Active Learning for Black-Box Semantic Role Labeling with Neural Factors

Chenguang Wang, Laura Chiticariu, Yunyao Li
IBM Research - Almaden
chenguang.wang@ibm.com, {chiti, yunyaoli}@us.ibm.com

Abstract

Active learning is a useful technique for tasks for which unlabeled data is abundant but manual labeling is expensive. One example of such a task is semantic role labeling (SRL), which relies heavily on labels from trained linguistic experts. One challenge in applying active learning algorithms for SRL is that the complete knowledge of the SRL model is often unavailable, against the common assumption that active learning methods are aware of the details of the underlying models. In this paper, we present an active learning framework for *black-box SRL models* (i.e., models whose details are unknown). In lieu of a query strategy based on model details, we propose a neural query strategy model that embeds both language and semantic information to automatically learn the query strategy from *predictions of an SRL model alone*. Our experimental results demonstrate the effectiveness of both this new active learning framework and the neural query strategy model.

1 Introduction

Active learning is a special case of semi-supervised machine learning in which a learning algorithm is able to interactively query the human to obtain the desired outputs at new data points. The goal of active learning is to carefully select the training data from which the model is being learnt in order to achieve good performance with less training data. A model (learner) starts with a small labeled set, then iteratively selects informative instances from unlabeled data based on a predefined query strategy and elicits labels for these instances; the new labeled instances are then added to the training set to train the model. Active learning is thus suitable for solving problems for which unlabeled data is abundant, but labels are expensive to obtain, such as parsing and speech recognition.

Semantic role labeling (SRL) [Gildea and Jurafsky, 2002] aims to recover the predicate-argument structure of an input sentence. Labeled data for training SRL models requires linguistic expertise and is time-consuming and labor-intensive to obtain, making active learning an attractive option. However, *query strategies* such as uncertainty sampling [Lewis and Gale, 1994; Scheffer *et al.*, 2001; Culotta and McCallum, 2005] and query-by-committee [Dagan and Engelson, 1995; Seung *et al.*, 1992], which are at the core of current active learning methods, require knowing the details of the underlying models. Unfortunately, the details of SRL models are often unavailable for the following two reasons.

High Complexity of SRL Model An SRL model typically contains four components: predicate identification and disambiguation, as well as argument identification and classification. Each component can be a different model with additional interplays with other components (e.g., logistic regression or neural network). The output of an SRL model contains four elements: predicate and frame label, argument and role label. For example, the output of a given sentence "Ms. Haag plays Elianti" would be the following: "plays" (predicate) and "play.02" (frame label), "Ms. Haag" (argument) and "A0" (role label)[1]. These characteristics make SRL models complex and difficult to understand.

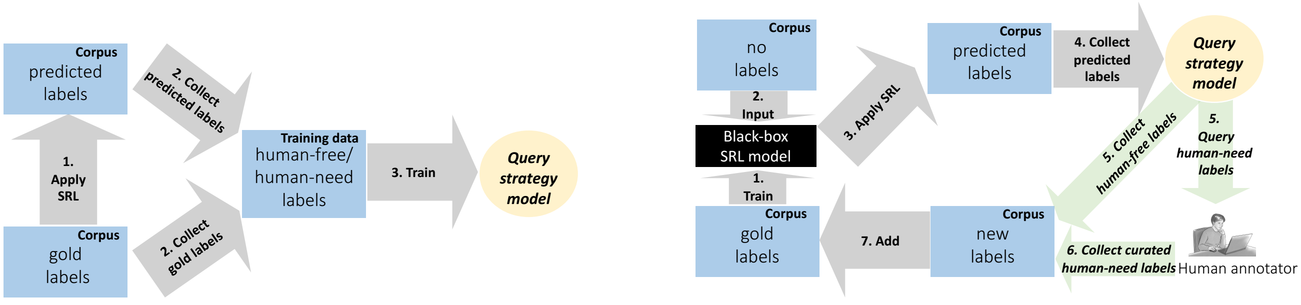
Low Accessibility of Most SRL Models Details Moreover the details of existing SRL models are often inaccessible. Many SRL models, such as the two used in our investigation (MATE[2] and CLEAR[3]) are provided in binary forms, simply do not expose their model details in their implementations.

Thus, the conventional assumption for an active learning method to have full knowledge of the model details no longer holds true for SRL models. We refer to models which are complex and whose details are unknown as *black-box* SRL models. In this paper, we propose an active learning framework that works for black-box SRL models (see Fig. 1(b)). The main idea is that instead of using a traditional query strategy, we automatically learn a query strategy model from *predictions of the black-box SRL model and a small set of annotated data*. In later active learning process, the query strategy model is able to identify both the most informative predicted SRL label needed to query the human annotator (referred as *human-need SRL label*) and the high-confidence predicted SRL label (referred as *human-free SRL label*) that can be directly added to the training set of the black-box SRL model. The above active learning can also be seen as a combination of traditional active learning and self-training. For the sake of simplicity, we use active learning to denote the

[1] Here we use the PropBank formalism of SRL.

[2] `code.google.com/archive/p/mate-tools/`

[3] `code.google.com/archive/p/clearparser/`



(a) Query strategy model training.

(b) Active model learning.

Figure 1: Active learning for black-box SRL model framework.

process in the rest of the paper.

Given a set of predicted SRL labels from an SRL model trained on an initial training set with little gold annotations, our framework consists of two phases:

Step 1: Query Strategy Model Training We propose a neural query strategy model to select labels for manual curation. The query strategy model is *trained only once* as shown in Fig. 1(a), using a small SRL labeled dataset[4]. It replaces a classic query strategy such as uncertainty sampling that is informed by the details of the SRL model.

Step 2: Active Model Learning The query strategy model predicts whether an SRL label requires manual curation. Expert human annotators curate the human-need SRL labels detected by the query strategy model. Human-free SRL labels and curated human-need SRL labels are then added to the initial training set to retrain the SRL model.

In summary, we use the query strategy model to recover the knowledge about an SRL model via a neural network (e.g., what kind of SRL labels the model is not good at predicting). This knowledge is then used in an active learning process to help generate a better SRL model.

Contributions In summary, our contributions include:

- We propose ACTIVESRL an active learning framework for black-box SRL models to enable higher model performance even when the model details are inaccessible.
- We present a neural query strategy model QUERYM to learn the strategy for selecting the data instances to be added in the next iteration of SRL model training. The neural network naturally incorporates joint language and semantic embeddings to optimize its capability towards SRL. It replaces conventional query strategies that can be employed only when the model details are known.
- Experimental results demonstrate the effectiveness of our query strategy model. With active model learning, the final SRL models achieve significant improvements over the initial ones.

2 Query Strategy Model

In this section, we describe the query strategy model QUERYM as the following classification problem: Given the predicted SRL labels (i.e., output) from the model, the goal of the QUERYM is to classify a predicted SRL label as a *Human-free SRL label* if the predicted SRL label is likely to be the gold SRL label, or a *Human-need SRL label* otherwise.

2.1 Model Overview

We design the query strategy model to address the following challenges: First, both language specific information (e.g., words and phrases) and semantic specific information (e.g., predicates and arguments) impact the predicted SRL labels. For example, the word form of the predicate will determine which role labels are possible. Thus the model needs to capture the interplay between predicate and its arguments. Second, models based on basic language-specific features suffer from data sparsity problem. For example, word form is often sparse in training data and hence does not generalize well across test set.

To address the two challenges, we jointly embed both language and semantic input into a shared low-dimensional vector space: joint language and semantic embedding tackles the former; and embedding is the state-of-the-art solution to the latter. The joint embeddings belong to a neural network which solves the classification problem.

Fig. 2 illustrates the neural QUERYM containing four layers: (1) an input layer: consisting of language text and its associated semantic labels; (2) an embedding layer: taking input layer and outputting language and semantic vector representation of the input; (3) a hidden layer: aligning and embedding language vector and semantic vector in the same vector space; (4) a softmax layer: predicting human-free or human-need SRL labels based on the hidden states as input.

2.2 Model Description

We now describe each layer of QUERYM in more details.

Language Embedding To embed the language specific information, we use Skip-gram model [Mikolov *et al.*, 2013] to find the low-dimensional word representations that are good at predicting the surrounding words in the input sentence. Formally, the basic Skip-gram formulation defines the following conditional likelihood using the softmax function:

$$q(w_j|w_i) = \frac{\exp(\vec{v}'^{T}_{j}\vec{v}_i)}{\sum_{k=1}^{|W|}\exp(\vec{v}'^{T}_{k}\vec{v}_i)} \qquad (1)$$

[4] The labeled dataset isn't necessarily the same as the initial training set for the SRL model.

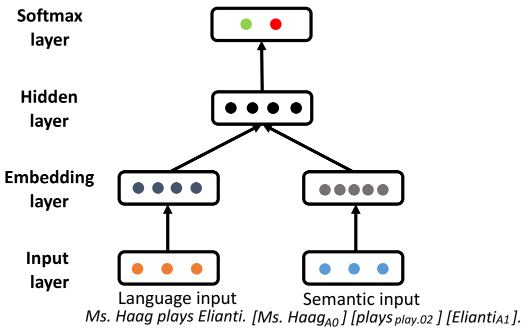

Softmax layer
Hidden layer
Embedding layer
Input layer
Language input
Semantic input
Ms. Haag plays Elianti. [Ms. Haag$_{A0}$] [plays$_{play.02}$] [Elianti$_{A1}$].

Figure 2: QUERYM : Neural query strategy model.

where w_j is the surrounding words of w_i within certain context size, $\vec{v}_k$ and $\vec{v}'_k$ are the word representations of w_k when it appears as the word itself, or the context of other words, $|W|$ is the size of vocabulary.

Given an input consisting of a sequence of words $w_1, w_2, \cdots, w_T$, and the context size c, the objective function of the language embedding is to maximize the average log probability of Eq. (1):

$$\mathcal{L}_{\vec{e}_L} = \sum_{i=1}^{T} \sum_{-c \le j-i \le c, j-i \ne 0} \log q(w_j|w_i) \tag{2}$$

Semantic Embedding We assume that the semantic labels belonging to one frame, i.e., frame labels for predicates, as well as role labels for arguments, should be close in the embedding vector space. To embed the semantic role labels, we explicitly model the interplays between certain argument associated with a predicate and the predicate. Inspired by *TransE* [Bordes *et al.*, 2013; Wang *et al.*, 2014], we first define the score of closeness between predicate and one of its arguments as:

$$z(\vec{p}, \vec{a}) = b - \frac{1}{2}||\vec{p} - \vec{a}||^2 \tag{3}$$

where $\vec{p}$ and $\vec{a}$ are the vector representation of predicate span p and argument span a respectively, b is a constant for bias designated for adjusting the scale for better numerical stability. $z(\vec{p}, \vec{a})$ is expected to be large if the presentation of argument and predicate are close in the vector space.

We define the conditional probability of a predicate-argument structure (p, a) as follows:

$$q(a|p) = \frac{\exp\{z(\vec{p}, \vec{a})\}}{\sum_{a' \in \mathcal{A}} \exp\{z(\vec{p}, \vec{a'})\}} \tag{4}$$

where $\mathcal{A}$ is the set of all argument spans in the corpus.

We also define $q(p|a)$ in the same way by choosing the corresponding normalization term. The objective function of the semantic embedding as below is to maximize the conditional likelihoods of existing predicate-argument structure (p, a) in the corpus.

$$\mathcal{L}_{\vec{e}_S} = \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}} \log(q(a|p) + q(p|a)) \tag{5}$$

where $\mathcal{P}$ is the set of all predicate spans in the corpus.

Hidden Layer The optimal dimensions of the language embedding space and semantic embedding space are usually different. The key challenge of jointly embedding language and semantic becomes how to align the language embedding space with the semantic embedding space. The hidden layer $\vec{h}$ is to combine the language and semantic embeddings through rectified linear units (ReLU).

$$\vec{h} = \max(0, \mathbf{W}^{e_L h}\vec{e}_L + \mathbf{W}^{esh}\vec{e}_S + \vec{b}^h) \tag{6}$$

where $\vec{h}$ represents the joint embedding result (hidden state vector), $\mathbf{W}^{e_L h}$ is the weight matrix connecting language embedding and hidden layer, $\vec{e}_L$ is the language embedding vector, $\mathbf{W}^{esh}$ is the weight matrix connecting semantic embedding and hidden layer, $\vec{e}_S$ is the semantic embedding vector, and $\vec{b}^h$ is the y-intercept in ReLU. The parameter and weight matrices learning methods will be introduced in this section.

Softmax Layer This layer outputs the predicted probability for n-th class using the hidden inputs of each instance.

$$q(y = n|\vec{h}) = \frac{e^{\mathbf{W}_n^{e_L s}\vec{e}_L + \mathbf{W}_n^{e_S s}\vec{e}_S + \mathbf{W}_n^{hs}\vec{h} + \vec{b}_n^s}}{\sum_{k=1}^{|K|} e^{\mathbf{W}_k^{e_L s}\vec{e}_L + \mathbf{W}_k^{e_S s}\vec{e}_S + \mathbf{W}_k^{hs}\vec{h} + \vec{b}_k^s}} \tag{7}$$

The category with the highest probability decides the final predicted category of the input instance.

2.3 Model Optimization

Embedding Layer Local Optimization It is impractical to directly compute the conditional probabilities in both language embedding, i.e., $q(w_j|w_i)$, and semantic embedding, i.e., $q(a|p)$ and $q(p|a)$, because that the computation cost is proportional to $|W|$, $|\mathcal{A}|$ and $|\mathcal{P}|$, which are often very large. To avoid this heavy computation, we adapt negative sampling [Mikolov *et al.*, 2013], which samples multiple negative samples according to some noisy distribution for language and semantic embedding respectively. For language embedding layer, $\log q(w_j|w_i)$ in Eq. (2) is replaced with the following objective function:

$$\log \sigma(\vec{v}'^{T}_j \vec{v}_i) + \sum_{k=1}^{K} \mathbb{E}_{w_k \sim q_n(w)}[\log \sigma(\vec{v}'^{T}_k \vec{v}_i)] \tag{8}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the logistic function. The formulation models the observed word co-occurrence as well as the negative word co-occurrence drawn from the noise distribution, and K is the number of negative samples. We choose $q_n(w) \propto q_u(w)^{\frac{3}{4}}$ as suggested in [Mikolov *et al.*, 2013], where $q_u(w)$ is an unigram distribution over vocabulary. The negative samples from the distribution are considered as words that are never concurrent.

For the semantic embedding layer, we also use negative sampling to convert original objective in Eq. (5) to a simple objective of a binary classification problem, to also differentiate data from noise. We define the probability of an existing predicate-argument structure (p, a) to be labeled as 1 $(y' = 1)$:

$$q(y' = 1|p, a) = \sigma(z(\vec{p}, \vec{a})) \tag{9}$$

where $y' \in \{0, 1\}$.

Similar to that in language embedding, we maximize the following objective instead of $\log q(a|p)$ in objective Eq. (5).

$$\log q(1|p, a) + \sum_{k=1}^{K'} \mathbb{E}_{a_k \sim q_n(a)}[\log q(0|p, a_k)] \tag{10}$$

where K' is the number of negative samples according to every positive sample. We also set $q_n(a) \propto q_u(a)^{\frac{3}{4}}$, where $q_u(a)$ is an unigram distribution over argument spans. The negative samples are then formed by replacing a with the argument span from the noise distribution. We similarly define the same objective for $\log q(p|a)$ by using the according noise distributions. The noise distribution is also set as the unigram distribution raised to the 3/4rd power.

We use the asynchronous stochastic gradient algorithm (ASGD) [Recht *et al.*, 2011] to optimize both Eq. (2) and Eq. (5) with the simplified objectives introduced in this section. In each step, the ASGD algorithm samples a mini-batch of samples to update the model parameters.

The two objectives are simultaneously optimized. Then we use the trained embedding layer to produce the optimized language and semantic embedding.

Neural Model Global Optimization To align the embeddings, we use the hidden layer in neural network structure in Fig. 2. We train the neural network model by taking the derivatives of the loss through backpropagation using the chain rule, with the respect to the whole set of parameters [Collobert *et al.*, 2011], i.e., parameters in Eq. (6) and Eq. (7). We use ASGD to update the parameters.

3 Active Model Learning

Algorithm 1 ACTIVESRL :
Active Learning for Black-box SRL Model.

```
Input: Labeled training data $\mathcal{D}^l_{train}$, labeled test data $\mathcal{D}^l_{test}$, unlabeled data $\mathcal{D}^u$, minimum accuracy change threshold $min\delta$, maximum number of iterations $maxIter$.
Output: An SRL model $\mathcal{L}^*_{srl}$.
1: Train an SRL model $\mathcal{L}_{srl}$ with $\mathcal{D}^l_{train}$;
2: Apply $\mathcal{L}_{srl}$ on $\mathcal{D}^l_{train}$, collect predicted labels;
3: Train QUERYM $\mathcal{L}_q$ based on $\mathcal{D}^l_{train}$ according to Sec. 2.
4: Accuracy change $\delta \leftarrow 0$, $iter \leftarrow 1$;
5: while $\delta > min\delta$ or $iter \leq maxIter$ or $\mathcal{D}^u \neq \emptyset$ do
6:     Apply $\mathcal{L}_q$ on $\mathcal{D}^u$, collect sampled human-free SRL labels $\mathcal{D}^l_f$ and all human-need SRL labels $\mathcal{D}^l_n$;
7:     $\mathcal{D}^l_{train} \leftarrow \mathcal{D}^l_{train} \cup \mathcal{D}^l_f$;
8:     Query human annotator to curate $\mathcal{D}^l_n$, collect curated SRL labels $\mathcal{D}'^l_n$;
9:     $\mathcal{D}^l_{train} \leftarrow \mathcal{D}^l_{train} \cup \mathcal{D}'^l_n$;
10:    An optimized SRL model $\mathcal{L}^*_{srl} \leftarrow$ Retrain $\mathcal{L}_{srl}$ with $\mathcal{D}^l_{train}$;
11:    Apply $\mathcal{L}^*_{srl}$ on $\mathcal{D}^l_{test}$ and record accuracy change $\delta$;
12:    $\mathcal{D}^u = \mathcal{D}^u - (\mathcal{D}^l_f \cup \mathcal{D}^l_n)$, $iter \leftarrow iter + 1$;
13: end while
14: return $\mathcal{L}^*_{srl}$.
```

In this section, we describe the active model learning algorithm ACTIVESRL with the neural query strategy model for black-box SRL model.

The details of ACTIVESRL are shown in Algorithm 1: We begin with two small sets of labeled training and test data, a large unlabeled data, and two stopping criteria for the learning approach: A minimum accuracy change threshold $min\delta$ and a maximum number of iterations $maxIter$. We first train an initial SRL model and the proposed neural query strategy model once (lines 1-3). Then while the stopping criteria are not reached, we repeat the following steps: We apply the query strategy model to the unlabeled data, and collect the predicted SRL labels (line 6). We next directly add all human-free SRL labels to the training data, shuffle partial human-need SRL labels to the human annotator based random sampling, and add the curated human-need SRL labels to the training data (lines 7-9). We retrain the SRL model on the updated training data and evaluate on test data (lines 10-11). We record the change in accuracy in this iteration. The algorithm converges when either the change in accuracy is below $min\delta$, or when $maxIter$ is reached.

4 Experimental Setup

Datasets We conduct all the experiments based on CoNLL-2009 shared task for English [Hajič *et al.*, 2009]. We split the training set into two equal portions: denoted as TRAIN and DEV, and denote the in-domain and out-of-domain test sets in the CoNLL-2009 shared task as TEST_{id} and TEST_{od}.

Black-box SRL Models We select three state-of-the-art SRL models as black-box models. (1) MATE [Björkelund *et al.*, 2010]: it combines the most advanced SRL system and syntactic parser in the CoNLL2009 shared task for English; (2) CLEAR [Choi and Palmer, 2011]: The labeler uses a transition-based SRL algorithm; (3) K-SRL [Akbik and Li, 2016]: it is the current best performing system in CoNLL-2009 shared task for English. All initial SRL models are trained using five-fold cross validation on 50% of TRAIN, denoted as TRAIN_{srl}.

Query Strategy Models We design several comparable query strategy models to compare with the proposed QUERYM, as summarized in Tab. 1. QUERYM_{bow} represents SVM model with traditional bag-of-words model with tf weighting mechanism. The following five models are based on QUERYM, but with only language embedding as the input layer, with QUERYM_{pca} using [Lebret and Collobert, 2013], QUERYM_{global} adopting [Huang *et al.*, 2012], QUERYM_{glove} using [Pennington *et al.*, 2014], QUERYM_{brown} using [Brown *et al.*, 1992], and QUERYM_{eigen} using [Dhillon *et al.*, 2011]. Since the proposed language embedding performs the best among all the above language embeddings as shown in Tab. 1, we don't combine them with the semantic embeddings in our study.

QUERYM_{le} denotes QUERYM with language embedding (trained on full English Wikipedia data[5]) only[6] in the input layer. QUERYM_{se} represents QUERYM with semantic embedding (trained on Propbank data.[7]) only in the input layer.

For each black-box SRL model, all the above query strategy models are trained on 80% of TRAIN_{srl} and tested on 20% of TRAIN_{srl}. The training and test sets are denoted as TRAIN_q and TEST_q respectively.

Active Learning Methods We compare ACTIVESRL (Algorithm 1) with two traditional active learning strategies: Random sampling (RANDSRL) and Uncertainty sampling

[5] http://goo.gl/g1EMX9

[6] Our language embedding is the same as the word2vec [Mikolov *et al.*, 2013] with Skip-gram model.

[7] http://propbank.github.io/

Method	dims	Mate	Clear	K-SRL
QueryM$_{bow}$	-	86.60	87.19	84.40
QueryM$_{pca}$	200	87.52	90.77	92.15
QueryM$_{global}$	50	90.67	92.80	92.10
QueryM$_{glove}$	300	84.60	92.32	91.65
QueryM$_{brown}$	320	91.10	91.27	92.00
QueryM$_{eigen}$	200	85.67	91.33	92.10
QueryM$_{le}$	300	91.80	92.25	92.50
QueryM$_{se}$	200	94.40	93.43	93.25
QueryM	200	**94.95**	**94.11**	**94.10**

Table 1: Results of query strategy models for black-box SRL models on Test$_q$ (accuracy).

(UncertaintySRL) [McCallumzy and Nigamy, 1998]. Among the three SRL models, only K-SRL exposes the model details. So the UncertaintySRL can only be applied to K-SRL. We use the reciprocal of confidence score of each prediction defined in [Akbik and Li, 2016] as the uncertainty score for K-SRL.

ActiveSRL$_{hf}$: ActiveSRL with human-free labels only, where lines 8-9 are skipped in Algorithm 1.

In each iteration of the active learning process (lines 5-13), we select n labels from the unlabeled set Dev to query the human annotators according to the different query strategies. Then we use the curated labels with initial labels to retrain the initial black-box SRL model. n is set as $\frac{|\text{Dev}|}{maxIter}$. We also empirically set $min\delta$ as 0.0001 and $maxIter$ as 10 [Settles, 2010] in line 5 of Algorithm 1. The same parameter values apply to all the above methods. For RandSRL, we ignore the $min\delta$ stopping criterion.

Human Annotators We simulate the expert annotators using the CoNLL-2009 gold SRL annotations.

Evaluation Metrics We use accuracy to measure the quality of the query strategy model and precision, recall and F1-score to measure the quality of SRL models.

5 Experimental Results

In this section, we evaluate the effectiveness of both QueryM and ActiveSRL.

5.1 Neural Query Strategy Model

To test the ability of the neural query strategy model with joint language and semantic embedding, we compare it with other comparable models shown in Sec. 4. We make the following observations from the results (Tab. 1):

Semantic embedding significantly outperforms other embeddings. This result indicates that semantic embedding can better preserve the predicate-argument structure information even with a lower-dimensional vector than the language model. Besides, QueryM$_{se}$ is able to leverage the knowledge of semantic embedding vectors to impact the predictions with at least +1% gain in accuracy.

Joint language and semantic embedding consistently performs the best. The results again show that the semantic embedding brings more semantic information in understanding the text to generate better QueryM. In addition, the language embedding is also useful to capture the hidden semantics in the text when the predicate-argument structure is missing or hard to capture. Even though the optimized dimensions of language embedding and semantic embedding are different, the hidden layer is able to align the two.

Addressing data sparsity. The results indicate that leveraging simple language features alone is not enough to understand the SRL tasks due to the sparsity issues, which can be relieved by leveraging embedding vectors. Embeddings trained over large open-domain datasets are capable to capture more information relevant to SRL task.

5.2 Active Model Learning

We compare the end-to-end SRL performance of ActiveSRL and ActiveSRL$_{hf}$ with other active learning methods described in Sec. 4, as well as the performance of the initial SRL model trained on Train (denoted as "Initial"), and the upper bound performance of the SRL model trained on the entire CoNLL-2009 training set (Train+Dev), denoted as "Upper Bound." All SRL models are learned with each active learning algorithm until reaching convergence. Tab. 2 summarizes the performance on both Test$_{id}$ and Test$_{od}$. We make the following observations:

All SRL models with ActiveSRL$_{hf}$ significantly outperform the initial SRL models. The gains of SRL models with our method suggest that 1) QueryM can identify the human-free labels with high accuracy; and 2) the unknown model knowledge recovered by the query strategy model is useful to improve the performance of SRL models.

ActiveSRL performs well across SRL models and close to upper bound. We observe consistent improvements from ActiveSRL on both datasets. The improvements indicate that QueryM is able to learn the preference of each model, despite of their complexity, their significant differences between each other, and the black-box nature of Mate and Clear. The results also show that ActiveSRL can effectively leverage QueryM to impact the final model quality. The final SRL models also evidently outperform the SRL models trained with ActiveSRL$_{hf}$, indicating that human annotations complement QueryM where the labels are rare and hard to be correctly identified. Furthermore, we observe

Method	Setting	Test$_{id}$			Test$_{od}$		
		P	R	F1	P	R	F1
Mate	Initial	86.11	81.11	83.53	75.70	68.38	71.86
	ActiveSRL$_{hf}$	87.07	82.45	84.70	76.25	70.47	73.25
	ActiveSRL	**87.69**	**83.42**	**85.50**	**76.79**	**71.27**	**73.93**
	Upper Bound	89.59	86.07	87.79	79.46	74.21	76.74
Clear	Initial	82.07	70.57	75.89	72.77	62.14	67.09
	ActiveSRL$_{hf}$	83.12	72.97	77.72	73.02	62.57	67.44
	ActiveSRL	**83.65**	**73.74**	**78.38**	**74.37**	**66.90**	**70.48**
	Upper Bound	84.74	74.47	79.27	75.44	67.20	71.08
K-SRL	Initial	89.54	80.50	84.78	81.39	69.34	74.88
	ActiveSRL$_{hf}$	90.37	82.90	86.48	82.15	71.27	76.33
	ActiveSRL	**91.05**	**84.44**	**87.62**	**82.67**	**72.74**	**77.39**
	Upper Bound	91.21	87.42	89.28	82.09	77.84	79.91

Table 2: SRL results on Test$_{id}$ and Test$_{od}$.

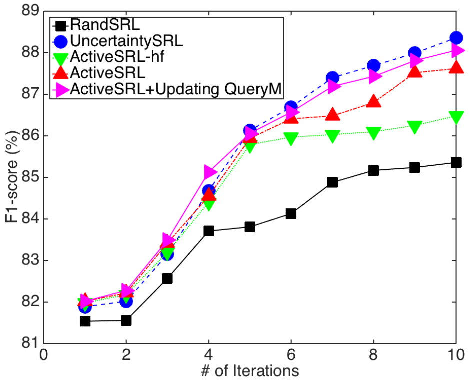


Figure 3: Performance comparison of different active learning methods for K-SRL on Test$_{id}$ (F1-score).

the final performance of SRL models with ActiveSRL are competitive with the "Upper Bound" performance, but with less annotation costs (31.5%).

We further investigate the active learning process by comparing our framework with other active learning methods. We further include ActiveSRL+Updating QueryM to compare: the active learning process is exactly the same as ActiveSRL but the QueryM is retrained on the curated labels from the previous iterations and the initial labels in every iteration. Fig. 3 shows the F1-scores of K-SRL on Test$_{id}$. We make the following observations:

K-SRL with ActiveSRL and ActiveSRL$_{hf}$ performs competitive with UncertaintySRL. UncertaintySRL only outperforms ActiveSRL with +0.74% in F1-score at iteration 10. More interestingly, at the earlier iterations, both ActiveSRL and ActiveSRL$_{hf}$ outperform UncertaintySRL. The reason is that our active learning framework could leverage more human-free labels identified by QueryM when UncertaintySRL only leverages the comparably less amounts of the curated human-need labels at the beginning of the learning process.

K-SRL with ActiveSRL and ActiveSRL$_{hf}$ significantly outperforms RandSRL. The results indicate that our active learning framework is able to identify both human-free labels with high-confidence, as well as assign correct human-need labels to human annotators, with the presence of the QueryM. We also notice that ActiveSRL and ActiveSRL$_{hf}$ finally come to the convergence.

ActiveSRL+Updating QueryM slightly outperforms ActiveSRL. The results mean that the semantic embedding in QueryM is effective in capturing the new label instances, since the embedding is trained on a large annotated label set. This also shows that it is possible to further improve the performance of final SRL model by retraining the query strategy model in each iteration of the active learning. However, the retraining process introduces extra training time. We therefore recommend training the QueryM once in practice.

6 Related Work

Embeddings for NLP aim to use distributional information to represent natural language in lower-dimensional spaces. Most embedding approaches, such as word2vec [Mikolov *et al.*, 2013], Glovec [Pennington *et al.*, 2014] and C&W [Collobert *et al.*, 2011] aim to embed language information (words, phrases and sentences [Palangi *et al.*, 2016]) into vectors, capturing only semantics induced from the distributional information in the data. Dependency path embeddings [Tai *et al.*, 2015; Roth and Lapata, 2016] aim to capture more semantic information from the syntactic structure. Different from the existing embeddings, our semantic embedding explicitly models the semantic information from an SRL annotated corpus. When combined with the language embedding, our joint embedding captures higher-level semantics that benefit semantic-based applications, e.g., SRL.

Neural networks for SRL such as [Collobert *et al.*, 2011], design neural structures to capture the context of words; recent models explore additional language features, e.g., word sequences [Zhou and Xu, 2015], dependency paths [FitzGerald *et al.*, 2015] and compositional embeddings [Roth and Woodsend, 2014]. Without designing a neural SRL model, we use a neural query strategy model to improve an existing SRL model via active learning. Our approach could be adapted to improve existing neural SRL models, an avenue which we leave for future work.

Active learning for NLP has been widely studied [Settles, 2010], e.g., information extraction [Thompson *et al.*, 1999], text classification [McCallumzy and Nigamy, 1998], part-of-speech tagging [Dagan and Engelson, 1995] and natural language parsing [Thompson *et al.*, 1999]. These studies assume that the details of the NLP model are known, and show that sampling techniques such as uncertainty sampling [Lewis and Gale, 1994] and query-by-committe [Dagan and Engelson, 1995] are effective. In contrast, we show how active learning is applicable even when the model details are inaccessible, and use SRL as an example task.

Self-training for NLP uses the existing model to label unlabeled data, which is then treated as additional ground truth to retrain the model. It has been found not too effective, and even damaging in several NLP tasks: parsing [Charniak, 1997], part-of-speech tagging [Clark *et al.*, 2003]. In contrast, our neural query strategy model automatically classifies predicted SRL labels into either suitable as additional ground truth as-is, or requiring human curation.

7 Conclusion

We study the problem of enabling active learning when the details of SRL models are missing or inaccessible. We propose a neural query strategy model to recover the model details (by distinguishing human-free and human-need SRL labels) using a joint language and semantic embedding of an input sentence, and hand over the decisions to the active learning process. We experimentally show that our approach boosts different SRL models to achieve state-of-the-art performance. In the future we plan to apply our active learning framework to other NLP tasks (e.g., dependency parser) and incorporate domain knowledge in the query strategy model [Wang *et al.*, 2015a; 2015b; 2016].

References

[Akbik and Li, 2016] Alan Akbik and Yunyao Li. K-srl: Instance-based learning for semantic role labeling. In *COLING*, pages 599–608, 2016.

[Björkelund *et al.*, 2010] Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. A high-performance syntactic and semantic dependency parser. In *COLING*, pages 33–36, 2010.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[Brown *et al.*, 1992] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *CL*, 18(4):467–479, 1992.

[Charniak, 1997] Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. *AAAI/IAAI*, page 18, 1997.

[Choi and Palmer, 2011] Jinho D Choi and Martha Palmer. Transition-based semantic role labeling using predicate argument clustering. In *ACL 2011 (Workshop)*, pages 37–45, 2011.

[Clark *et al.*, 2003] Stephen Clark, James R Curran, and Miles Osborne. Bootstrapping pos taggers using unlabelled data. In *HLT-NAACL*, pages 49–55, 2003.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12(Aug):2493–2537, 2011.

[Culotta and McCallum, 2005] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, pages 746–51, 2005.

[Dagan and Engelson, 1995] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *ICML*, pages 150–157, 1995.

[Dhillon *et al.*, 2011] Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. Multi-view learning of word embeddings via cca. In *NIPS*, pages 199–207, 2011.

[FitzGerald *et al.*, 2015] Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. Semantic role labeling with neural network factors. In *EMNLP*, pages 960–970, 2015.

[Gildea and Jurafsky, 2002] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *CL*, 28(3):245–288, 2002.

[Hajič *et al.*, 2009] Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *CoNLL*, pages 1–18, 2009.

[Huang *et al.*, 2012] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *ACL*, pages 873–882, 2012.

[Lebret and Collobert, 2013] Rémi Lebret and Ronan Collobert. Word emdeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*, 2013.

[Lewis and Gale, 1994] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR*, pages 3–12, 1994.

[McCallumzy and Nigamy, 1998] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *ICML*, pages 359–367, 1998.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Palangi *et al.*, 2016] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *TASLP*, 24(4):694–707, 2016.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[Recht *et al.*, 2011] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[Roth and Lapata, 2016] Michael Roth and Mirella Lapata. Neural semantic role labeling with dependency path embeddings. In *ACL*, page to appear, 2016.

[Roth and Woodsend, 2014] Michael Roth and Kristian Woodsend. Composition of word representations improves semantic role labelling. In *EMNLP*, pages 407–413, 2014.

[Scheffer *et al.*, 2001] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *IDA*, pages 309–318, 2001.

[Settles, 2010] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[Seung *et al.*, 1992] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Annual workshop on Computational learning theory*, pages 287–294, 1992.

[Tai *et al.*, 2015] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, pages 1556–1566, 2015.

[Thompson *et al.*, 1999] Cynthia A Thompson, Mary Elaine Califf, and Raymond J Mooney. Active learning for natural language parsing and information extraction. In *ICML*, pages 406–414, 1999.

[Wang *et al.*, 2014] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *EMNLP*, pages 1591–1601, 2014.

[Wang *et al.*, 2015a] Chenguang Wang, Yangqiu Song, Ahmed El-Kishky, Dan Roth, Ming Zhang, and Jiawei Han. Incorporating world knowledge to document clustering via heterogeneous information networks. In *KDD*, pages 1215–1224, 2015.

[Wang *et al.*, 2015b] Chenguang Wang, Yangqiu Song, Dan Roth, Chi Wang, Jiawei Han, Heng Ji, and Ming Zhang. Constrained information-theoretic tripartite graph clustering to identify semantically similar relations. In *IJCAI*, pages 3882–3889, 2015.

[Wang *et al.*, 2016] Chenguang Wang, Yangqiu Song, Haoran Li, Ming Zhang, and Jiawei Han. Text classification with heterogeneous information network kernels. In *AAAI*, pages 2130–2136, 2016.

[Zhou and Xu, 2015] Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL*, pages 1127–1137, 2015.