Learning how to Active Learn: A Deep Reinforcement Learning Approach

Meng Fang and **Yuan Li** and **Trevor Cohn**
School of Computing and Information Systems
The University of Melbourne
meng.fang@unimelb.edu.au, yuanl4@student.unimelb.edu.au,
t.cohn@unimelb.edu.au

Abstract

Active learning aims to select a small subset of data for annotation such that a classifier learned on the data is highly accurate. This is usually done using heuristic selection methods, however the effectiveness of such methods is limited and moreover, the performance of heuristics varies between datasets. To address these shortcomings, we introduce a novel formulation by reframing the active learning as a reinforcement learning problem and explicitly learning a data selection policy, where the policy takes the role of the active learning heuristic. Importantly, our method allows the selection policy learned using simulation on one language to be transferred to other languages. We demonstrate our method using cross-lingual named entity recognition, observing uniform improvements over traditional active learning.

1 Introduction

For most Natural Language Processing (NLP) tasks, obtaining sufficient annotated text for training accurate models is a critical bottleneck. Thus active learning has been applied to NLP tasks to minimise the expense of annotating data (Thompson et al., 1999; Tong and Koller, 2001; Settles and Craven, 2008). Active learning aims to reduce cost by identifying a subset of unlabelled data for annotation, which is selected to maximise the accuracy of a supervised model trained on the data (Settles, 2010). There have been many successful applications to NLP, e.g., Tomanek et al. (2007) used an active learning algorithm for CoNLL corpus to get an F_1 score 84% with a reduction of annotation cost of about 48%. In prior work most active learning algorithms are designed for English based on heuristics, such as using uncertainty or informativeness. There has been comparatively little work done about how to learn the active learning strategy itself.

It is no doubt that active learning is extremely important for other languages, particularly low-resource languages, where annotation is typically difficult to obtain, and annotation budgets more modest (Garrette and Baldridge, 2013). Such settings are a natural application for active learning, however there is little work to this end. A potential reason is that most active learning algorithms require a substantial 'seed set' of data for learning a basic classifier, which can then be used for active data selection. However, given the dearth of data in the low-resource setting, this assumption can make standard approaches infeasible.

In this paper,[1] we propose PAL, short for *Policy based Active Learning*, a novel approach for learning a dynamic active learning strategy from data. This allows for the strategy to be applied in other data settings, such as cross-lingual applications. Our algorithm does not use a fixed heuristic, but instead learns how to actively select data, formalised as a reinforcement learning (RL) problem. An intelligent agent must decide whether or not to select data for annotation in a streaming setting, where the decision policy is learned using a deep Q-network (Mnih et al., 2015). The policy is informed by observations including sentences' content information, the supervised model's classifications and its confidence. Accordingly, a rich and dynamic policy can be learned for annotating new data based on the past sequence of annotation decisions.

Furthermore, in order to reduce the dependence on the data in the target language, which may be low resource, we first learn the policy of active

[1] Source code available at https://github.com/mengf1/PAL

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 604–614
Copenhagen, Denmark, September 7–11, 2017.

learning on another language and then transfer it to the target language. It is easy to learn a policy on a high resource language, where there is plentiful data, such as English. We use cross-lingual word embeddings to learn compatible data representations for both languages, such that the learned policy can be easily ported into the other language.

Our work is different for prior work in active learning for NLP. Most previous active learning algorithms developed for NER tasks is based on one language and then applied to the language itself. Another main difference is that many active learning algorithms use a fixed data selection heuristic, such as uncertainty sampling (Settles and Craven, 2008; Stratos and Collins, 2015; Zhang et al., 2016). However, in our algorithm, we implicitly use uncertainty information as one kind of observations to the RL agent.

The remainder of this paper is organised as follows. In Section 2, we briefly review some related work. In Section 3, we present active learning algorithms, which cross multiple languages. The experimental results are presented in Section 4. We conclude our work in Section 5.

2 Related work

As supervised learning methods often require a lot of training data, active learning is a technique that selects a subset of data to annotate for training the best classifier. Existing active learning (AL) algorithms can be generally considered as three categories: 1) uncertainty sampling (Lewis and Gale, 1994; Tong and Koller, 2001), which selects the data about which the current classifier is the most uncertain; 2) query by committee (Seung et al., 1992), which selects the data about which the "committee" disagree most; and 3) expected error reduction (Roy and McCallum, 2001), which selects the data that can contribute the largest model loss reduction for the current classifier once labelled. Applications of active learning to NLP include text classification (McCallumzy and Nigamy, 1998; Tong and Koller, 2001), relation classification (Qian et al., 2014), and structured prediction (Shen et al., 2004; Settles and Craven, 2008; Stratos and Collins, 2015; Fang and Cohn, 2017). Qian et al. used uncertainty sampling to jointly perform on English and Chinese. Stratos and Collins and Zhang et al. deployed uncertainty-based AL algorithms for languages with the minimal supervision.

Deep reinforcement learning (DRL) is a general-purpose framework for decision making based on representation learning. Recently, there are some notable examples include deep Q-learning (Mnih et al., 2015), deep visuomotor policies (Levine et al., 2016), attention with recurrent networks (Ba et al., 2015), and model predictive control with embeddings (Watter et al., 2015). Other important works include massively parallel frameworks (Nair et al., 2015), dueling architecture (Wang et al., 2016) and expert move prediction in the game of Go (Maddison et al., 2015), which produced policies matching those of the Monte Carlo tree search programs, and squarely beaten a professional player when combined with search (Silver et al., 2016). DRL has been also studied in NLP tasks. For example, recently, DRL has been studied for information extraction problem (Narasimhan et al., 2016). They designed a framework that can decide to acquire external evidence and the framework is under the reinforcement learning method. However, there has been fairly little work on using DRL to learn active learning strategies for language processing tasks, especially in cross-lingual settings.

Recent deep learning work has also looked at transfer learning (Bengio, 2012). More recent work in deep learning has also considered transferring policies by reusing policy parameters between environments (Parisotto et al., 2016; Rusu et al., 2016), using either regularization or novel neural network architectures, though this work has not looked at transfer active learning strategies between languages with shared feature space in state.

3 Methodology

We now show how active learning can be formalised as as a decision process, and then show how this allows for the active learning selection policy to be learned from data using deep reinforcement learning. Later we introduce a method for transferring the policy between languages.

3.1 Active learning as a decision process

Active learning is a simple technique for labelling data, which involves first selecting some instances from an unlabelled dataset, which are then annotated by a human oracle, which is then repeated many times until a termination criterion is satisfied, e.g., the annotation budget is exhausted. Most often the selection function is based on the pre-

dictions of a trained model, which has been fit to the labelled dataset at each stage in the algorithm, where datapoints are selected based on the model's predictive uncertainty (Lewis and Gale, 1994), or divergence in predictions over an ensemble (Seung et al., 1992). The key idea of these methods is to find the instances on which the model is most likely to make errors, such that after their labelling and inclusion in the training set, the model becomes more robust to these types of errors on unseen data.

The steps in active learning can be viewed as a decision process, a means of formalising the active learning algorithm as a sequence of decisions, where the stages of active learning correspond to the state of the system. Accordingly, the *state* corresponds to the selected data for labelling and their labels, and each step in the active learning algorithm corresponds to a selection *action*, wherein the heuristic selects the next items from a pool. This process terminates when the budget is exhausted.

Effectively the active learning heuristic is operating as a decision *policy*, a form of function taking as input the current state — comprising the labelled data, from which a model is trained — and a candidate unlabelled data point — e.g., the model uncertainty. This raises the opportunity to consider general policy functions, based on the state and data point inputs, and resulting in a labelling decision, and, accordingly a mechanism for learning such functions from data. We now elaborate on the components of this process, namely the formulation of the decision process, architecture of the policy function, and means of learning the decision policy automatically from data.

3.2 Stream-based learning

For simplicity, we make a streaming assumption, whereby unlabelled data (sentences) arrive in a stream (Lewis and Gale, 1994).[2] As each instance arrives, an agent must decide the action to take, namely whether or not the instance should be manually annotated. This process is illustrated in Figure 1, which illustrates the space of decision sequences for a small corpus. As part of this process, a separate model, p_ϕ, is trained on the labelled data, and updated accordingly as the labelled dataset is expanded as new annotations arrive. This model is central to the policy for choosing the labelling actions at each stage, and for determining the reward for a sequence of actions.

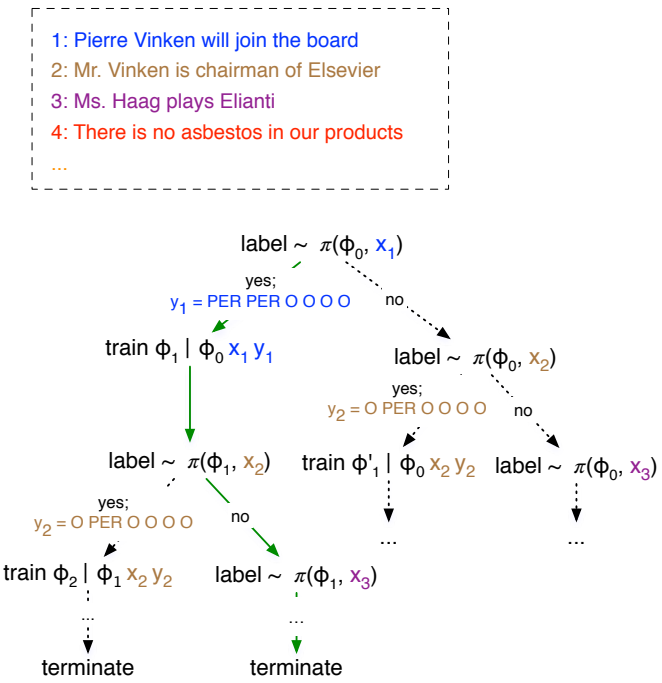

Figure 1: Example illustrating sequential active learning as a Markov Decision process. Data arrives sequentially, and at each time the active learning policy, π, must decide whether it should be labelled or not, based on the state which includes a predictive model parameterised by ϕ, and an unlabelled data instance $\mathbf{x}$. The process continues until termination, e.g., when the annotation budget is exhausted. The solid green path shows the maximum scoring decision sequence.

This is a form of Markov Decision Process (MDP), which allows the learning of a policy that can dynamically select instances that are most informative. As illustrated in Figure 1 at each time, the agent observes the current state s_i which includes the sentence $\mathbf{x}_i$, and the learned model ϕ. The agent selects a binary action a_i, denoting whether to label $\mathbf{x}_i$, according to the policy π. For $a_i = 1$, the corresponding sentence is labelled and added to the labelled data, and the model p_ϕ updated to include this new training point. The process then repeats, terminating when either the dataset is exhausted or a fixed annotation budget is reached. After termination a reward is computed based on the accuracy of the final model, ϕ. We represent the MDP framework as a tuple $\langle S, A, Pr(s_{i+1}|s_i, a), R\rangle$, where $S = \{s\}$

[2] This is different to pool-based active learning, where one of several options is chosen for annotation. Our setup permits simpler learning, while remaining sufficiently general.

is the space of all possible states, $A = \{0, 1\}$ is the set of actions, $R(s, a)$ is the reward function, and $Pr(s_{i+1}|s_i, a)$ is the transition function.

3.2.1 State

The state at time i comprises the candidate instance being considered for annotation and the labelled dataset constructed in steps $1 \ldots i$. We represent the state using a continuous vector, using the concatenation of the vector representation of $\mathbf{x}_i$, and outputs of the model p_ϕ trained over the labelled data. These outputs use both the predictive marginal distributions of the model on the instance, and a representation of the model's confidence. We now elaborate on each component.

Content representation A key input to the agent is the content of the sentence, $\mathbf{x}_i$, which we encode using a convolutional neural network to arrive at a fixed sized vector representation, following Kim (2014). This involves embedding each of the n words in the sentence to produce a matrix $X_i = \{x_{i,1}, x_{i,2}, \cdots, x_{i,n}\}$, after which a series of wide convolutional filters is applied, using multiple filters with different gram sizes. Each filter uses a linear transformation with a rectified linear unit activation function. Finally the filter outputs are merged using a max-pooling operation to yield a hidden state $\mathbf{h}_c$, which is used to represent the sentence.

Representation of marginals The prediction outputs of the training model, $p_\phi(\mathbf{y}|\mathbf{x}_i)$, are central to all active learning heuristics, and accordingly, we include this in our approach. In order to generalise existing techniques, we elect to use the predictive marginals directly, rather than only using statistics thereof, e.g., entropy. This generality allows for different and more nuanced concepts to be learned, including patterns of probabilities that span several adjacent positions in the sentence (e.g., the uncertainty about the boundary of a named entity).

We use another convolutional neural network to process the predictive marginals, as shown in Figure 2. The convolutional layer contains j filters with ReLU activation, based on a window of width 3 and height equal to the number of classes, and with a stride of one token. We use a wide convolution, by padding the input matrix to either size with vectors of zeros. These j feature maps are then subsampled with mean pooling, such that the network is easily able to capture the average uncertainty in each window. The final hidden layer $\mathbf{h}_e$ is used to represent the predictive marginals.

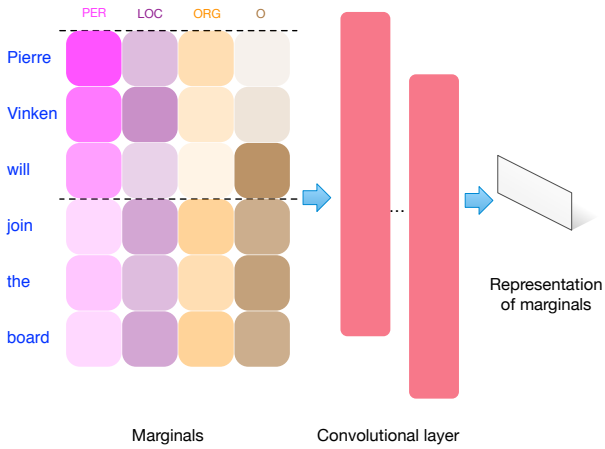

Figure 2: The architecture for representing predictive marginal distributions, $p_\phi(\mathbf{y}|\mathbf{x}_i)$, as a fixed dimensional vector, to form part of the MDP state.

Confidence of sequential prediction The last component is a score C which indicates the confidence of the model prediction. This is defined based on the most probable label sequence under the model, e.g., using Viterbi algorithm with a CRF, and the probability of this sequence is used to represent the confidence, $C = \sqrt[n]{\max_{\mathbf{y}} p_\phi(\mathbf{y}|\mathbf{x}_i)}$, where $n = |\mathbf{x}_i|$ is the length of the sentence.

3.2.2 Action

We now turn to the action, which denotes whether the human oracle must annotate the current sentence. The agent selects either to annotate $\mathbf{x}_i$, in which case $a_i = 1$, or not, with $a_i = 0$, after which the agent proceeds to consider the next instance, $\mathbf{x}_{i+1}$. When action $a_i = 1$ is chosen, an oracle is requested to annotate the sentence, and the newly annotated sentence is added to the training data, and ϕ updated accordingly. A special 'terminate' option applies when no further data remains or the annotation budget is exhausted, which concludes the active learning run (referred to as an 'episode' or 'game' herein).

3.2.3 Reward

The training signal for learning the policy takes the form of a scalar 'reward', which provides feedback on the quality of the actions made by the agent. The most obvious reward is to wait for a game to conclude, then measure the held-out performance of the model, which has been trained

on the labelled data. However, this reward is delayed, and is difficult to related to individual actions after a long game. To compensate for this, we use reward shaping, whereby small intermediate rewards are assigned which speeds up the learning process (Ng, 2003; Lample and Chaplot, 2016). At each step, the intermediate reward is defined as the change in held-out performance, i.e., $R(s_{i-1}, a) = \text{Acc}(\phi_i) - \text{Acc}(\phi_{i-1})$, where Acc denotes predictive accuracy (here F1 score), and ϕ_i is the trained model after action a has take place, which may include an additional training instance. Accordingly, when considering the aggregate reward over a game, the intermediate terms cancel, such that the total reward measures the performance improvement over the whole game. Note that the value of $R(s, a)$ can be positive or negative, indicating a beneficial or detrimental effect on the performance.

3.2.4 Budget

There is a fixed budget $\mathcal{B}$ for the total number of instances annotated, which corresponds to the terminal state in the MDP. It is a predefined number and chosen according to time and cost constraints. A game is finished when the data is exhausted or the budget reached, and with the final result being the dataset thus created, upon which the final model is trained.

3.2.5 Reinforcement learning

The remaining question is how the above components can be used to *learn* a good policy. Different policies make different data selections, and thus result in models with different performance. We adopt a reinforcement learning (RL) approach to learn a policy resulting a highly accurate model.

Having represented the problem as a MDP, episode as a sequence of transitions (s_i, a, r, s_{i+1}). One episode of active learning produces a finite sequence of states, actions and rewards. We use a deep Q-learning approach (Mnih et al., 2015), which formalises the policy using function $Q^\pi(s, a) \rightarrow \mathcal{R}$ which determines the utility of taking a from state s according to a policy π. In Q-learning, the agent iteratively updates $Q(s, a)$ using rewards obtained from each episode, with updates based on the recursive Bellman equation for the optimal Q:

$$Q^\pi(s, a) = \mathbb{E}[R_i | s_i = s, a_i = a, \pi]. \quad (1)$$

Here, $R_i = \sum_{t=i}^{T} \gamma^{t-i} r_t$ is the discounted future reward and $\gamma \in [0, 1]$ is a factor discounting the value of future rewards and the expectation is taken over all transitions involving state s and action a.

Algorithm 1 Learn an active learning policy

Input: data $\mathcal{D}$, budget $\mathcal{B}$
Output: π

for episode = $1, 2, \ldots, N$ **do**
  $\mathcal{D}_l \leftarrow \emptyset$ and shuffle $\mathcal{D}$
  $\phi \leftarrow$ Random
  for $i \in \{0, 1, 2, \ldots, |\mathcal{D}|\}$ **do**
    Construct the state s_i using $\mathbf{x}_i$
    The agent makes a decision according to $a_i = \arg\max Q^\pi(s_i, a)$
    if $a_i = 1$ **then**
      Obtain the annotation $\mathbf{y}_i$
      $\mathcal{D}_l \leftarrow \mathcal{D}_l + (\mathbf{x}_i, \mathbf{y}_i)$
      Update model ϕ based on $\mathcal{D}_l$
    end if
    Receive a reward r_i using held-out set
    if $|\mathcal{D}_l| = \mathcal{B}$ **then**
      Store $(s_i, a_i, r_i, \text{Terminate})$ in $\mathcal{M}$
      Break
    end if
    Construct the new state s_{i+1}
    Store transition (s_i, a_i, r_i, s_{i+1}) in $\mathcal{M}$
    Sample random minibatch of transitions $\{(s_j, a_j, r_j, s_{j+1})\}$ from $\mathcal{M}$, and perform gradient descent step on $\mathcal{L}(\theta)$
    Update policy π with θ
  end for
end for
return the latest policy π

Following Deep Q-learning (Mnih et al., 2015), we make use of a deep neural network to compute the expected Q-value, in order to update the parameters. We implement the Q-function using a single hidden layer neural network, taking as input the state representation $(\mathbf{h}_c, \mathbf{h}_e, C)$ (defined in §3.2.1), and outputting two scalar values corresponding to the values $Q(s, a)$ for $a \in \{0, 1\}$. This network uses a rectified linear unit (ReLU) activation function in its hidden layer.

The parameters in the DQN are learnt using stochastic gradient descent, based on a regression objective to match the Q-values predicted by the DQN and the expected Q-values from the Bellman equation, $r_i + \gamma \max_a Q(s_{i+1}, a; \theta)$. Following (Mnih et al., 2015), we use an experi-

Algorithm 2 Active learning by policy transfer

Input: unlabelled data $\mathcal{D}$, budget $\mathcal{B}$, policy π
Output: $\mathcal{D}_l$

$\mathcal{D}_l \leftarrow \emptyset$
$\phi \leftarrow$ Random
for $|\mathcal{D}_l| \neq \mathcal{B}$ and $\mathcal{D}$ not empty **do**
  Randomly sample $\mathbf{x}_i$ from the data pool $\mathcal{D}$ and construct the state s_i
  The agent chooses an action a_i according to $a_i = \arg\max Q^{\pi}(s_i, a)$
  if $a_i = 1$ **then**
    Obtain the annotation $\mathbf{y}_i$
    $\mathcal{D}_l \leftarrow \mathcal{D}_l + (\mathbf{x}_i, \mathbf{y}_i)$
    Update model ϕ based on $\mathcal{D}_l$
  end if
  $\mathcal{D} \leftarrow \mathcal{D} \backslash \mathbf{x}_i$
  Receive a reward r_i using held-out set
  Update policy π
end for
return $\mathcal{D}_l$

Algorithm 3 Active learning by policy *and* model transfer, for 'cold-start' scenario

Input: unlabelled data $\mathcal{D}$, budget $\mathcal{B}$, policy π, model ϕ
Output: $\mathcal{D}_l$

$\mathcal{D}_l \leftarrow \emptyset$
for $|\mathcal{D}_l| \neq \mathcal{B}$ and $\mathcal{D}$ not empty **do**
  Randomly sample $\mathbf{x}_i$ from the data pool $\mathcal{D}$ and construct the state s_i
  The agent chooses an action a_i according to $a_i = \arg\max Q^{\pi}(s_i, a)$
  if $a_i = 1$ **then**
    $\mathcal{D}_l \leftarrow \mathcal{D}_l + (\mathbf{x}_i, -)$
  end if
  $\mathcal{D} \leftarrow \mathcal{D} \backslash \mathbf{x}_i$
end for
Obtain all the annotations for $\mathcal{D}_l$
return $\mathcal{D}_l$

ence replay memory $\mathcal{M}$ to store each transition (s, a, r, s') as it is used in an episode, after which we sample a mini-batch of transitions from the memory and then minimize the loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}\left[\left(y_i(r, s') - Q(s, a; \theta)\right)^2\right], \quad (2)$$

where $y_i(r, s') = r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})$ is the target Q-value, based on the current parameters θ_{i-1}, and the expectation is over the mini-batch. Learning updates are made every training step, based on stochastic gradient descent to minimise Eq. 2 w.r.t. parameters θ.

The algorithm for learning is summarised in Algorithm 1. We train the policy by running multiple active learning episodes over the training data, where each episode is a simulated active learning run. For each episode, we shuffle the data, and hide the known labels, which are revealed as requested during the run. A disjoint held-out set is used to compute the reward, i.e., model accuracy, which is fixed over the episodes. Between each episode the model is reset to its initialisation condition, with the main changes being the different (random) data ordering and the evolving policy function.

3.3 Cross-lingual policy transfer

We now turn to the question of how the learned policy can be applied to another dataset. Given the extensive use of the training dataset, the policy application only makes sense when employed in a different data setting, e.g., where the domain, task or language is different. For this paper, we consider a cross-lingual application of the same task (NER), where we train a policy on a source language (e.g., English), and then transfer the learned policy to a different target language. Cross-lingual word embeddings provide a common shared representation to facilitate application of the policy to other languages.

We illustrate the policy transfer algorithm in Algorithm 2. This algorithm is broadly similar to Algorithm 1, but has two key differences. Firstly, Algorithm 2 makes only one pass over the data, rather than several passes, as befits an application to a low-resource language where oracle labelling is costly. Secondly, the algorithm also assumes an initial policy, π, which is fine tuned during the episode based on held-out performance such that the policy can adapt to the test scenario.[3]

3.4 Cold-start transfer

The above transfer algorithm has some limitations, which may not be realistic for low-resource settings: the requirement for held-out evaluation data and the embedding of the oracle annotator in-

[3]Moreover, the algorithm can be extended to a traditional batch setting by evaluating a batch of data instances and selectinag the best k instances for labelling under the policy. This could be applied in either the transfer step (Algorithm 2) or initial policy training (Algorithm 1), or both.

side the learning loop. The former implies more supervision than is ideal in a low-resource setting, while the latter places limitations on the communication with annotator as well as a necessity for real-time processing, both which are unlikely in a field linguistics setting.

For this data and- communication-impoverished setting, denoted as *cold-start*, we allow only one chance to request labels for the target data, and, having no held-out data, do not allow policy updates. The agent needs to select a batch of unlabelled target instances for annotations, but cannot use these resulting annotations or any other feedback to refine the selection. In this, more difficult cold-start setting, we bootstrap the process with an initial model, such that the agent can make informative decisions in the absence of feedback.

The procedure is outlined in Algorithm 3. Using the cross-lingual word embeddings, we transfer both a policy and a model into the target language. The model, ϕ, is trained on one source language, and the policy is learned on a different source language. Policy learning uses Alg 1, with the small change that in step 3 the model is initialised using ϕ. Consequently the learned policy can exploit the knowledge from cross-lingual initialisation, such that it can figure out which aspects that need to be corrected using target annotated data. Overall this allows for estimates and confidence values to be produced by the model, thus providing the agent with sufficient information for data selection.

4 Experiments

We conduct experiments to validate the proposed active learning method in a cross-lingual setting, whereby an active learning policy trained on a source language is transferred to a target language. We allow repeated active learning simulations on the source language, where annotated corpora are plentiful, to learn a policy, while for target languages we only permit a single episode, to mimic a language without existing resources.

We use NER corpora from CoNLL2002/2003 shared tasks,[4] which comprise NER annotated text in English (en), German (de), Spanish (es), and Dutch (nl), each annotated using the IOB1 labelling scheme, which we convert to the IO labelling scheme. We use the existing corpus partions, with `train` used for policy training, `testb` used as held-out for computing rewards, and final results are reported on `testa`.

Bilingual		Multilingual		Cold-start		
tgt	**src**	**tgt**	**src**	**tgt**	**src**	**pre**
de	en	de	en,nl,es	de	nl	en
nl	en	nl	en,de,es	nl	de	en
es	en	es	en,de,nl	es	de	en
-	-	-	-	de	es	en
-	-	-	-	nl	es	en
-	-	-	-	es	nl	en

Table 1: Experimental configuration for the three settings, showing target language (**tgt**), source language (**src**) as used for policy learning, and language used for pre-training the model (**pre**).

We consider three experimental conditions, as illustrated in Table 1:

bilingual where English is the source (used for policy learning) and we vary the target language;

multilingual where several source languages are the used in joint learning of the policy, and a separate language is used as target; and

cold-start where a pretrained English NER tagger is used to initialise policy learning on a source language, and in cold-start application to a separate target language.

Configuration We now outline the parameter settings for the experimental runs. For learning an active learning policy, we run $N = 10,000$ episodes with budget $\mathcal{B} = 200$ sentences using Alg. 1. Content representations use three convolutional filters of size $3, 4$ and 5, using 128 filters for each size, while for predictive marginals, the convolutional filters are of width 3, using 20 filters. The size of the last hidden layer is 256. The discount factor is set to $\gamma = 0.99$. We used the ADAM algorithm with mini-batches of size 32 for training the neural network. To report performance, we apply the learned policy to the target training set (using Alg. 2 or 3, again with budget 200),[5] after which we use the final trained model for which we report F_1 score.

[4] http://www.cnts.ua.ac.be/conll2002/ner/, http://www.cnts.ua.ac.be/conll2003/ner/

[5] Although it is possible the policy may learn not to use the full budget, this does not occur in practise.

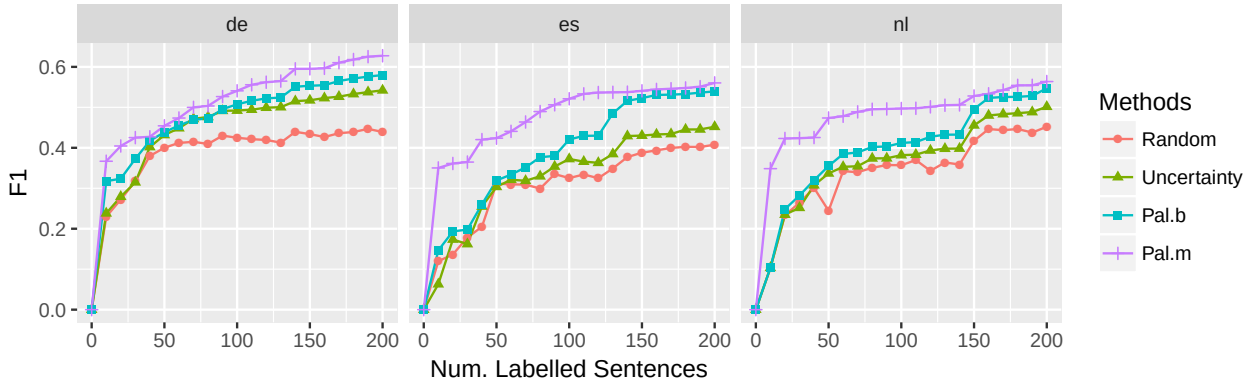

Figure 3: The performance of active learning methods on the bilingual and multilingual settings for three target languages, whereby the active learning policy is trained on only en, or all other languages excluding the target, respectively.

For word embeddings, we use off the shelf CCA trained multilingual embeddings (Ammar et al., 2016),[6] using a 40 dimensional embedding and fixing these during training of both the policy and model. As the model, we use a standard linear chain CRF (Lafferty et al., 2001) for the first two sets of experiments, while for cold-start case we use a basic RNN classifier with the same multilingual embeddings as before, and a 128 dimensional hidden layer.

The proposed method is referred to as PAL, as shorthand *Policy based Active Learning*. Subscripts b, m, c are used to denote the bilingual, multilingual and cold-start experimental configurations. For comparative baselines, we use the following methods:

Uncertainty sampling we use the total token entropy measure (Settles and Craven, 2008), which takes the instance $\mathbf{x}$ maximising $\sum_{t=1}^{|\mathbf{x}|} H(y_t|\mathbf{x}, \phi)$, where H is the token entropy. We use the whole training set as the data pool, and select a single instance for labelling in each active learning step. This method was shown to achieve the best result among model-independent active learning methods on the CoNLL data.

Random sampling which randomly selects examples from the unlabelled pool.

Results Figure 3 shows results the bilingual case, where PAL$_b$ consistently outperforms the Random and Uncertainty baselines across the three target languages. Uncertainty sampling is ineffective, particularly towards the start of the run, as a consequence of its dependence on a high quality model. The use of content information allows PAL$_b$ to make a stronger start, despite the poor initial model.

Also shown in Figure 3 are results for multilingual policy learning, PAL$_m$, which outperform all other approaches including PAL$_b$. This illustrates that the additional training over several languages gives rise to a better policy, than only using one source language. The superior performance is particularly marked in the early stages of the runs for Spanish and Dutch, which may indicate that the approach was better able to learn to exploit the sentence content information.

We evaluate the cold-start setting in Figure 4. Recall that in this setting there are no policy or model updates, as no heldout data is used, and all annotations arrive in a batch. The model, however, is initialised with a NER tagger trained on a different language, which explains why the performance for all methods starts from around 40% rather than 0%. Even in this challenging evaluation setting, our algorithm PAL$_c$ outperforms both baseline methods, showing that deep Q learning allows for better exploitation of the pretrained classifier, alongside the sentence content.

Lastly, we report the results for all approaches in Table 2, based on training on the full 200 labelled sentences as selected under the different methods. It is clear that the PAL methods all outperform the baselines, and among these the multilingual training of PAL$_m$ outperforms the bilingual setting in PAL$_b$. Surprisingly, PAL$_c$ gives the over-

[6] http://128.2.220.95/multilingual

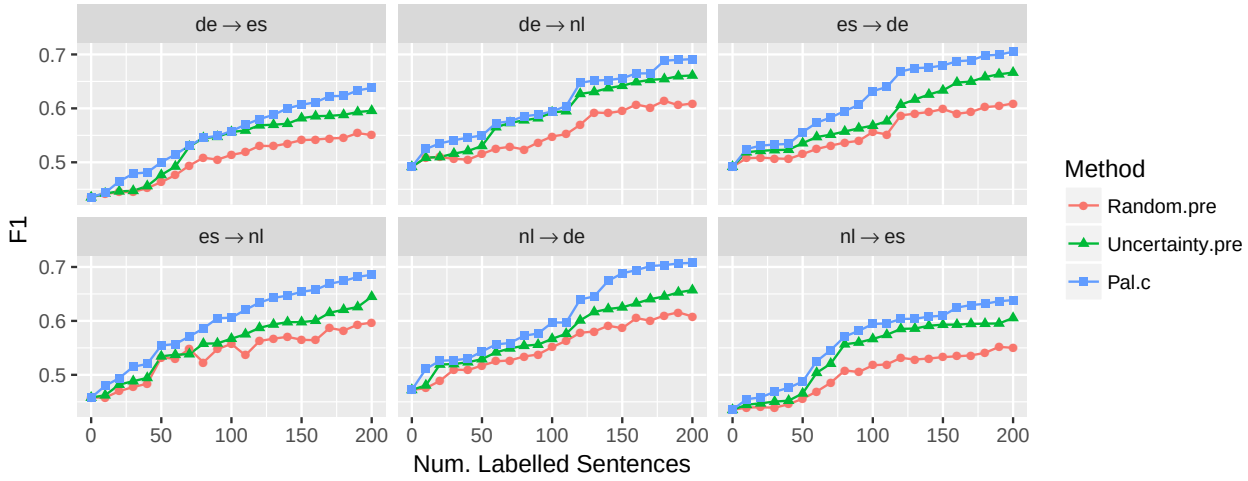

Figure 4: The performance of active learning methods on the cold-start setting, each showing different source → target configurations, in all cases pretraining in en.

	de		nl		es	
	F1	C/R	F1	C/R	F1	C/R
Rand.	44.6	100	45.2	100	40.7	100
Uncert.	54.2	60	50.1	25	45.1	30
PAL$_b$	57.9	60	54.7	25	53.9	40
PAL$_m$	62.7	25	56.3	30	56.0	25
PAL$_c$	70.7	10	69.1	10	63.8	10

Table 2: Results from active learning using the different methods, where each approach constructs a training set of 200 sentences. The three target languages are shown as columns, reporting in each F_1 score (%) and the relative cost reduction to match the stated performance of the Random strategy.

all best results, despite using a static policy and model during target application, underscoring the importance of model pretraining. Table 2 also reports the cost reduction versus random sampling, showing that the PAL methods can reduce the annotation burden to as low as 10%.

5 Conclusion

In this paper, we have proposed a new active learning algorithm capable of learning active learning strategies from data. We formalise active learning under a Markov decision framework, whereby active learning corresponds to a sequence of binary annotation decisions applied to a stream of data. Based on this, we design an active learning algorithm as a policy based on deep reinforcement learning. We show how these learned active learning policies can be transferred between languages, which we empirically show provides consistent and sizeable improvements over baseline methods, including traditional uncertainty sampling. This holds true even in a very difficult cold-start setting, where no evaluation data is available, and there is no ability to react to annotations.

Acknowledgments

This work was sponsored by the Defense Advanced Research Projects Agency Information Innovation Office (I2O) under the Low Resource Languages for Emergent Incidents (LORELEI) program issued by DARPA/I2O under Contract No. HR0011-15-C-0114. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Trevor Cohn was supported by an Australian Research Council Future Fellowship.

References

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

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2015. Multiple object recognition with visual attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yoshua Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. pages 17–36.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of the 55th Annual Meeting on Association for Computational Linguistics (ACL)*.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*. pages 138–147.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*. pages 282–289.

Guillaume Lample and Devendra Singh Chaplot. 2016. Playing FPS games with deep reinforcement learning. *arXiv preprint arXiv:1609.05521* .

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17(39):1–40.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 3–12.

Chris J Maddison, Aja Huang, Ilya Sutskever, and David Silver. 2015. Move evaluation in go using deep convolutional neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Andrew Kachites McCallumzy and Kamal Nigamy. 1998. Employing em and pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning (ICML)*. pages 359–367.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. 2015. Massively parallel methods for deep reinforcement learning. In *Proceedings of ICML Workshop on Deep Learning*.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of the 2016 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Andrew Y. Ng. 2003. *Shaping and Policy Search in Reinforcement Learning*. Ph.D. thesis, University of California, Berkeley.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2016. Actor-mimic: Deep multitask and transfer reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Longhua Qian, Haotian Hui, Yanan Hu, Guodong Zhou, and Qiaoming Zhu. 2014. Bilingual active learning for relation classification via pseudo parallel corpora. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 582–592.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through monte carlo estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*. pages 441–448.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671* .

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

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1070–1079.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the 5th annual workshop on Computational Learning Theory*. pages 287–294.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL)*.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Karl Stratos and Michael Collins. 2015. Simple semi-supervised pos tagging. In *Proceedings of the 2015 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). pages 79–87.

Cynthia A Thompson, Mary Elaine Califf, and Raymond J Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*. pages 406–414.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. pages 486–495.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* 2(Nov):45–66.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. 2015. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems (NIPS)*. pages 2746–2754.

Boliang Zhang, Xiaoman Pan, Tianlu Wang, Ashish Vaswani, Heng Ji, Kevin Knight, and Daniel Marcu. 2016. Name tagging for low-resource incident languages based on expectation-driven learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Language Technologies (NAACL-HLT)*. pages 249–259.