13.37**	**28.78**
50%	Annotation	29.11	57.95	31.11	11.65	31.70
	+ SSL	**23.71**	**50.43**	**25.78**	**10.97**	**24.98**
80%	Annotation	28.08	50.70	26.92	11.20	31.19
	+ SSL	**24.85**	**43.40**	**23.38**	**10.62**	**24.54**
100%	Annotation	28.36	50.82	26.13	10.90	30.30
	+ SSL	**23.40**	**41.40**	**21.90**	**10.40**	**22.73**

Table 2. The impact of SSL on NLU Performance for five functionalities. Numbers are reported in SER.

6. EXPERIMENTAL SETUP

The NLU model we used for this experiment has three statistical models; domain classifier (DC), intent classifier (IC), and named entity classifier (NER).

For this experiment, we used maximum entropy (ME) classifiers for DC and IC and conditional random fields for NER modeling. We used n-grams extracted from training data as features for the models. A detailed description of our NLU system can be found in [25].

NLU performance is measured in Slot Error Rate (SER), as described in [26]. SER is obtained by comparing NLU hypothesis and reference and counting slot errors in substitution (S), insertion (I), and deletion (D). Intent error is treated as a slot substitution. We obtain SER as follows:

$$SER = \frac{S + I + D}{S + D + C} \tag{11}$$

where C denotes the number of correct slots/intent.

6.1. Functionalities and Augmentation for SSL

As described in Section 5, we experiment with augmentation data selection on five functionalities.

Table 1 summarizes characteristics of each functionality, the number of utterances of annotation data at 100% increment and test data. Characteristics of each functionality are represented by their domain, number of intents and slots that the functionality covers. We also show how many new slots (in parentheses) are introduced for named entity modeling by this new functionality. Each functionality covers one to two NLU intents in one NLU domain. We can also see that depending on the functionality, slot modeling complexity is drastically different. For example, it can be more challenging for models to learn a new functionality that does not bring any new slots than learning one with new additional slots, as the model needs to re-define and learn existing slots for a new functionality.

For each annotation increment, we halve the corresponding augmented data using our diversity techniques. Our goal is to halve the amount of augmentation data while achieving comparable NLU performance, compared to using all augmentation data.

Our model performance is not deterministic given retraining with the same training data, because of parallel computing and subsequent randomization. Thus, in this work we report averaged performance of five times of model training.

6.2. Comparative Systems

In SSL experiments, baseline uses no augmentation data for training. We will show how much NLU accuracy improvement can be obtained by applying the SSL technique.

In experiments for data selection using diversity, the baseline uses all augmentation data. As comparative systems, we explored four systems.

Experiments *random* (shown as rand) demonstrate whether we can achieve comparable NLU performance by randomly selecting half of the augmentation data for an efficient training.

We also explored data selection based on *sub-modular optimization* (shown as sub-m.). We use feature-based submodular optimization [27] to select a subset (with a budget 50% of the original set) from SSL augmentation data. In our experiments, we use 2-4 n-grams as features with 1.0 weight, tf-idf score as the modular function, and square root as the concave function. For submodular maximization, we use a lazy greedy algorithm [28, 29].

Finally, we aim to provide another comparative system where the augmentation data is *uniqued* (shown as uniq) and used for training. As the data size reduction of augmentation data would vary by functionality and by annotation increment, an exact comparison to 50% data selection is not feasible by simply taking unique utterances. For comparison, we match the same amount of data selection, either by selecting further utterances randomly out of the unique utterances, or randomly taking a subset of the unique utterances.

Table 3. SER score of five functionalities for each annotation increment using different selection schemes. The best selection mechanism is shown in bold letters.

Incr.	Select	Annc.	Quotes	Playlist	Alarms	Chat
10%	ALL	27.16	67.45	33.32	15.12	26.73
	Para	**28.14**	**67.10**	**33.82**	**14.91**	**27.59**
	Rand	29.25	68.07	34.56	15.45	28.72
	Sub-m.	28.95	68.04	34.44	15.05	28.35
	Uniq	28.96	67.42	34.29	15.00	27.90
20%	ALL	25.75	59.98	32.38	13.37	28.78
	Para	**26.59**	61.37	32.34	13.51	**28.84**
	Rand	27.84	61.35	33.37	13.55	29.87
	Sub-m.	27.05	**60.75**	32.11	**13.33**	29.62
	Uniq	26.92	60.97	**31.70**	13.57	29.22
50%	ALL	23.71	50.43	25.78	10.97	24.98
	Para	**24.40**	50.41	**25.38**	11.11	**26.16**
	Rand	24.99	51.62	26.18	11.17	27.57
	Sub-m.	24.69	50.43	25.76	**11.04**	26.56
	Uniq	24.64	**50.23**	25.63	11.19	27.06
80%	ALL	28.08	43.40	23.38	10.62	24.54
	Para	**24.52**	**44.00**	23.41	**10.61**	**25.31**
	Rand	24.86	44.87	24.13	10.74	26.58
	Sub-m.	24.85	44.25	**23.38**	10.61	25.72
	Uniq	25.35	44.03	23.42	10.63	25.79
100%	ALL	23.40	41.40	21.90	10.40	22.73
	Para	**23.41**	42.28	21.98	10.41	**22.80**
	Rand	24.12	43.05	22.47	**10.35**	24.20
	Sub-m.	23.86	42.61	22.17	10.43	23.57
	Uniq	23.51	**42.06**	**21.65**	10.48	23.01

7. RESULTS

7.1. SSL Augmentation Results

Table 2 shows the impact of the feature-specific SSL. The row of Increment 0% shows the NLU functionality performance in SER when there is no annotated data from live traffic used for training.

We show that as the increment increases, we were able to obtain more annotation data for the functionality, improving the NLU performance gradually. By finding the funtionality-relevant utterances in given live traffic utterances and applying the SSL approach as described in Section 2, we could consistently obtain further improvements SER.

Another noticeable thing is that we can achieve NLU performance of 100% annotation increment in a much earlier annotation increment, with SSL. For example, without using SSL, Announce functionality's SER goes down to 28.36 at the annotation increment 100%. We can easily achieve this level of performance by using SSL at the annotation increment 10%. For functionalities such as Quotes, Playlist, and Alarms, we can achieve the 100% annotation increments' NLU performance by using SSL at the annotation increment 50%. This shows that via SSL we can offer significantly better NLU performance to users, with significantly smaller manual annotation efforts. Overall, the results show that we can significantly improve NLU performance of the target functionality using automatic augmentation across all functionalities and annotation increments.

7.2. Data Selection for Diversity

Table 3 shows NLU performance for five functionalities given annotation increment and selection mechanisms. The best selection mechanism is shown in bold letters.

Numbers in the row of ALL represent NLU performance on the functionality test set when all augmentation data is used for the training, without any selection. In each row for each annotation increment, we show the performance of different selection methods (paraphrase, random, sub-modular and unique) which were applied to select half of the augmentation data. The numbers are reported in SER.

We performed a ranking based test to compare different methods to select the data. Details of the significance test can be found in [30, 31]. The test is applied for all functionalities and their all increments. When comparing four selection schemes, we learned that with 90% confidence the paraphrase detection model is better than any of the other selection methods.

We observe that selecting half of the augmentation data using paraphrase model does not cause a drastic performance drop compared to using all of the augmentation data—changes range from a 1.34% relative decrease to a 3.61% relative increase in SER. Further, a paired t-test [32] shows that for 7 out of 25 experiments shown in Table 3, training on all augmentation data is equivalent with 95% confidence to training on 50% augmentation data using paraphrase-based selection. Thus, we conclude that using greedy selection with paraphrase detection gives the best consistent performance when selecting half of the augmentation data for semi-supervised learning.

8. CONCLUSIONS AND FUTURE WORK

In this work, we showed that using functionality-specific SSL can greatly improve performance of production NLU systems. We explored multiple approaches for *efficient* SSL by optimizing diversity of augmentation data. Experiments showed that we can reach comparable performance by selecting half of the augmentation data based on diversity measures compared to using all augmentation data.

Future work includes applying diversity-measure-based utterance clustering and paraphrase-based strategies for utterance generation and augmentation. Also, we would like to investigate how the suggested SSL approach performs on the state-of-the-art approaches.

9. REFERENCES

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

[2] Xiaojin Zhu, "Semi-supervised learning literature survey," 2005.

[3] Xiaojin Zhu and Andrew B Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[4] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien, *Semi-Supervised Learning*, The MIT Press, 1st edition, 2010.

[5] Jeff Ma, Spyros Matsoukas, Owen Kimball, and Richard Schwartz, "Unsupervised training on large amounts of broadcast news data," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. IEEE, 2006, vol. 3, pp. III–III.

[6] Gokhan Tur, Dilek Hakkani-Tür, and Robert E Schapire, "Combining active and semi-supervised learning for spoken language understanding," *Speech Communication*, vol. 45, no. 2, pp. 171–186, 2005.

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

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

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

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

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

[12] Shoushan Li, Sophia Yat Mei Lee, Wei Gao, and Chu-Ren Huang, "Semi-supervised text categorization by considering sufficiency and diversity," in *Natural Language Processing and Chinese Computing*, pp. 105–115. Springer, 2013.

[13] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu, "Towards universal paraphrastic sentence embeddings," *International Conference on Learning Representations*, 2016.

[14] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III, "Deep unordered composition rivals syntactic methods for text classification," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, vol. 1, pp. 1681–1691.

[15] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] Wei-Lin Wu, Ru-Zhan Lu, Jian-Yong Duan, Hui Liu, Feng Gao, and Yu-Quan Chen, "Spoken language understanding using weakly supervised learning," *Computer Speech & Language*, vol. 24, no. 2, pp. 358–382, 2010.

[17] Pierre Gotab, Geraldine Damnati, Frederic Bechet, and Lionel Delphin-Poulat, "Online slu model adaptation with a partial oracle," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[18] Ozan Sener and Silvio Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[19] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar, "Deep active learning for named entity recognition," *arXiv preprint arXiv:1707.05928*, 2017.

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

[21] Laurens van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[22] Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch, "Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification," in

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2015, vol. 2, pp. 425–430.

[23] Eunah Cho, He Xie, and William M. Campbell, "Paraphrase generation for semi-supervised learning in NLU," in *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, Minneapolis, Minnesota, June 2019, p. 45–54, Association for Computational Linguistics.

[24] Vinod Nair and Geoffrey E Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

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

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

[27] Katrin Kirchhoff and Jeff Bilmes, "Submodularity for data selection in machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 131–141.

[28] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 420–429.

[29] Michel Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization techniques*, pp. 234–243. Springer, 1978.

[30] Milton Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[31] Peter Nemenyi, "Distribution-free multiple comparisons," in *Biometrics*. International Biometric SOC 1441 I ST, NW, Suite 700, Washington, DC 20005-2210, 1962, vol. 18, p. 263.

[32] Philipp Koehn, "Statistical significance tests for machine translation evaluation," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.