On the Limits of Learning to Actively Learn Semantic Representations

Omri Koshorek[1] **Gabriel Stanovsky**[2,4] **Yichu Zhou**[3]

Vivek Srikumar[3] **Jonathan Berant**[1,2]

[1]Tel-Aviv University, [2]Allen Institute for AI
[3]The University of Utah, [4]University of Washington
{omri.koshorek,joberant}@cs.tau.ac.il
gabis@allenai.org, {flyaway,svivek}@cs.utah.edu

Abstract

One of the goals of natural language understanding is to develop models that map sentences into meaning representations. However, training such models requires expensive annotation of complex structures, which hinders their adoption. Learning to actively-learn (LTAL) is a recent paradigm for reducing the amount of labeled data by learning a policy that selects which samples should be labeled. In this work, we examine LTAL for learning semantic representations, such as QA-SRL. We show that even an *oracle* policy that is allowed to pick examples that maximize performance on the test set (and constitutes an upper bound on the potential of LTAL), does not substantially improve performance compared to a *random* policy. We investigate factors that could explain this finding and show that a distinguishing characteristic of successful applications of LTAL is the interaction between optimization and the oracle policy selection process. In successful applications of LTAL, the examples selected by the oracle policy do not substantially depend on the optimization procedure, while in our setup the stochastic nature of optimization strongly affects the examples selected by the oracle. We conclude that the current applicability of LTAL for improving data efficiency in learning semantic meaning representations is limited.

1 Introduction

The task of mapping a natural language sentence into a *semantic representation*, that is, a structure that represents its meaning, is one of the core goals of natural language processing. This goal has led to the creation of many general-purpose formalisms for representing the structure of language, such as semantic role labeling (SRL; Palmer et al., 2005), semantic dependencies (SDP; Oepen et al., 2014), abstract meaning representation (AMR; Banarescu et al., 2013), universal conceptual cognitive annotation (UCCA; Abend and Rappoport, 2013), question-answer driven SRL (QA-SRL; He et al., 2015), and universal dependencies (Nivre et al., 2016), as well as domain-specific semantic representations for particular users in fields such as biology (Kim et al., 2009; Nédellec et al., 2013; Berant et al., 2014) and material science (Mysore et al., 2017; Kim et al., 2019).

Currently, the dominant paradigm for building models that predict such representations is supervised learning, which requires annotating thousands of sentences with their correct structured representation, usually by experts. This arduous data collection is the main bottleneck for building parsers for different users in new domains.

Past work has proposed directions for accelerating data collection and improving data efficiency through multi-task learning across different representations (Stanovsky and Dagan, 2018; Hershcovich et al., 2018), or having non-experts annotate sentences in natural language (He et al., 2015, 2016). One of the classic and natural solutions for reducing annotation costs is to use *active learning*, an iterative procedure for selecting unlabeled examples which are most likely to improve the performance of a model, and annotating them (Settles, 2009).

Recently, learning to actively-learn (LTAL) has been proposed (Fang et al., 2017; Bachman et al., 2017; Liu et al., 2018), where the procedure for selecting unlabeled examples is *trained* using methods from reinforcement and imitation learning. In recent work by Liu et al. (2018), given a labeled dataset from some domain, active learning is simulated on this dataset, and a policy is trained to iteratively select the subset of examples that maximizes performance on a development set. Then, this policy is used on a target domain to select unlabeled examples for annotation. If the learned policy generalizes well, we can reduce the cost of learning semantic representations. Liu et al. (2018) and Vu et al. (2019) have shown that such learned policies significantly reduce annotation costs on both text classification and named entity recognition (NER).

In this paper, we examine the potential of LTAL for learning a semantic representation such as QA-SRL. We propose an *oracle setup* that can be considered as an upper bound to what can be achieved with a learned policy. Specifically, we use an *oracle policy* that is allowed to always pick a subset of examples that maximizes its target metric on a development set, which has the same distribution as the *test set*. Surprisingly, we find that even this powerful oracle policy does not substantially improve performance compared to a policy that randomly selects unlabeled examples on two semantic tasks: QA-SRL span (argument) detection and QA-SRL question (role) generation.

To elucidate this surprising finding, we perform a thorough analysis, investigating various factors that could negatively affect the oracle policy selection process. We examine possible explanatory factors including: (a) the search strategy in the unlabeled data space (b) the procedure for training the QA-SRL model (c) the architecture of the model and (d) the greedy nature of the selection procedure. We find that for all factors, it is challenging to get consistent gains with an oracle policy over a random policy.

To further our understanding, we replicate the experiments of Liu et al. (2018) on NER, and compare the properties of a successful oracle policy in NER to the less successful case of QA-SRL. We find that optimization stochasticity negatively affects the process of sample selection in QA-SRL; different random seeds for the optimizer result in different selected samples. We propose a measure for quantifying this effect, which can be used to assess the potential of LTAL in new setups.

To conclude, in this work, we conduct a thorough empirical investigation of LTAL for learning a semantic representation, and find that it is difficult to substantially improve data efficiency compared to standard supervised learning. Thus, other approaches should be explored for the important goal of reducing annotation costs in building such models. Code for reproducing our experiments is available at `https://github.com/koomri/LTAL_SR/`.

2 Learning to Actively Learn

Classic pool-based active learning (Settles, 2009) assumes access to a small labeled dataset $\mathcal{S}_{\text{lab}}$ and a large pool of unlabeled examples $\mathcal{S}_{\text{unlab}}$ for a target task. In each iteration, a heuristic is used to select L unlabeled examples, which are sent to annotation and added to $\mathcal{S}_{\text{lab}}$. An example heuristic is *uncertainty sampling* (Lewis and Gale, 1994), which at each iteration chooses examples that the current model is the least confident about.

LTAL proposes to replace the heuristic with a learned policy π_θ, parameterized by θ. At *training time*, the policy is trained by simulating active learning on a labeled dataset and generating training data from the simulation. At *test time*, the policy is applied to select examples in a new domain. Figure 1 and Algorithm 1 describe this data collection procedure, on which we build our *oracle policy* (§3).

In LTAL, we assume a labeled dataset $\mathcal{D}$ which is partitioned into three disjoint sets: a small labeled set $\mathcal{S}_{\text{lab}}$, a large set $\mathcal{S}_{\text{unlab}}$ that will be treated as unlabeled, and an evaluation set $\mathcal{S}_{\text{eval}}$ that will be used to estimate the quality of models. Then, active learning is simulated for B iterations. In each iteration i, a model m_ϕ^i, parameterized by ϕ, is first trained on the labeled dataset. Then, K subsets $\{\mathcal{C}_j\}_{j=1}^K$ are randomly sampled from $\mathcal{S}_{\text{unlab}}$, and the model m_ϕ^i is fine-tuned on each candidate set, producing K models $\{m_{\phi_j}^i\}_{j=1}^K$. The performance of each model is evaluated on $\mathcal{S}_{\text{eval}}$, yielding the scores $\{s(\mathcal{C}_j)\}_{j=1}^K$. Let the candidate set with highest accuracy be $\mathcal{C}_t^i$. We can create training examples for π_θ, where $(\mathcal{S}_{\text{lab}}, \mathcal{S}_{\text{unlab}}, m_\phi^i, \{s(\mathcal{C}_j)\}_{j=1}^K)$ are the inputs and $\mathcal{C}_t^i$ is the label. Then $\mathcal{C}_t^i$ is moved from $\mathcal{S}_{\text{unlab}}$ to $\mathcal{S}_{\text{lab}}$.

Simulating active learning is a computationally expensive procedure. In each iteration we need to train K models over $\mathcal{S}_{\text{lab}} \cup \mathcal{C}_j$. However, a trained network can potentially lead to a policy that is better than standard active learning heuristics.

3 An Oracle Active Learning Policy

Our goal is to examine the potential of LTAL for learning a semantic representation such as QA-SRL. Towards this goal, we investigate an *oracle policy* that should be an upper bound for what can be achieved with a learned policy π_θ.

The oracle policy is allowed to use Algorithm 1

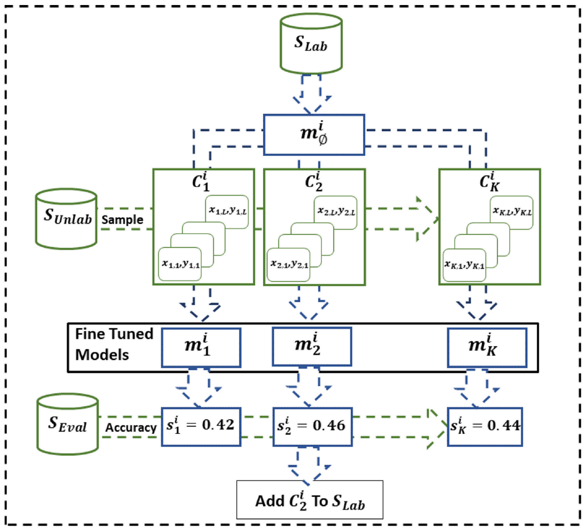

Figure 1: A single iteration of LTAL, where examples are sampled from $\mathcal{S}_{\text{unlab}}$, trained with examples in $\mathcal{S}_{\text{lab}}$, and performance on $\mathcal{S}_{\text{eval}}$ is used to select examples to annotate. See §2 for details.

Algorithm 1: Simulating active learning

```
   Input: $\mathcal{S}_{\text{lab}}, \mathcal{S}_{\text{unlab}}, \mathcal{S}_{\text{eval}}$
1  for $i \in \{1 \dots B\}$ do
2      $m^i_\phi \leftarrow$ Train($\mathcal{S}_{\text{lab}}$)
3      $\mathcal{C}^i_1, \dots, \mathcal{C}^i_K$ = SampleCandidates($\mathcal{S}_{\text{unlab}}$)
4      for $j \in \{1, \dots, K\}$ do
5          $m^i_{\phi_j} \leftarrow$ FineTune($m^i_\phi, \mathcal{S}_{\text{lab}} \cup \mathcal{C}^i_j$)
6          $s^i_j \leftarrow$ Accuracy($m^i_{\phi_j}, \mathcal{S}_{\text{eval}}$)
7      $t \leftarrow \operatorname{argmax}_{j \in \{1,\dots,K\}} s^i_j$
8      CreateTrainEx(($\mathcal{S}_{\text{lab}}, \mathcal{S}_{\text{unlab}}, m^i_\phi, \{\mathcal{C}^i_j\}^K_{j=1}$), $\mathcal{C}^i_t$)
9      $\mathcal{S}_{\text{lab}} \leftarrow \mathcal{S}_{\text{lab}} \cup \mathcal{C}^i_t$, $\mathcal{S}_{\text{unlab}} \leftarrow \mathcal{S}_{\text{unlab}} \setminus \mathcal{C}^i_t$
10 return $\mathcal{S}_{\text{lab}}$
```

at test time (it does not create training examples for π_θ, thus Line 8 is skipped). Put differently, the oracle policy selects the set of unlabeled examples that maximizes the target metric of our model on a set sampled from the *same distribution* as the test set. Therefore, the oracle policy enjoys extremely favorable conditions compared to a trained policy, and we expect it to provide an upper bound on the performance of π_θ. Despite these clear advantages, we will show that an oracle policy struggles to substantially improve performance compared to a random policy.

While the oracle policy effectively "peeks" at the label to make a decision, there are various factors that could explain the low performance of a model trained under the oracle policy. We now list several hypotheses, and in §5.4 and §6 methodologically examine whether they explain the empirical results of LTAL.

- **Training**: The models $m^i_{\phi_j}$ are affected by the training procedure in Lines 2 and 5 of Alg. 1. Different training procedures affect the performance of models trained with the oracle policy.
- **Search space coverage**: Training over all unlabeled examples in each iteration is intractable, so the oracle policy randomly samples K subsets, each with L examples. Because $K \cdot L << |\mathcal{S}_{\text{unlab}}|$, it is possible that randomly sampling these sets will miss the more beneficial unlabeled examples. Moreover, the parameter L controls the diversity of candidate subsets, since as L increases the similarity between the K different subsets grows. Thus, the hyper-parameters K and L might affect the outcome of the oracle policy.
- **Model architecture**: The model architecture (e.g., number of parameters) can affect the efficacy of learning under the oracle policy.
- **Stochasticity:** The oracle policy chooses an unlabeled set based on performance after training with stochastic gradient descent. Differences in performance between candidate sets might be related to this stochasticity, rather than to the actual value of the examples (especially when $\mathcal{S}_{\text{lab}}$ is small).
- **Myopicity:** The oracle policy chooses the set $\mathcal{C}^i_j$ that maximizes its performance. However, the success of LTAL depends on the *sequence* of choices that are made. It is possible that the greedy nature of this procedure results in sub-optimal performance. Unfortunately, improving search through beam search or similar measures is intractable in this already computationally-expensive procedure.

We now describe QA-SRL (He et al., 2015), which is the focus of our investigation, and then describe the experiments with the oracle policy.

4 QA-SRL Schema

QA-SRL was introduced by He et al. (2015) as an open variant of the predefined role schema in traditional SRL. QA-SRL replaces the predefined set of *roles* with the notion of *argument questions*. These are natural language questions centered around a target predicate, where the answers to the given question are its corresponding arguments. For example, for the sentence *"Elizabeth Warren decided to **run** for president"*, traditional SRL will label *"Elizabeth Warren"* as `ARG0` of the **run** predicate (the agent of the predicate, or the entity running in this case), while QA-SRL

Elizabeth Warren ***announced*** *her candidacy at a rally in Massachusetts.*

Argument	QA-SRL role	PropBank role
Elizabeth Warren	Who **announced** something?	`ARG0`
her candidacy	What did someone **announce**?	`ARG1`
at a rally in Massachusetts	Where did someone **announce** something?	`ARGM-LOC`

Table 1: Example of QA-SRL versus traditional SRL annotation for a given input sentence (top). Each line shows a single argument, and its role in QA-SRL (in question form) followed by its traditional SRL role, using PropBank notation. Roles in QA-SRL have a structured open representation, while SRL assigns discrete roles from a predefined set.

will assign the more subtle question *"who might run?"*, indicating the uncertainty of this future event. Questions are generated by assigning values to 7 pre-defined slots (where some of the slots are potentially empty). See Table 1 for an example QA-SRL annotation of a full sentence.

Recently, FitzGerald et al. (2018) demonstrated the scalability of QA-SRL by crowdsourcing the annotation of a large QA-SRL dataset, dubbed QA-SRL bank 2.0. It consists of 250K QA pairs over 64K sentences on three different domains (Wikipedia, news, and science). Following, this large dataset has enabled the development a neural model which breaks QA-SRL into a pipeline of two tasks, given a target predicate in an input sentence. First, a *span detection* algorithm identifies arguments of the predicate as continuous spans in the sentence (e.g., *"Elizabeth Warren"* in the previous example), then a *question generation* model predicts an appropriate role question (e.g., *"who might run?"*).

We find that QA-SRL is a good test-bed for active learning of semantic representations, for several key reasons: (1) it requires semantic understanding of the sentence, beyond syntactic or surface-level features (e.g., identifying the factuality of a given predicate), (2) adopting the formulation of FitzGerald et al. (2018), it consists of two semantic tasks, allowing us to test active learning on both of them, (3) we can leverage the large QA-SRL dataset to simulate active learning scenarios, and lastly (4) QA-SRL's scalability is attractive for the application of active learning policies, as they may further reduce costs for researchers working on developing specialized semantic representations in low-resource domains (e.g., medical, biological, or educational domains).

5 Experimental Evaluation

We now perform a series of experiments comparing the performance of an oracle policy to a random policy. We describe the experimental settings (§5.1), tasks and models (§5.2), present the main results (§5.3), and conclude by investigating factors that might affect our empirical findings (§5.4).

5.1 Experimental Settings

We evaluate the potential of the oracle policy on QA-SRL Bank 2.0 (FitzGerald et al., 2018). We use the training set of the *science* domain as $\mathcal{D}$, randomly split it into $\mathcal{S}_{\text{lab}}$, $\mathcal{S}_{\text{unlab}}$, and $\mathcal{S}_{\text{eval}}$. We evaluate the success of a model m^i_ϕ trained with the oracle policy by periodically measuring performance on the development set of the *science* domain. Unless mentioned, all results are an average of 3 experiments, where a different split of $\mathcal{D}$ was performed. Each experiment used K threads of a 40-core 2.2GHz Xeon Silver 4114 machine.

We compare the results of a base oracle policy (BASEORACLE) corresponding to the best policy we were able to obtain using the architecture from FitzGerald et al. (2018) to the following baselines:

- RANDOM: One of the candidate sets $\mathcal{C}^i_j$ is chosen at random and added to $\mathcal{S}_{\text{lab}}$.
- LONGEST: The set $\mathcal{C}^i_j$ with the maximal average number of tokens per sentence is added to $\mathcal{S}_{\text{lab}}$.
- UNCERTAINTY: For each candidate set, we use m^i_ϕ to perform predictions over all of the sentences in the set, and choose the set $\mathcal{C}^i_j$ that has the maximal average entropy over the set of predictions.

5.2 Tasks and Models

We now describe the three tasks and corresponding models in our analysis:

Span Detection: Here we detect spans that are arguments of a predicate in a sentence (see Table 1). We start with a labeled set of size $|\mathcal{S}_{\text{lab}}| = 50$, and select examples with the oracle policy for $B = 460$ iterations. We set the number of candidate sets to $K = 5$, and the size of each set to $L = 1$, thus the size of the final labeled set is 510 examples. We train the publicly available span detection model released by FitzGerald et al. (2018), which consumes as input a sentence $x_1, \ldots, x_n$, where x_i is the concatenation of the embedding of the ith word in the sentence and a learned embedding of a binary indicator for whether this word is the target predicate. This input is fed into a multi-layer encoder, producing a representation h_i for every token. Each span $x_{i:j}$ is represented by concatenating the respective hidden states: $s_{ij} = [h_i; h_j]$. A fully connected network consumes the span representation s_{ij}, and predicts a probability whether the span is an argument or not.

To accelerate training, we reduce the number of parameters to 488K by freezing the token embeddings, reducing the number of layers in the encoder, and by shrinking the dimension of both the hidden representations and the binary predicate indicator embedding. Following FitzGerald et al. (2018), we use GLoVe embeddings (Pennington et al., 2014).

Question Generation: We generate the question (role) for a given predicate and corresponding argument. We start with a labeled set of size $|\mathcal{S}_{\text{lab}}| = 500$ and perform $B = 250$ iterations, where in each iteration we sample $K = 5$ candidate sets each of size $L = 10$ (lower values were intractable). Thus, the final size of $\mathcal{S}_{\text{lab}}$ is 3,000 samples. We train the publicly available *local* question generation model from FitzGerald et al. (2018), where the learned argument representation s_{ij} is used to independently predict each of the 7 question slots. We reduce the number of parameters to 360K with the same modifications as in the span detector model. As a metric for the quality of question generation models, we use its official metric exact match (EM), which reflects the percentage of predicted questions that are identical to the ground truth questions.

Named Entity Recognition: To reproduce the experiments of Liu et al. (2018) we run the oracle policy on the CoNLL-2003 NER English dataset (Sang and De Meulder, 2003), replicating the experimental settings described in Liu et al. (2018) (as their code is not publicly available). We run the oracle policy for $B = 200$ iterations, starting from an empty $\mathcal{S}_{\text{lab}}$, and adding one example ($L = 1$) from $K = 5$ candidate sets in each iteration. We use a CRF sequence tagger from AllenNLP (Gardner et al., 2018), and experiment with two variants: (1) NER-MULTILANG: A BiLSTM CRF model (20K parameters) with 40 dimensional multi-lingual word embeddings (Ammar et al., 2016), and (2) NER-LINEAR: A linear CRF model which was originally used by Liu et al. (2018).

5.3 Results

Span Detection: Table 2 shows F_1 score (the official metric) of the QA-SRL span detector models for different sizes of $\mathcal{S}_{\text{lab}}$ for BASEORACLE and the other baselines. Figure 2 (left) shows the relative improvement of the baselines over RANDOM. We observe that the maximal improvement of BASEORACLE over RANDOM is 9% given 200 examples, but with larger $\mathcal{S}_{\text{lab}}$ the improvement drops to less than 5%. This is substantially less than the improvements obtained by Liu et al. (2018) on text classification and NER. Moreover, LONGEST outperforms BASEORACLE in most of the observed results. This shows that there exists a selection strategy that is better than BASEORACLE, but it is not the one chosen by the oracle policy.

Question Generation: To check whether the previous result is specific to span detection, we conduct the same experiment for question generation. However, training question generation models is slower compared to span detection and thus we explore a smaller space of hyper-parameters. Table 3 reports the EM scores achieved by BASEORACLE and LONGEST, and Figure 2 (center) shows the relative improvement. Here, the performance of BASEORACLE is even worse compared to span detection, as its maximal relative improvement over RANDOM is at most 5%.

Named Entity Recognition: Figure 2 (right) shows the relative improvement of NER-LINEAR and NER-MULTILANG compared to RANDOM. We observe that in NER-LINEAR, which is a replication of Liu et al. (2018), the oracle policy indeed obtains a large improvement over RANDOM for various sizes of $\mathcal{S}_{\text{lab}}$, with at least 9.5% relative improvement in performance. However, in NER-MULTILANG the relative gains are smaller, especially when the size of $\mathcal{S}_{\text{lab}}$ is small.

5.4 Extended Experiments

Surprisingly, we observed in §5.3 that even an oracle policy, which is allowed to pick the examples

# samples	100	150	200	250	300	350	400	450	500
BASEORACLE	42.7	**49.2**	52.9	54.2	**56.6**	**57.4**	58.4	**59.5**	59.9
RANDOM	42.8	47.2	48.3	52.4	53.3	56.1	57.0	57.5	58.5
LONGEST	**44.1**	49.1	**53.0**	**55.5**	56.4	**57.4**	**58.7**	58.6	**60.0**
UNCERTAINTY	42.8	47.0	50.1	51.3	52.2	54.4	55.1	55.6	56.9

Table 2: Span detection F_1 on the development set for all models across different numbers of labeled examples.

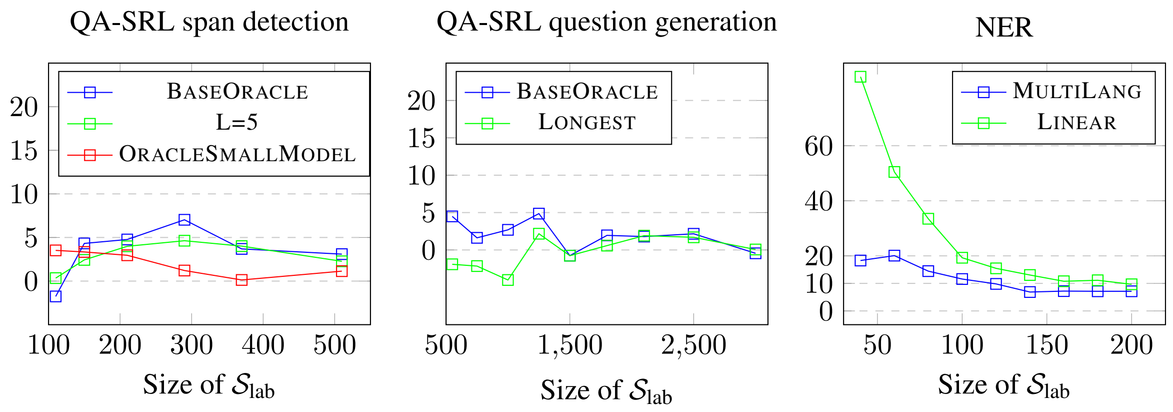

Figure 2: Relative improvement (in %) of different models compared to RANDOM on the development set. Note that the range of the y-axis in NER is different from QA-SRL.

that maximize performance on samples from the same distribution as the test set, does not substantially improve performance over a random policy. One possibility is that no active learning policy is better than random. However, LONGEST outperformed BASEORACLE showing that the problem is at least partially related to BASEORACLE itself.

We now examine the possible factors described in §3 and investigate their interaction with the performance of models trained with BASEORACLE. All modifications were tested on *span detection*, using the experimental settings described in §5.1.

Search space coverage We begin by examining the effect of the parameters K and L on the oracle policy. As K increases, we cover more of the unlabeled data, but training time increases linearly. As L increases, the subsets $\{\mathcal{C}_j\}_{j=1}^{K}$ become more similar to one another due to the fact that we are randomly mixing more examples from the unlabeled data. On the other hand, when L is small, the fine-tuning process is less affected by the candidate sets and more by $\mathcal{S}_{\text{lab}}$. In such case, it is likely that the difference in scores is also affected by stochasticity.

BASEORACLE uses $K = 5, L = 1$. We examine the performance of the oracle policy as these values are increased in Table 4. We observe that performance does not improve, and perhaps even decreases for larger values of K. We hypothesize that a large K increases the greediness of the procedure, and may result in selecting an example that seems promising in the current iteration but is suboptimal in the long run, similar to large beam sizes reducing performance in neural machine translation (Yang et al., 2018). A moderate K results in a more random and possibly beneficial selection.

Increasing the size of each candidate set to $L = 5$ or 20 results in roughly similar performance to $L = 1$. We hypothesize that there is a trade-off where as L increases the similarity between the different sets increases but training becomes more stable and vice versa, and thus performance for different L values does not vary substantially.

Training In Lines 2 and 5 of Alg. 1 we train on $\mathcal{S}_{\text{lab}}$ and then fine-tune on the union $\mathcal{S}_{\text{lab}} \cup \mathcal{C}_j^i$ until s_j^i does not significantly improve for 5 epochs. It is possible that fine-tuning from a fixed model reduces the efficacy of training, and training on $\mathcal{S}_{\text{lab}} \cup \mathcal{C}_j^i$ from random weights will improve performance. Of course, training from scratch will substantially increase training time. We run an experiment, termed INDEP., where Line 2 is skipped, and in Line 5 we independently train each of the candidate models from random weights. We find that this modification does not achieve better results than BASEORACLE, possibly because training a model from scratch for each of the candidates increases the stochasticity in the optimization.

# samples	550	750	1000	1250	1500	1800	2100	2500	3000
BaseOracle	**18.9**	**21.7**	**24.4**	**26.4**	27.1	**28.4**	**29.1**	**30.6**	31.1
Random	18.1	21.4	23.7	25.2	**27.3**	27.9	28.6	29.9	**31.3**
Longest	17.8	20.9	22.8	25.7	27.1	28.0	**29.1**	30.4	**31.3**

Table 3: Question generation scores (exact match) on the development set across different numbers of labeled examples.

# samples	110	150	210	290	370	510
Random	45.2	47.2	50.5	53.1	55.8	58.5
BaseOracle	43.3	**49.2**	**52.9**	**56.8**	57.8	**60.3**
$K = 10$	**46.6**	48.8	51.4	55.8	57.6	58.6
$K = 20$	44.8	47.4	52.1	55.9	—	—
$L = 5$	44.2	48.3	52.5	55.5	**58.0**	59.8
$L = 20$	45.2	47.5	51.9	55.1	56.7	58.7
Loss-Score	30.0	38.2	41.0	51.5	53.7	57.2
Indep.*	40.9	44.6	50.1	54.1	—	—
Epsilon-Greedy-0.3	45.1	48.6	52.4	55.6	57.1	59.8
Oracle-100	44.9	48.8	51.4	53.9	57.0	59.2
RandomSmallModel	51.9	54.8	57.3	59.5	61.4	62.6
OracleSmallModel	53.8	56.6	58.9	60.3	61.5	63.3

Table 4: Span detection F_1 scores on the development set for different size of $\mathcal{S}_{\text{lab}}$. We highlight the best performing policy for the standard span detector architecture. (*) indicates that the results are for a single run.

In addition, we also experiment with fine-tuning on $\mathcal{C}_j$ only, rather than $\mathcal{S}_{\text{lab}} \cup \mathcal{C}_j$. As we expect, results are quite poor since the model uses only a few examples for fine-tuning and forgets the examples in the labeled set.

Lastly, we hypothesize that selecting a candidate set based on the target metric (F_1 for span detection) might not be sensitive enough and thus we run an experiment, termed Loss-Score, where we select the set $\mathcal{C}_j$ that minimizes the loss on the development set. We find that this modification achieves lower results than Random, especially when $\mathcal{S}_{\text{lab}}$ is small, reflecting the fact that the loss is not perfectly correlated with our target metric.

Model Architecture In §5.3 we observed that results on NER vary with the model architecture. To see whether this phenomenon occurs also for span detection we perform a modification to the model – we reduce the number of parameters from 488K to 26K by reducing the hidden state size and replacing GLoVe embeddings with multi-lingual embeddings (Ammar et al., 2016). We then compare an oracle policy (OracleSmallModel) with a random policy (RandomSmallModel). Table 4 shows that while absolute F_1 actually improves in this setup, the oracle policy improves performance compared to a random policy by no more than 4%. Thus, contrary to NER, here architecture modifications do not expose an advantage of the oracle policy compared to the random one. We did not examine a simpler linear model for span detection, in light of recent findings (Lowell et al., 2019) that it is important to test LTAL with state-of-the-art models, as performance is tied to the specific model being trained.

Myopicity We hypothesized that greedily selecting an example that maximizes performance in a specific iteration might be suboptimal in the long run. Because non-greedy selection strategies are computationaly intractable, we perform the following two experiments.

First, we examine Epsilon-Greedy-p, where in each iteration the oracle policy selects the set $\mathcal{C}_j$ that maximizes target performance with probability $1 - p$ and randomly chooses a set with probability p. This is meant to check whether adding random exploration to the oracle policy might prevent it from getting stuck in local optima. We find that when $p = 0.3$ its performance is comparable to BaseOracle while reducing the computational costs.

Second, we observe that most of the gain of BaseOracle compared to Random is in the beginning of the procedure. Thus, we propose to use BaseOracle in the first b iterations, and then transition to a random policy (termed Oracle-b). We run this variation with $b = 100$ and find that it leads to similar performance.

To summarize, we have found that an oracle policy only slightly improves performance for QA-SRL span detection and question generation compared to a random policy, and that improvements in NER are also conditioned on the underlying model. Our results echo recent findings by Lowell et al. (2019), who have shown that gains achieved by active learning are small and inconsistent when modifying the model architecture.

We have examined multiple factors that might affect the performance of models trained with an oracle policy including the training procedure, model architecture, and search procedure, and have shown that in all of them the oracle policy struggles to improve over the random one. Thus, a *learned policy* is even less likely to obtain meaningful gains using LTAL.

In the next section we analyze the differences between NER-LINEAR, where LTAL works well, and BASEORACLE, in order to better understand the underlying causes for this phenomenon.

6 When does LTAL Work?

A basic underlying assumption of active learning (with or without a learned policy), is that some samples in $\mathcal{S}_{\text{unlab}}$ are more informative for the learning process than others. In LTAL, the informativeness of a candidate example set is defined by the accuracy of a trained model, as evaluated on $\mathcal{S}_{\text{eval}}$ (Line 6 in Alg. 1). Thus, for active learning to work, the candidate set that is selected should not be affected by the stochasticity of the training process. Put differently, the ranking of the candidate sets by the oracle policy should be *consistent* and not be dramatically affected by the optimization.

To operationalize this intuition, we use Alg. 1, but run the for-loop in Line 4 twice, using two different random seeds. Let $\mathcal{C}_t^i$ be the chosen or *reference* candidate set according to the first run of the for-loop in iteration i. We can measure the consistency of the optimization process by looking at the ranking of the candidate sets $\mathcal{C}_1^i, \dots \mathcal{C}_K^i$ according to the second fine-tuning, and computing the mean reciprocal rank (MRR) with respect to the reference candidate set $\mathcal{C}_t^i$ across all iterations:

$$\text{MRR} = \frac{1}{B}\sum_{i=1}^{B}\frac{1}{\text{rank}(\mathcal{C}_t^i)}, \tag{1}$$

where $\text{rank}(\mathcal{C}_t^i)$ is the rank of $\mathcal{C}_t^i$ in the second fine tuning step. The only difference between the two fine-tuning procedures is the random seed. Therefore, an MRR value that is close to 1 means that the ranking of the candidates is mostly affected by the quality of the samples, while a small MRR hints that optimization plays a large role. We prefer MRR to other correlation-based measures (such as Spearman's rank-order correlation), because the oracle is only affected by the candidate set that is ranked first. We can now examine whether the MRR score correlates with whether LTAL works or not.

We measure the MRR in 3 settings: (1) NER-LINEAR, a linear CRF model for NER which replicates the experimental settings in (Liu et al., 2018), where LTAL works, (2) NER-MULTILANG, a BiLSTM-CRF sequence tagger from AllenNLP (Gardner et al., 2018) with 40 dimensional multi-lingual word embeddings of Ammar et al. (2016), and (3) BASEORACLE, the baseline model for span detection task. In all experiments the initial $\mathcal{S}_{\text{lab}}$ was empty and B = 200, following the experimental settings in which LTAL has shown good performance (Liu et al., 2018; Fang et al., 2017; Vu et al., 2019). Since the MRR might change as the size of $\mathcal{S}_{\text{lab}}$ is increasing, we compute and report MRR every 10 iterations.

Figure 3 (left) presents the MRR in the three experiments. We observe that in NER-LINEAR the MRR has a stable value of 1, while in NER-MULTILANG and BASEORACLE the MRR value is substantially lower, and closer to an MRR value of a random selection (~.46). The right side of Figure 3 shows that NER-LINEAR oracle policy outperforms a random policy by a much larger margin, compared to the other 2 experiments.

These results show that the ranking in NER-LINEAR is not affected by the stochasticity of optimization, which is expected given its underlying convex loss function. On the other hand, the optimization process in the other experiments is over a non-convex loss function and a small $\mathcal{S}_{\text{lab}}$, and thus optimization is more brittle. Interestingly, we observe in Figure 3 that the gains of the oracle policy in NER-LINEAR are higher than NER-MULTILANG, although the task and the dataset are exactly same in the two experiments. This shows that the potential of LTAL is affected by the model, where a more complex model leads to smaller gains by LTAL.

We view our findings as a guideline for future work: by tracking the MRR one can assess the potential of LTAL at development time – when the MRR is small, the potential is limited.

7 Related Work

Active learning has shown promising results on various tasks. The commonly used *uncertainty* criteria (Lewis and Catlett, 1994; Culotta and McCallum, 2005) is focused on selecting the samples on which the confidence of the model is low. Among other notable approaches, in *query by committee* (Seung et al., 1992) a disagreement between a set of trained models on the prediction of an example is used to select what samples to label.

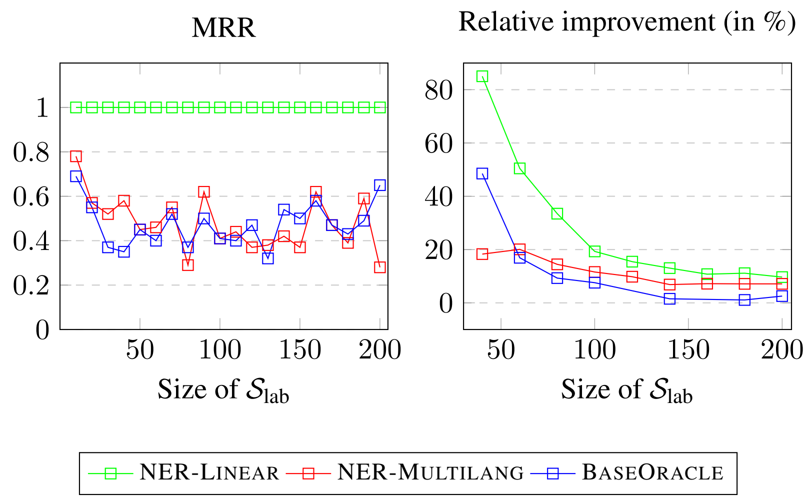

Figure 3: MRR (on the left) and relative improvement (in %) of different models compared to RANDOM on the development set.

In a large empirical study, Lowell et al. (2019) have recently shown other limitations in active learning. They investigate the performance of active learning across NLP tasks and model architectures, and demonstrate that it does not achieve consistent gains over supervised learning, mostly because the collected samples are beneficial to a specific model architecture, and does not yield better results than random selection when switching to a new architecture.

There has been little research regarding active learning of semantic representations. Among the relevant work, Siddhant and Lipton (2018) have shown that *uncertainty* estimation using dropout and Bayes-By-Backprop (Blundell et al., 2015) achieves good results on the SRL formulation. The improvements in performance due to LTAL approaches on various tasks (Konyushkova et al., 2017; Bachman et al., 2017; Fang et al., 2017; Liu et al., 2018) has raised the question whether learned policies can be applied also to the field of learning semantic representations.

8 Conclusions

We presented the first experimentation with LTAL techniques in learning parsers for semantic representations. Surprisingly, we find that LTAL, a learned method which was shown to be effective for NER and document classification, does not do significantly better than a random selection on two semantic representation tasks within the QA-SRL framework, even when given extremely favourable conditions. We thoroughly analyze the factors leading to this poor performance, and find that the stochasticity in the model optimization negatively affects the performance of LTAL. Finally, we propose a metric which can serve as an indicator for whether LTAL will fare well for a given dataset and model. Our results suggest that different approaches should be explored for the important task of building semantic representation models.

Acknowledgements

We thank Julian Michael and Oz Anani for their useful comments and feedback. This research was supported by The U.S-Israel Binational Science Foundation grant 2016257, its associated NSF grant 1737230 and The Yandex Initiative for Machine Learning.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 228–238.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Philip Bachman, Alessandro Sordoni, and Adam Trischler. 2017. Learning algorithms for active learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 301–310. JMLR. org.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin

Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. *EMNLP*.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-scale qa-srl parsing. *arXiv preprint arXiv:1805.05377*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 643–653.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proc. of ACL*, pages 373–385.

Edward Kim, Zach Jensen, Alexander van Grootel, Kevin Huang, Matthew Staib, Sheshera Mysore, Haw-Shiuan Chang, Emma Strubell, Andrew McCallum, Stefanie Jegelka, and Elsa Olivetti. 2019. Inorganic materials synthesis planning with literature-trained neural networks. *CoRR*, abs/1901.00032.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*. Association for Computational Linguistics.

Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. 2017. Learning active learning from data. In *Advances in Neural Information Processing Systems*, pages 4225–4235.

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

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR*.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1874–1883.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. 2019. Practical obstacles to deploying active learning. In *Proceedings of EMNLP*.

Sheshera Mysore, Edward Kim, Emma Strubell, Ao Liu, Haw-Shiuan Chang, Srikrishna Kompella, Kevin Huang, Andrew McCallum, and Elsa Olivetti. 2017. Automatically extracting action graphs from materials science synthesis procedures. *CoRR*, abs/1711.06872.

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Burr Settles. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM.

Aditya Siddhant and Zachary C Lipton. 2018. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. *arXiv preprint arXiv:1808.05697*.

Gabriel Stanovsky and Ido Dagan. 2018. Semantics as a foreign langauge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium. Association for Computational Linguistics.

Thuy Vu, Ming Liu, Dinh Phung, and Gholamreza Haffari. 2019. Learning how to active learn by dreaming. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4091–4101.

Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the beam search curse: A study of (re-) scoring methods and stopping criteria for neural machine translation. *EMNLP*.