Reinforcement Learning for Multi-Step Expert Advice

Patrick Philipp
Institute AIFB
Karlsruhe Institute of Technology
patrick.philipp@kit.edu

Achim Rettinger
Institute AIFB
Karlsruhe Institute of Technology
rettinger@kit.edu

ABSTRACT

Complex tasks for heterogeneous data sources, such as finding and linking named entities in text documents or detecting objects in images, often require multiple steps to be solved in a processing pipeline. In most of the cases, there exist numerous, exchangeable software components for a single step, each an "expert" for data with certain characteristics. Which expert to apply to which observed data instance in which step becomes a challenge that is even hard for humans to decide. In this work, we treat the problem as Single-Agent System (SAS) where a centralized agent learns how to best exploit experts. We therefore define locality-sensitive relational measures for experts and data points, so-called "meta-dependencies", to assess expert performances, and use them for decision-making via Online Model-Free- and Batch Reinforcement Learning (RL) approaches, building on techniques from Contextual Bandits (CBs) and Statistical Relational Learning (SRL). The resulting system automatically learns to pick the best pipeline of experts for a given set of data points. We evaluate our approach for Entity Linking on text corpora with heterogeneous characteristics (such as news articles or tweets). Our empirical results improve the estimation of expert accuracies as well as the out-of-the-box performance of the original experts without manual tuning.

Keywords

Decision-Making with Multi-Step Expert Advice, Expert Processes, Reinforcement Learning, Collective Learning, Entity Linking

1. INTRODUCTION

Complex tasks for heterogeneous data sources often require multiple steps to be solved in a processing pipeline. Entity linking, for example, is a two-step problem where text is tokenized to (i) find named entities and (ii) link them to a knowledge base. In most of the cases, there exist numerous, exchangeable software components for a single step, each an "expert" for data with certain characteristics. We, thus, call such problems *Decision Making with Multi-Step Expert Advice*.

Combining the outputs of exchangeable experts, e.g. by majority voting, can improve the overall performance, but might fail if experts correlate or display high degrees of variance in their performances. Moreover, two experts solving two subsequent steps might be syntactically compatible while producing bad results when plugged together.

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

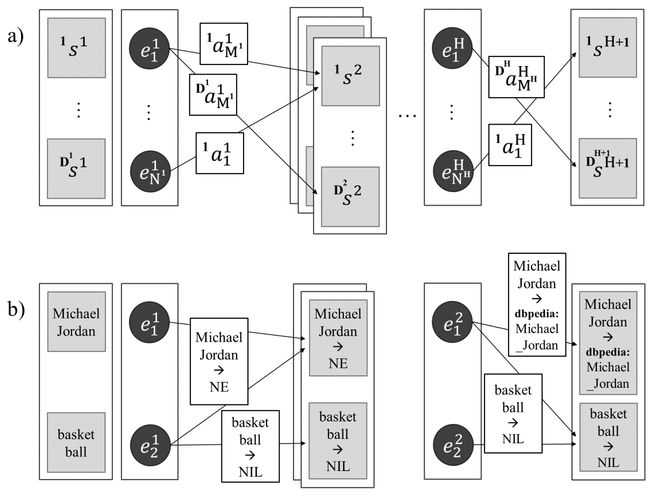

Figure 1: Multi-Step Expert Advice in an Expert Process with a) the formal process and b) an Entity Linking example with two arbitrary experts.

To account for uncertainty, a number of promising candidates might be kept and passed to suitable experts for the subsequent step, instead of using a single output. However, for unstructured data with high dimensionality, such as text or images, the number of potential candidates might grow quickly, rendering brute-force approaches impractical for finding optimal configurations. Learning weights for experts is, thus, crucial to query promising experts and keep "good" candidates throughout multiple steps. This entails learning about impacts of the current decision on future outcomes.

Well-established decision-theoretic frameworks partially deal with these problems, including Decision Making with Expert Advice (EA) [11] where exchangeable experts are sequentially weighted for single-step problems, Budgeted EA [1] where additional budgets are set on expert queries, Contextual Bandits (CBs) [3] where all experts might be queried but only the subset with the chosen action get feedback and, finally, Contextual Markov Decision Processes (CMDPs) [21] which consider a multi-step CB problem setting, where one outcome is rewarded per step.

Still, none of the frameworks cover *Decision Making with Multi-Step Expert Advice* at once, leading us to define so-called *Expert Processes* (EPs). EPs enable learning fine-grained expert weights by considering the impact of current decisions on future experts and not being overly restrictive in rewarding outcomes. To this end, we construct relational measures – so-called "meta-dependencies" – for experts as well as expert pairs, quantifying their predicted performance for specific decision candidates.

EPs with meta-dependencies enable the application of techniques from both Reinforcement Learning (RL) as well as Statistical Relational Learning (SRL) to continuously improve decision-making. Here, relational measures are promising to overcome generalization problems for heterogeneous data distributions, for example available in text or images. These problems occur from assuming data to be independent and identically distributed (i.i.d.), limiting generalization for individual data points. We, therefore, exploit meta-dependencies in two different RL scenarios, where we use the Multiplicative Weights / Hedge (MWH) algorithm [11, 2] for online updates and Probabilistic Soft Logic (PSL) [19, 10] for collective inference.

Our main contributions are:

(i) We provide a formalization for EPs, aligning them with prominent decision-theoretic frameworks.

(ii) We define and use so-called meta-dependencies, enabling the assessment of experts.

(iii) We present two Model-Free RL approaches – Online RL based on the MWH algorithm and Batch RL[23] based on PSL.

(iv) We instantiate our approaches for Entity Linking and empirically evaluate it against Entity Linking experts as well as other expert combination approaches.

The results clearly show that both our approaches outperform all competitors, but more importantly, are the first to consider meta-dependencies between experts and single data points for learning which expert to pick in a pipeline.

The paper proceeds as follows: We, first, formalize EPs (Sec. 2) and introduce meta-dependencies between experts and data (Sec. 3). We, then, present an Online RL- as well as a Batch RL approach to solve EPs in Sec. 4 and Sec. 5. After describing our experimental scenario and setup, we present and elaborate on our results in Sec. 6. We finally integrate our work into prior research (Sec. 7), summarize it (Sec. 8) and point out possible directions (Sec. 9).

2. EXPERT PROCESSES

An expert process EP is a 7-tuple $(S, E, A, R, T, \mathrm{H}, \mathrm{M})$ based on Markov Decision Processes (MDPs) [31] with S the set of all states, E the set of all experts, A the set of all possible actions, $T : S^h \times A^h \to S^{h+1}$ the deterministic transition function, $R : S^h \times A^h \to [0,1]^{\mathrm{D}^h}$ the reward function, H the number of steps and M the budget per step. The state set $S = S^1, \ldots, S^{\mathrm{H}+1}$ consists of $\mathrm{H}+1$ disjunct, heterogeneous state spaces. Hence, a state s^h is uniquely assigned to a step and can be represented in any form, e.g., as a set, sequence, vector, graph or otherwise. Each state s^h also has a set of D^h decision components ${}^1s^h, \ldots, {}^{\mathrm{D}^h}s^h$, i.e. D^h decisions need to be taken for s^h. As EPs are acyclic and deterministic, choosing $T(s^h, a^h)$ will always result in s^{h+1}. The reward $R(s^h, a^h)$ is a vector of local feedback received after choosing action a^h in state s^h, i.e. the vector contains feedback for each decision component. We are given a set E of N^h experts, where in step h of the EP a subset of $E = E^1, \ldots, E^{\mathrm{H}}$ is available for being queried. An expert e^h is a function $e^h : S^h \to AS$, where $AS \subseteq S^h \times S^{h+1}$ are relations for mapping decision candidates. Possible actions ${}^da^h_j \in {}^dA^h$ for ${}^ds^h$ can finally be defined as ${}^dA^h := \{{}^ds^{h+1} | \exists e^h \in E : (s^h, s^{h+1}) \in e^h(s^h)\}$. For simplicity, we denote $\Delta_E : S^h \to S^{h+1}$ the action suggestion of expert e^h for state s^h, which can be inferred from A. The reward function given an expert's suggested action is, then, defined by $R({}^ds^h, \Delta_{e^h}({}^ds^h))$. The budget M^h confines the number of state-expert pairs (e^h, s^h) one is allowed to query in step h.

The distribution over (state,expert)-pairs $P(s^h, e^h)$ is constructed based on weights $w_{e^h} : S^h \to [0,1]^{\mathrm{D}^h}$ for expert e^h.

$$P(s^h, e^h) = \frac{\sum_{{}^ds \in s^h} w_{e^h}({}^ds)}{\sum_{s' \in S^h} \sum_{{}^ds' \in s'} \sum_{e' \in E^h} w_{e'}({}^ds')} \tag{1}$$

The state-action value function (also referred to as Q-function) estimate for taking an action a^h in s^h, is recursively defined as:

$$Q(s^h, a^h) = \sum_{{}^ds \in s^h} \sum_{e \in E^h} w_e({}^ds) * \mathbb{1}_{\{\Delta_e({}^ds) = {}^da\}} \tag{2}$$

with expert weights:

$$\begin{aligned} w_{e^h}({}^ds^h) &= R({}^ds^h, \Delta_{e^h}({}^ds^h)) \\ &+ \delta \sum_{s^{h+1} \in S^{h+1}} T(s^{h+1} | s^h, a^h) \max_{a^{h+1} \in A^{h+1}} Q(s^{h+1}, a^{h+1}) \end{aligned} \tag{3}$$

where δ defines the impact of future rewards. Finally, the state-action distribution $P(s^h, a^h)$ is defined as the normalized state-action value function estimate:

$$P(s^h, a^h) = \frac{Q(s^h, a^h)}{\sum_{s' \in S^h} \sum_{a' \in A^h} Q(s', a')} \tag{4}$$

Similar to MDPs, the $Q-$function takes into account future actions for the current state-action value. More specifically, learning to balance exploration and exploitation, as prevalent in CBs or MDPs, also appears in EPs in terms of learning which experts $e^1, \ldots, e^{\mathrm{H}}$ to query for $s^1, \ldots, s^{\mathrm{H}}$. For multi-step decision problems, this is generally referred to as Reinforcement Learning (RL).

An EP proceeds in episodes $z \in Z$, where for each episode z the process goes through all steps. In each step h for $h = 1, \ldots, \mathrm{H}$:

- We are given $\hat{S}^{h+1}$. // with $|\hat{S^h}| = 1$ for $h = 1$.
- Build state-expert pair distribution $P(s^h, e^h)$.
- Choose and query M^h state-expert pairs (s^h, e^h) and retrieve $a^h_1, \ldots, a^h_{\mathrm{M}}$.
- Build state-action distribution $P(s^h, a^h)$ based on $\hat{A^h} := \{a^h | {}^da^h \in {}^dA^h\}$, containing all possible actions based on $a^h_1, \ldots, a^h_{\mathrm{M}}$.

- Choose $a^h = {}^1a^{\mathrm{h}}, \dots, {}^{\mathrm{D^h}}a^h$ and *receive* $R(s^h, a^h)$, but *observe* all residual rewards $\forall \hat{a}^h \in \hat{A}^h : R(s^h, \hat{a}^h)$ [1].

- If $h < \mathrm{H}$ then set $\hat{S}^{h+1} = \{s^{h+1} | s^{h+1} \in \hat{A}^h\}$.

The reward of the decision process R^{cum} can be defined as the expected cumulative reward collected over H, i.e:

$$R^{cum} = \mathbb{E}[\sum_{h=1}^{\mathrm{H}} \sum_{d=1}^{\mathrm{D}^h} R^h({}^ds^h, {}^da^h)] \tag{5}$$

The overall goal of an EP is to learn how to act in the given decision process, i.e. finding the optimal policy π^* with $\pi : S \to A$ which maximizes R^{cum}.

Our Learning Goal

In this work, we are interested in learning weights $w_{e^h}({}^ds^h)$ for experts, constrained by a budget M^h. This entails continuously or periodically improving policy π while collecting samples (s^h, e^h), as the sample collection process directly influences the quality of the learned weights and vice versa. For learning $w_{e^h}({}^ds^h)$, we exploit rich state spaces to calculate so-called meta-dependencies between experts and decision candidates.

We will next introduce the concept of meta-dependencies between experts and decision candidates, which is the baseline for learning expert weights for EPs.

3. META-DEPENDENCIES

A meta-dependency expresses a relation between a decision candidate ${}^ds^h$ and either (i) two experts e_i^h, e_j^h of the same step, (ii) two experts e_i^h, e_j^{h+1} of subsequent steps or (iii) a single expert e^h. It approximates how single- or pairs of experts behave for a decision candidate based on behavioral measures for one action $\mu : S^h \times A^h \to [0,1]$ or two actions $\mu : S^h \times A^h \times A^h \to [0,1]$ to assess ${}^ds_i^h$ on similar decision candidates ${}^cs_j^h$ in its neighbourhood, using a domain dependent kernel $\kappa \in K$ with $\kappa : S^h \times S^h \to [0,1]$.

While (iii) is the analogy to straight-forward function approximation techniques for the meta-analysis of single experts, (i) and (ii) are classes of pairwise measures which enable an agent to decide which experts cooperate well. Given $\mathbf{MD}(e^h, {}^ds^h)$, the set of all calculated meta-dependencies for $(e^h, {}^ds^h)$, we aim to approximate expert weight $w_{e^h}({}^ds^h)$ assuming different kinds of relationships. We will now present meta-dependencies for each of the three classes.

3.1 Pairwise Intra-Step Experts

Pairwise intra-step meta-dependencies for $(e_i^h, e_j^h, {}^ds^h)$ with $i \neq j$ quantify the relative behaviour between two experts of the same step given a decision candidate, exploiting its neighbourhood. The (ϵ, θ)-dependency of a pair $(e_i^h, e_j^h, {}^ds^h)$ with κ, μ is expressed as:

$$\mathrm{MD}_1^{\kappa,\mu}(e_i^h, e_j^h, {}^ds^h) = \frac{\sum_{{}^ls^h \in N_\kappa({}^ds^h)} \kappa({}^ds^h, {}^ls^h)\mu({}^ls^h, \Delta_{e_i^h}({}^ls^h), \Delta_{e_j^h}({}^ls^h))}{\sum_{{}^ls^h \in N_\kappa({}^ds^h)} \kappa({}^ds^h, {}^ls^h)} \tag{6}$$

with $N_\kappa({}^ds_i^h) = \{{}^cs_j^h | \kappa({}^ds_i^h, {}^cs_j^h) \geq \gamma\}$ [2] the decreasingly sorted neighbourhood of ${}^ds^h$ with minimal similarity γ, $\epsilon = \frac{\sum_{{}^ls^h \in N_\kappa({}^ds^h)} \kappa({}^ds^h, {}^ls^h)}{|N_\kappa({}^ds^h)|}$ and $|N_\kappa({}^ds^h)| \geq \theta$.

To be able to compare two expert actions $\Delta_{e_i^h}({}^ds^h), \Delta_{e_j^h}({}^ds^h)$ we define two behavioural measures for intra-step expert meta-dependencies: joint precision μ_1 and independence μ_2:

$$\mu_1({}^ds^h, {}^da_i^h, {}^da_j^h) = \frac{R({}^ds^h, {}^da_i^h) + R({}^ds^h, {}^da_j^h)}{2} \tag{7}$$

$$\mu_2({}^ds^h, {}^da_i^h, {}^da_j^h) = |R({}^ds^h, {}^da_i^h) - R({}^ds^h, {}^da_j^h)| \tag{8}$$

3.2 Pairwise Inter-Step Experts

Pairwise inter-step meta-dependencies for $(e_i^h, e_j^{h+1}, {}^ds^h)$ with $i \neq j$ quantify the impact of e_i^h on the subsequent step [3]. The (ϵ, θ)-dependency of a pair $(e_i^h, e_j^{h+1}, {}^ds^h)$ with κ, μ is expressed as:

$$\mathrm{MD}_2^{\mu,\kappa}(e_i^h, e_j^{h+1}, {}^ds^h) = \frac{\sum_{{}^ls^h \in N({}^ds^h)} \kappa({}^ds^h, {}^ls^h)\mu(\Delta_{e_i^h}({}^ls^h), \Delta_{e_j^{h+1}}(\Delta_{e_i^h}({}^ls^h)))}{\sum_{{}^ls^h \in N_\kappa({}^ds^h)} \kappa({}^ds^h, {}^ls^h)} \tag{9}$$

with ϵ and θ as defined as before. For inter-step expert meta-dependencies, we define behavioural measure μ_3 as precision of a single expert, i.e.:

$$\mu_3({}^ds^h, {}^da^h) = R({}^ds^h, {}^da^h) \tag{10}$$

3.3 Single Experts

Single expert meta-dependencies for $(e^h, {}^ds^h)$ quantify the behaviour of e^h for ${}^ds^h$ based on similar decision candidates ${}^ls^h$. The (ϵ, θ)-dependency of $(e_i^h, {}^ds^h)$ with κ, μ is expressed as:

$$\mathrm{MD}_3^{\mu,\kappa}(e^h, {}^ds^h) = \frac{\sum_{{}^ls^h \in N_\kappa({}^ds^h)} \kappa({}^ds^h, {}^ls^h)\mu({}^ls^h, \Delta_{e^h}({}^ls^h))}{\sum_{{}^ls^h \in N_\kappa({}^ds)} \kappa({}^ds^h, {}^ls^h)} \tag{11}$$

with ϵ and θ as defined as before. We use the precision of a single expert μ_3 as behavioural measure for single expert meta-dependencies.

We, finally, define a simple kernel-dependent density measure $\psi_\kappa^{\epsilon,\theta}$ to reflect how certain we are in the value of a meta-dependency:

$$\psi_\kappa^{\epsilon,\theta}({}^ds^h) = \begin{cases} \frac{N_\kappa({}^ds^h)}{\theta}\epsilon, & \text{if } |N_\kappa({}^ds^h)| < \theta \\ \epsilon, & \text{otherwise} \end{cases} \tag{12}$$

For expert e^h, the set of meta-dependencies for ${}^ds^h$ is construed as union over all kernels, i.e. $\mathbf{MD}(e^h, {}^ds^h) = \cup_{\forall \kappa \in K} \mathbf{MD}^\kappa(e^h, {}^ds^h)$ with $\mathbf{MD}^\kappa(e^h, {}^ds^h)$:

[1] Feedback might be available in retrospect at $h = \mathrm{H}$.

[2] For simplicity, we assume that $N_\kappa({}^ds_i^h)$ only returns candidates which can be evaluated for a given μ, as numerous state-expert pairs are omitted due to budgets M^h.

[3] As the current state might contain wrong decisions of prior steps, one could also incorporate the impact of e_j^{h-1} on e_i^h if $\exists^c s^{h-1} | e_j^{h-1}({}^cs^{h-1}) = {}^ds^h$.

$$\mathbf{MD}^{\kappa}(e_i^h, {}^d s^h) = \cup_{\forall e_j^h \in E^h \setminus e_i^h \forall \mu \in \{\mu_1,\mu_2\}} \mathrm{MD}_1^{\mu,\kappa}(e_i^h, e_j^h, {}^d s^h)$$
$$\cup_{\forall e_j^{h+1} \in E^{h+1}\}} \mathrm{MD}_2^{\mu_3,\kappa}(e_i^h, e_j^{h+1}, {}^d s^h)$$
$$\mathrm{MD}_3^{\mu_3,\kappa}(e_i^h, {}^d s^h) \tag{13}$$

To approximate $w_{e^h}({}^d s^h)$, single expert meta-dependencies measure the likelihood of an expert's success, while pairwise intra-step expert meta-dependencies use correlations between two experts. As pairwise inter-step expert meta-dependencies take into account future rewards based on experts of the subsequent step, they support approximating $Q({}^d s^h, {}^d a^h)$ (cmp. eq. 2).

Expert weights can be learned in different RL scenarios. In general, there are Model-based, Value Function-based and Policy Search-based RL approaches, whereas the latter two are Model-Free. Both of our approaches are Value Function-based, as we learn the Q-function based on expert weights. To this end there are two classes of methods, namely Online- and Batch RL: Online RL updates Q after each new state, while in Batch RL updates are deferred for a fixed number of episodes to learn supervised models. We next present an Online Model-Free RL approach to learn expert weights.

4. ONLINE RL FOR EXPERT PROCESSES

In the Online RL setting, one aims to continuously improve predictions with every observed reward. We take a Model-Free RL approach, thus not learning R and T (T is known in our special case) but approximating expert weights. Note that most works in RL do not aim to generalize across world states [21] (although numerous approaches deal with function approximation and limited generalization [35, 25, 40]).

To this end, we use meta-dependencies as direct estimators for expert performances. Over multiple episodes, we keep and update meta-weights for meta-dependencies to gradually get more accurate predictions.

We will first describe the Multiplicative Weights/Hedge (MWH) algorithm and map it to meta-dependencies. Based on the latter, different measures related to EA, CBs and MDPs are integrated to account for imperfect information due to the budget and the influence of current decision on future ones.

4.1 MWH with Meta-Dependencies

The Multiplicative Weights (MW) update method [11, 2] entails drawing experts from a normalized probability distribution and updating their weights according to this probability. Hedging over experts [16] extends MW by exponential weights. For an EP, the essential MWH update rule for $w_{e^h}(z)$ can be defined as:

$$w_{e^h}(z+1) =$$
$$w_{e^h}(z)\ exp\left(\frac{\sum_{s^h \in \hat{S}^h} \sum_{{}^d s^h \in s} R^{scaled}({}^d s^h, \Delta_{e^h}({}^d s^h))}{\sum_{s^h \in \hat{S}^h} \sum_{{}^d s^h \in s} \mathbb{1}_{\{\mathrm{q}(e^h, s^h)=1\}}}\ \eta\right) \tag{14}$$

where $\hat{S}^h$ are the states of episode z, $\mathrm{q} : E \times S \to [0,1]$ denotes if expert e was queried for s, $\eta \leq 1$ is the influence parameter of the update, and R^{scaled} returns the rewards rescaled to $[-1, 1]$.

The update rule keeps a global weight for each expert as it does not incorporate any available contextual information. To be able to exploit contextual information, we apply MWH directly to meta-dependencies and refer to them as meta-experts $me \in ME$, where $\forall \mathrm{MD}_i({}^d s^h) \in \mathbf{MD}(e^h, {}^d s^h) \exists me_i : me_i({}^d s^h) = \mathrm{MD}_i({}^d s^h)\ \psi_\kappa^{\epsilon,\theta}({}^d s^h)$ and $\chi : ME \to E$ is the assignment function from meta-experts to experts. A meta-expert, thus, predicts the value of meta-dependency $\mathrm{MD}_i({}^d s^h)$ [4] for candidate ${}^d s^h$ weighted by $\psi_\kappa^{\epsilon,\theta}$. Let now $R^{meta}({}^d s^h, me) = me({}^d s^h)\ R^{scaled}({}^d s^h, \Delta_{\chi(me)}({}^d s^h))$ be the reward function for meta-experts. The resulting update rule for meta-weights $w_{me}(z+1)$ is:

$$w_{me}(z+1) = w_{me}(z)\ exp\left(R^{avg}(\hat{S}^h, me)\ \eta\right) \tag{15}$$

where $R^{avg}(\hat{S}^h, me) = \frac{\sum_{s^h \in \hat{S}^h} \sum_{{}^d s^h \in s^h} R^{meta}({}^d s^h, me)}{\sum_{s^h \in \hat{S}^h} \sum_{{}^d s^h \in s^h} \mathbb{1}_{\{\mathrm{q}(\chi(me), s^h)=1\}}}$. An expert weight, then, is defined by:

$$w_{e^h}({}^d s^h) = \frac{\sum_{me} \mathbb{1}_{\{\chi(me)=e^h\}}\ me({}^d s^h)\ w_{me}(z)}{\sum_{me} w_{me}(z)} \tag{16}$$

However, MWH does not deal with incomplete information resulting from budget M^h. We, thus, adapt techniques from Budgeted EA [1] and adversarial CBs to account for state-expert pairs we did not choose to query.

4.2 MWH with Incomplete Information

As an EP allows to query M^h different state-expert pairs, we extend the weight updates with so-called importance weighting, which was proposed for Budgeted EA. This is achieved by normalizing the reward by the probability of all experts queried in z, h with importance weight $iw(z,h) = \sum_{s \in \hat{S}^h} \sum_{e \in E^h} \mathbb{1}_{\{\mathrm{q}(e,s)=1\}} p(e|s)$:

$$w_{me}(z+1) = w_{me}(z)\ exp\left(\frac{R^{avg}(\hat{S}^h, me)}{iw(z,h)}\ \eta\right) \tag{17}$$

Adversarial CBs do not consider data to be i.i.d. but try to gradually adapt their strategies online. The EXP family of CBs (e.g. [3]) are based on MWH, but extend the approach to update expert weights for non-taken actions, i.e. experts in EPs. While in adversarial CBs all experts are queried to suggest an action, all meta-experts are queried in EPs to suggest a weight for their assigned expert. This is equivalent to a weighted binary decision of each meta-expert, either voting for or against their expert.

EXP4.P [7] is an adversarial bandit approach with low regret (i.e. negative reward) bound. Here, the probability of choosing an action is equivalent with an expert weight in EPs, which we adapt to:

$$w_{e^h}({}^d s^h) = (1 - \mathrm{N}^h\ p_{min}^h)$$
$$\frac{\sum_{me} \mathbb{1}_{\{\chi(me)=e^h\}}\ me({}^d s^h)\ w_{me}(z)}{\sum_{me} w_{me}(z)} + p_{min}^h \tag{18}$$

[4]For simplicity $\mathrm{MD}_i({}^d s^h)$ refers to either $\mathrm{MD}_1^{\kappa,\mu}(e_j^h, e_k^h, {}^d s^h)$, $\mathrm{MD}_2^{\kappa,\mu}(e_j^h, e_k^{h+1}, {}^d s^h)$ or $\mathrm{MD}_3^{\kappa,\mu}(e^h, {}^d s^h)$

where $p^h_{min} = [0, \frac{1}{\mathrm{N}^h}]$ is the minimum probability for choosing any expert.

The update rule of EXP4.P is a modification of MWH (cmp. eq. 15) and Budgeted EA (cmp. eq. 17) in that one ensures exploration for unlikely experts through p^h_{min} and uses confidence bounds to express the variance of the reward. It can be directly used for EPs:

$$w_{me}(z+1) = w_{me}(z)\ exp\Bigg(\frac{p^h_{min}}{2} \Bigg(\mathbb{1}_{\{\chi(me)=e\}} \frac{R^{avg}(\hat{S}^h, me)}{iw(z,h)} + \frac{1}{iw(z,h)}\ \eta\Bigg)\Bigg) \tag{19}$$

Finally, we incorporate outcomes of the chosen joint action by boosting expert weights if an expert's suggested action was wrongfully chosen or wrongfully not chosen, i.e.:

$$\begin{aligned}\mathrm{wrong}^h_z(me) = &1_{\{\mathrm{q}^h_z(\chi(me))=1\}} \\ &\Bigg(1_{\{(\mathrm{q}^h_z(\Delta_e(s))=0)\wedge(R^{meta}(s,\Delta_e(s))>0))\}} \\ &+ 1_{\{(\mathrm{q}^h_z(\Delta_e(s))=1)\wedge(R^{meta}(s,\Delta_e(s))<0)\}}\Bigg)\end{aligned} \tag{20}$$

The resulting boosting factor, then, consists of parameter β which is either active or inactive:

$$\mathrm{boost}^h_z(me) = \beta^{\ \mathrm{wrong}^h_z(me)} \tag{21}$$

The final weight update rule is:

$$w_{me}(z+1) = w_{me}(z)(me)\ exp\Bigg(\frac{p^h_{min}}{2}\ \mathrm{boost}^h_z(me) \Bigg(\mathbb{1}_{\{\chi(me)=e\}} \frac{R^{avg}(\hat{S}^h, me)}{iw(z,h)} + \frac{1}{iw(z,h)}\ \eta\Bigg)\Bigg) \tag{22}$$

Both update rule and expert probability do not specifically incorporate techniques from Model-Free RL approaches, such as the well-known Q-Learning algorithm [41], i.e. future rewards are not explicitly optimized (cmp. eq. 3 for explicit optimization). Here, we rather incorporate future rewards via meta-dependencies, where meta-experts for pairwise inter-step experts take into account how an expert's suggestion influences future results.

Online RL enables reacting to each change in an expert's performance, but this often yields a suboptimal use of samples. Batch RL [23], on the other hand, enables more efficient use of samples while stabilizing the learning process by updating Q less frequently. Batch RL approaches, therefore, defer updates for a fixed number of episodes.

We, now, take a Batch RL perspective on EPs and use SRL for function approximation.

5. BATCH RL FOR EXPERT PROCESSES

Other than in Online RL, Batch RL [23] approaches conduct updates after a fixed number of episodes Z_{batch}, thereby enabling the use of Batch Supervised Learning. The central idea is to use a scalable SRL approach – i.e. Probabilistic Soft Logic (PSL) – for meta-dependencies to learn a collective model for experts of same or subsequent steps.

We consider an adapted version of an EP where batch updates and -predictions are possible. The new EP is changed to deal with Z_{batch} samples within a single step h before continuing to $h+1$. To use all available information of the EP, we train two models for each step h – an a priori model before querying any expert and an a posteriori model after having queried all state-expert pairs [5]. To this end, PSL enables directly integrating kernels into the model, thereby transferring inferred weights to other decision candidates by collective inference.

The section proceeds as follows: we first describe PSL, a template language for generating Hinge-Loss Markov Random Fields. We, then, elaborate on how to model weights $w_{e^h}({}^ds^h)$ to approximate Q.

5.1 Probabilistic Soft Logic and Hinge-loss Markov Random Fields

A Hinge-loss Markov Random Field (HL-MRF) [5] is a conditional probabilistic model over continuous random variables. The use of hinge-loss feature functions makes inference tractable. HL-MRFs are defined as probability densities:

$$\mathrm{P}(Y|X) \propto \frac{1}{Z^{norm}} exp[-\sum_{j=1}^{M} \lambda_j \phi_j(Y,X)]$$

with weights λ_m, linear function l_j, $\Delta_j \in \{0,1\}$, Z^{norm} the respective normalization constant and $\phi_j(X,Y) = [max\{l_j(Y,X), 0\}]^{p_j}$ the hinge-loss potential functions.

PSL [19, 10] is a modeling language for HL-MRFs. A model in PSL comprises weighted, first-order logic rules (templates for ϕ_j) with features defining a Markov network. Here, PSL relaxes conjunctions, disjunctions and negations of Boolean variables A, B as $A \wedge B = \max\{A+B-1, 0\}$, $A \vee B = \min\{A+B, 1\}$ and $\neg A = 1 - A$.

As exact inference for a PSL model is a convex optimization problem, it potentially scales to large datasets, making it applicable for EPs with large state spaces, using meta-dependencies.

5.2 Meta-Dependencies with PSL

We aim to learn expert weights $w_{e^h}({}^ds^h)$ by using PSL to construct different potential functions $\phi_j(X,Y)$ and learn their weights λ_i, where $Y = \cup_{\forall s \in \hat{S}^h \forall^d s \in s \forall e \in E^h} w_e({}^ds)$, $X = \cup_{\forall s \in \hat{S}^h \forall^d s \in s^h \forall e \in E^h} \mathbf{MD}^{\kappa,\mu}(e, {}^ds)$. We construct one PSL model per step h and, other than in our online approach, learn one expert weight function for E^h as our intuition is that meta-dependencies enable to generalize across different experts of the same step.

All n-ary relations we define map their variables to a real number between zero and one, e.g. Relation : $A \times B \rightarrow [0,1]$ for $n = 2$. An expert weight $w_{e^h}({}^ds^h)$ is represented as relation $\mathrm{W}(e^h, {}^ds^h)$. Meta-dependencies for pairwise intra-step experts $\mathrm{MD}^{\kappa,\mu}_1(e^h_i, e^h_j, {}^ds^h)$, pairwise inter-step experts $\mathrm{MD}^{\mu,\kappa}_2(e^h_i, e^{h+1}_j, {}^ds^h)$ and single experts $\mathrm{MD}^{\mu,\kappa}_3(e^h, {}^ds^h)$ are pre-computed meta-dependencies with respective densities $\mathrm{MD}_1\mathrm{-D}^{\kappa,\mu}(e^h_i, e^h_j, {}^ds^h)$,$\mathrm{MD}_2\mathrm{-D}^{\mu,\kappa}(e^h_i, e^{h+1}_j, {}^ds^h)$ and $\mathrm{MD}_3\mathrm{-D}^{\mu,\kappa}(e^h, {}^ds^h)$ (cmp. eq. 12). Densities are calculated

[5]Note that we gradually increase our training set with each batch, where densities provide a natural way to passively forget older samples. An alternative would be to actively forget all old batches but reuse old weights as priors.

based on $\psi_\kappa^{\epsilon,\theta}$. $\mathrm{Tr}(e^h, {}^d s_1^h, {}^d s_2^{h+1})$, the transition relation, denotes that an expert suggests ${}^d s_2^{h+1}$ in state ${}^d s_1^h$.

Pairwise intra-step expert rules express that the weight of an expert increases if $(e_i^h, e_j^h, {}^d s^h)$ is (ϵ, θ)-dependent and (e_i^h, e_j^h) agree on ${}^d s^{h+1}$ with $\mu \in \{\mu_1, \mu_2\}$. We, first, model rules for the a priori setting where no information about expert suggestions is available. Here, we set relation $\mathrm{Agree}(e_1^h, e_2^h, {}^d s_1^h) = \mathrm{MD}_2^{\kappa,\mu_4}(e_1^h, e_2^h, {}^d s_1^h)$ with $\mu_4({}^d s^h, {}^d a_i^h, {}^d a_j^h) = \mathbb{1}_{\{{}^d a_i^h = {}^d a_j^h\}}$.

$$\begin{aligned}\mathrm{MD}_1^{\kappa,\mu}(e_1^h, e_2^h, {}^d s_1^h) &\wedge \mathrm{MD}_1\!-\!\mathrm{D}^{\kappa,\mu}(e_1^h, e_2^h, {}^d s_1^h)\\ &\wedge \mathrm{Agree}(e_1^h, e_2^h, {}^d s_1^h) \implies \mathrm{W}(e_1^h, {}^d s_1^h)\\ \neg\mathrm{MD}_1^{\kappa,\mu}(e_1^h, e_2^h, {}^d s_1^h) &\wedge \mathrm{MD}_1\!-\!\mathrm{D}^{\kappa,\mu}(e_1^h, e_2^h, {}^d s_1^h)\\ &\wedge \mathrm{Agree}(e_1^h, e_2^h, {}^d s_1^h) \implies \neg\mathrm{W}(e_1^h, {}^d s_1^h)\end{aligned}$$

After having queried all state-expert pairs, we can leverage new information by only increasing expert weights if there is agreement.

$$\begin{aligned}&\mathrm{MD}_1^{\kappa,\mu}(e_1^h, e_2^h, {}^d s_1^h) \wedge \mathrm{MD}_1\!-\!\mathrm{D}^{\kappa,\mu}(e_1^h, e_2^h, {}^d s_1^h)\\ &\wedge \mathrm{Tr}(e_1^h, {}^d s_1^h, {}^d s_2^{h+1}) \wedge \mathrm{Tr}(e_2^h, {}^d s_1^h, {}^d s_2^{h+1}) \implies \mathrm{W}(e_1^h, {}^d s_1^h)\\ &\neg\mathrm{MD}_1^{\kappa,\mu}(e_1^h, e_2^h, {}^d s_1^h) \wedge \mathrm{MD}_1\!-\!\mathrm{D}^{\kappa,\mu}(e_1^h, e_2^h, {}^d s_1^h)\\ &\wedge \mathrm{Tr}(e_1^h, {}^d s_1^h, {}^d s_2^{h+1}) \wedge \mathrm{Tr}(e_2^h, {}^d s_1^h, {}^d s_2^{h+1}) \implies \neg\mathrm{W}(e_1^h, {}^d s_1^h)\end{aligned}$$

Rules for pairwise inter-step expert meta-dependencies are direct applications of (ϵ, θ)-dependency for $(e_i^h, e_j^{h+1}, {}^d s^h)$ and its negative implication with $\mu = \mu_3$. Here, we use the same rules for both the a priori and the a posteriori setting.

$$\begin{aligned}\mathrm{MD}_2^{\kappa,\mu}(e_1^h, e_2^{h+1}, {}^d s_2^h) &\wedge \mathrm{MD}_2\!-\!\mathrm{D}^{\kappa,\mu}(e_1^h, e_2^{h+1}, {}^d s_2^h)\\ &\implies \mathrm{W}(e_1^h, {}^d s_2^h)\\ \neg\mathrm{MD}_2^{\kappa,\mu}(e_1^h, e_2^{h+1}, {}^d s_2^h) &\wedge \mathrm{MD}_2\!-\!\mathrm{D}^{\kappa,\mu}(e_1^h, e_2^{h+1}, {}^d s_2^h)\\ &\implies \neg\mathrm{W}(e_1^h, {}^d s_2^h)\end{aligned}$$

Similarly, we directly apply (ϵ, θ)-dependency for $(e^h, {}^d s^h)$ as positive influence on $w_{e^h}({}^d s^h)$ with $\mu = \mu_3$, while the negation also holds. A priori and a posteriori models, again, consist of the same rules.

$$\begin{aligned}\mathrm{MD}_3^{\kappa,\mu}(e^h, {}^d s^h) \wedge \mathrm{MD}_3\!-\!\mathrm{D}^{\kappa,\mu}(e^h, {}^d s^h) &\implies \mathrm{W}(e^h, {}^d s^h)\\ \neg\mathrm{MD}_3^{\kappa,\mu}(e^h, {}^d s^h) \wedge \mathrm{MD}_3\!-\!\mathrm{D}^{\kappa,\mu}(e^h, {}^d s^h) &\implies \neg\mathrm{W}(e^h, {}^d s^h)\end{aligned}$$

We propagate inferred weights by relation $\mathrm{Similar}_\kappa({}^d s_1^h, {}^c s_2^h)$, thus using kernels directly to exploit similarities among decision candidates. It becomes clear that batch predictions might potentially increase the performance of the approach, as kernels cluster similar decision candidates.

$$\begin{aligned}\mathrm{Similar}_\kappa({}^d s_1^h, {}^c s_2^h) \wedge \mathrm{W}(e^h, {}^d s_1^h) &\implies \mathrm{W}(e^h, {}^c s_2^h)\\ \mathrm{Similar}_\kappa({}^d s_1^h, {}^c s_2^h) \wedge \neg\mathrm{W}(e^h, {}^d s_1^h) &\implies \neg\mathrm{W}(e^h, {}^c s_2^h)\end{aligned}$$

We finally align experts with same outputs by driving their weights to be the same as well. For the a priori setting, we define rules based on the agreement probability:

$$\begin{aligned}\mathrm{W}(e_1, {}^d s_1^h) \wedge \mathrm{Agree}(e_1^h, e_2^h, {}^d s_1^h) &\implies \mathrm{W}(e_2^h, {}^d s_1^h)\\ \neg\mathrm{W}(e_1, {}^d s_1^h) \wedge \mathrm{Agree}(e_1^h, e_2^h, {}^d s_1^h) &\implies \neg\mathrm{W}(e_2^h, {}^d s_1^h)\end{aligned}$$

Based on the observed expert action candidates, rules for the a posteriori case are modelled as:

$$\begin{aligned}\mathrm{W}(e_1, {}^d s_1^h) \wedge \mathrm{Tr}(e_1^h, {}^d s_1^h, {}^d s_2^{h+1}) &\wedge \mathrm{Tr}(e_2^h, {}^d s_1^h, {}^d s_2^{h+1})\\ &\implies \mathrm{W}(e_2^h, {}^d s_1^h)\\ \neg\mathrm{W}(e_1^h, {}^d s^h) \wedge \mathrm{Tr}(e_1^h, {}^d s_1^h, {}^c s_2^{h+1}) &\wedge \mathrm{Tr}(e_2^h, {}^d s_1^h, {}^c s_2^{h+1})\\ &\implies \neg\mathrm{W}(e_2^h, {}^d s^h)\end{aligned}$$

For learning weights of the PSL model and inferring $w_{e^h}({}^d s^h)$, we use an approximate maximum likelihood weight learning algorithm and maximum a posteriori (MAP) inference [4].

6. EMPIRICAL EVALUATION

We evaluate our methods based on an EP for named entity recognition (NER) and disambiguation (NED) (cmp. Fig. 1 b)). The resulting EP has H = 2 with S^1 a state space over words, S^2 a state space over named entities and S^3 a state space over disambiguated named entities, i.e. resources available in a (semi-) structured knowledge base (e.g. DBpedia [24], YAGO [36] or Wikipedia[6]). For NER ($h = 1$), decision candidates are n-grams with $n = 1, \dots, |\mathrm{words}(s^1)|$. The actual number of decision candidates is determined by E^1, the set of all NER experts. For NED ($h = 2$), decision candidates are possible named entities, suggested by E^2, the set of all NED experts. Decision candidates for NED outputs ($h = 3$) are single disambiguated named entities.

6.1 Setup

6.1.1 Data

We cross-validate our approaches on the Microposts 2014 corpus [6] and evaluate on the Spotlight corpus [28] after training on Microposts 2014, where we treat each tweet and each sentence within an article as single state sample. We only use tweets for training phases throughout the evaluation to test the ability of transferring meta-dependencies to different data distributions, as the Spotlight Corpus is comparably small and such problems might often occur in practice.

6.1.2 Meta-Dependencies

Kernels: We summarize the kernel set K and the implementations [7] [8] we used in Tab. 1, where *state* or *candidate* respectively indicates whether the kernel is calculated on a piece of text or a decision candidate.

Densities: We vary θ and γ for κ and set:

- $\theta^{\text{length}} = \theta^{\text{extra}} = \theta^{\text{state lingual}} = \frac{Z}{4}$
- $\gamma^{\text{length}} = \gamma^{\text{extra}} = \gamma^{\text{state lingual}} = 0.7$
- $\theta^{\text{embeddings}} = \theta^{\text{minhash}} = \theta^{\text{candidate lingual}} = \frac{Z}{10}$
- $\gamma^{\text{embeddings}} = \gamma^{\text{minhash}} = \gamma^{\text{candidate lingual}} = 0.5$

[6]http://www.wikipedia.org/
[7]https://code.google.com/archive/p/word2vec/
[8]https://github.com/ekzhu/datasketch

Kernel	Implementation
Candidate Lingual Type	Part-of-Speech (POS) tag
Candidate Word embeddings	Pretrained on articles [7]
Candidate MinHash	Jaccard similarity[8]
State Length	Number of characters
State Extra Characters	Count #, @
State Lingual Type	Average POS tag

Table 1: Used kernels for evaluation.

6.1.3 *Experts*

We use expert implementations available as Web services and do not tune parameters.

For NER experts, we use Stanford Tagger [15], FOX [34] and Spotlight Spotter [28]. The NED experts are AGDISTIS [38], AIDA [17] and Spotlight Tagger [28].

6.1.4 *Evaluation measures*

We evaluate our approach in terms of two target measures: a) accuracy estimation of experts and b) outcome optimization of the EP by performing 10-fold cross-validation and using precision, recall and F1-measure, as proposed by entity linking benchmark GERBIL [39] [9]. For all experiments, we set $M^1 = 2, M^2 = 4$.

a) Accuracy Estimation:
The goal is to correctly estimate the weights $w_{e^h}({}^d s^h) \forall e^h \in E^1 \cup E^2$ and, thus, to predict if the suggested action of an expert is correct or not.

- Gl-Avg: Compute global performance weights for experts on left-out training data [33]
- Loc-Avg: Single expert meta-dependencies for ${}^d s^h$
- On-MHW: Online MWH approach to learn weights $w_{e^h}({}^d s^h)$ with $p_{min} = 0.05, \eta = 0.8, \beta = 1.5$
- Batch-PSL: Complete PSL model [10] to learn weights $w_{e^h}({}^d s^h)$ with $Z_{batch} = \frac{Z}{10}$

b) Outcome optimization:
The goal is to optimize the overall outcome of the EP, i.e. R^{cum}.

- AIDA: e^1 = Stanford Tagger, e^2 = AIDA
- DBS: e^1 = DBpedia Spotlight Spotter (dictionary lookup), e^2 = DBpedia Spotlight Tagger
- AGD: e^1 = FOX, e^2 = AGDISTIS
- Gl-Avg, Loc-Avg, On-MHW, Batch-PSL as defined for a) with majority voting according to $P(s^h, a^h)$.

6.2 Results

We show the evaluation results for a) accuracy estimation for $h = 1$ and $h = 2$, and b) outcome optimization in Tab. 2, Tab. 3 and Tab. 4.

[9]For NER a true positive is a correctly found named entity, a true negative is the correctly predicted abstinence of a named entity, a false positive is a mistakenly predicted named entity and a false negative is the mistakenly predicted abstinence of a named entity.

[10]https://github.com/linqs/psl

	Microposts '14			Spotlight		
Appr.	F1	Prec	Rec	F1	Prec	Rec
Gl-Avg	0.71	0.75	0.67	0.7	0.69	0.72
Loc-Avg	0.76	0.65	**0.93**	0.78	0.66	**0.94**
On-MWH	**0.85**	**0.84**	0.87	**0.82**	**0.84**	0.8
Batch-PSL	0.82	0.8	0.85	0.79	0.78	0.81

Table 2: Evaluation results for a) $h = 1$.

	Microposts '14			Spotlight		
Appr.	F1	Prec	Rec	F1	Prec	Rec
Gl-Avg	0.56	0.63	0.51	0.42	0.39	0.46
Loc-Avg	0.71	0.6	0.87	0.56	0.44	**0.78**
On-MWH	0.73	0.67	0.81	0.67	0.62	0.72
Batch-PSL	**0.8**	**0.73**	**0.88**	**0.74**	**0.72**	0.77

Table 3: Evaluation results for a) $h = 2$.

The results show that – for a) (accuracy estimation) – our approaches (Batch-PSL and On-MHW) are able to outperform both Gl-Avg without meta-dependencies and Loc-Avg with single expert meta-dependencies. For b) (outcome optimization), Batch-PSL is able to outperform the straightforward use of experts as well as all learning approaches in terms of F1-measure.

The results for On-MHW are stable and always close to Batch-PSL, only outperforming the latter for a) (accuracy estimation) for $h = 1$.

In general, our approaches are able to automatically learn fine-grained expert weights and produce good combinations of expert outputs.

6.2.1 *Qualitative Results of Meta-Dependencies*

The learned meta-weights show that single- as well as pairwise intra-step expert meta-dependencies both drive weights towards one, while pairwise inter-step expert meta-dependencies have the opposite effect. The meta-dependencies, thus, complement each other to provide well-balanced estimates.

Although kernels for state text length and -extra characters are heuristics, they improve the predictive power. As both the PSL model as well as the adapted version of MWH are easily extensible and scale well, adaptation to new tasks or domains is straightforward.

The textual kernels are most influential, suggesting that additional textual kernels, covering more decision candidates, might substantially improve the predictions. Our approaches, thus, enable exploiting numerous embedded text representations to be combined for specific complex tasks.

Finally, the results suggest that meta-dependencies learned on tweets also improve learning expert weights for news articles, making knowledge transfer possible.

7. RELATED WORK

We, first, situate EPs into well-established decision-theoretic frameworks of SAS and compare our approaches to related SAS techniques. We, then, integrate our work into related MAS branches, as we argue that our approaches might lead towards better coordination and communication for solving EPs in decentralized MAS.

7.1 Single-Agent Systems

	Microposts '14			Spotlight		
Appr.	F1	Prec	Rec	F1	Prec	Rec
DBS	0.33	0.24	**0.51**	0.45	0.38	**0.57**
AGD	0.28	0.59	0.19	0.13	0.56	0.07
AIDA	0.33	0.69	0.21	0.24	**0.69**	0.15
Gl-Avg	0.41	0.51	0.34	0.36	0.48	0.29
Loc-Avg	0.47	0.59	0.38	0.42	0.49	0.37
On-MHW	0.54	0.6	0.5	0.43	0.44	0.42
Batch-PSL	**0.56**	**0.72**	0.46	**0.48**	0.64	0.39

Table 4: Evaluation results for b)

Decision Making with Multi-Step Expert Advice directly maps to SAS where one agent controls all actions (i.e. experts) and is related to (i) Decision Making with Expert Advice (EA) [11], as we are able to observe the rewards of all queried experts. Still, since there is a budget for querying experts we need to take decisions inbetween steps. There are overlaps with (ii) Budgeted EA [1] where not all experts can be queried and (iii) Contextual Bandits (CBs) [3] which reward only one outcome. In addition, as we deal with decision processes over multiple steps, (iv) Contextual Markov Decision Processes (CMDPs) [21] are strongly related. In comparison, we are allowed to explore multiple decision candidates for a single state and need to keep respective probability distributions.

Knowledge-based trust [13] extends knowledge fusion approaches [12], which combine the outputs of multiple triple extractors from multiple websites and could thus be seen as instance of (i), where a combination function over experts is learned based on heterogeneous data. While we do not deal with problem settings where input data might be flawed, our approach could be integrated into knowledge-based trust by providing accurate confidence measures for the used information extractors.

To determine the accuracy of experts in an independent and identically distributed (i.i.d.) data setting, [29, 30] use pairwise error independence measures between binary classifiers without requiring labelled data and, also, fall under (i). We do not solely rely on unlabeled data, but leverage Collective Learning to integrate labeled and unlabeled data. In addition, we assume relations among experts to be i.i.d. and experts are allowed to perform worse than average.

The algorithm selection problem has widely been studied for combinatorial search problems [20] or automatic machine learning (AutoML) [37], and was originally stated by [32]. Algorithm selection problem is a special case of (ii) and (iii), and AutoML falls under (iv) as very rich state spaces have to be dealt with, multiple steps partially occur and a budget is imposed during the learning phase. Automating machine learning pipelines, as approach by auto-sklearn [14], entails automating feature selection, algorithm selection and hyperparameter optimization. While prominent Bayesian optimization techniques share numerous overlaps with strategies to balance exploitation and exploration in CBs or MDPs, our focus is guiding the decision process for single states and their decision candidates.

While ensemble learning deals with learning functions over individual models [43], meta-learning aims to optimize the application of learned models to new data sets based on performance histories and so-called meta-features describing the datasets [9]. Both types of approaches can be seen as batch versions of (i). Our work, however, focuses on selecting experts (i.e. algorithms) for specific data points by exploiting meta-dependencies, where multiple instances of the same expert with different parameterizations could be used.

7.2 Multi-Agent Systems

While we defined EPs for centralized decision-making in SAS, a MAS view would distribute control of one or more experts to decentralized agents, rendering agent communication and coordination important. Here, one would deal with a multi-stage MAS where agents are fully cooperative. Works for single-stage coordination [18, 26] in MAS deal with agents for one-step problems, but extensions to the multi-stage case exist [8]. The latter are MDP extensions and partition the action space according to available agents.

Communication might be limited if agents cannot query their experts in each episode due to a budget or if agents have to choose a subset of agents to communicate with (eg. [42]). To this end, central differences occur for payoffs, where in cooperative scenarios either the joint action is rewarded for all agents [18] (i.e. they receive the same payoffs) or, such as in bandit settings, only those agents receive a reward which voted for the chosen action [26].

8. CONCLUSION

We formalized Expert Processes for Multi-Step Expert Advice by reusing central concepts of Decision Making with Expert Advice, -Budgeted Expert Advice, Markov Decision Processes and Contextual Bandits. For learning distribution-independent expert weights, we introduced meta-dependencies based on characteristics of data points and their decision candidates. We argued that these meta-dependencies can be best exploited by Online Model-Free Reinforcement Learning based on the Multiplicative Weights/Hedge algorithm and Batch Model-Free Reinforcement Learning with a probabilistic relational model as function approximator, both enabling to learn expert weights. We evaluated our approaches for an Entity Linking Expert Process, where they were able to provide superior results for estimating accuracies of experts and optimizing the overall outcome.

9. FUTURE WORK

EPs enable applying RL approaches to Multi-Step Expert Advice. As our Online Model-Free RL approach is based on EA, we do not directly deal with function approximation. There are extensions to well-known RL approaches, such as Q-learning with linear function approximation [27] or Least Squares Policy Iteration (LSPI) [22, 25], which might be helpful to learn functions over meta-experts.

We solve EPs in a SAS, assuming a central agent to decide which experts to query for which states. However, a decentralized MAS might enable higher degrees of scalability due to parallelization and higher degrees of generalizability of learned weights due to additional self-interest of each agent. Approaches for learning decentralized coordination [42] or learning to communicate in MAS [8, 18] might, thus, be beneficial for EPs. In addition, pairwise meta-dependencies, e.g. for intra- or inter-step experts, seem to be readily applicable for optimizing coordination and communication in such a setting.

REFERENCES

[1] K. Amin, S. Kale, G. Tesauro, and D. S. Turaga. Budgeted prediction with expert advice. In *AAAI*, pages 2490–2496, 2015.

[2] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

[4] S. H. Bach, M. Broecheler, L. Getoor, and D. P. O'Leary. Scaling MPE inference for constrained continuous markov random fields with consensus optimization. In *NIPS*, pages 2663–2671, 2012.

[5] S. H. Bach, B. Huang, B. London, and L. Getoor. Hinge-loss markov random fields: Convex inference for structured prediction. In *UAI*, 2013.

[6] A. E. C. Basave, G. Rizzo, A. Varga, M. Rowe, M. Stankovic, and A. Dadzie. Making sense of microposts (#microposts2014) named entity extraction & linking challenge. In *Making Sense of Microposts (WWW)*, pages 54–60, 2014.

[7] A. Beygelzimer, J. Langford, L. Li, L. Reyzin, and R. E. Schapire. Contextual bandit algorithms with supervised learning guarantees. In *AISTATS*, pages 19–26, 2011.

[8] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *IJCAI*, pages 478–485, 1999.

[9] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer, 1 edition, 2008.

[10] M. Bröcheler, L. Mihalkova, and L. Getoor. Probabilistic similarity logic. In *UAI*, pages 73–82, 2010.

[11] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, 1997.

[12] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang. From data fusion to knowledge fusion. *PVLDB*, 7(10):881–892, 2014.

[13] X. L. Dong, E. Gabrilovich, K. Murphy, V. Dang, W. Horn, C. Lugaresi, S. Sun, and W. Zhang. Knowledge-based trust: Estimating the trustworthiness of web sources. *PVLDB*, 8(9):938–949, 2015.

[14] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *NIPS*, pages 2962–2970, 2015.

[15] J. R. Finkel, T. Grenager, and C. D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.

[16] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119 – 139, 1997.

[17] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *EMNLP*, pages 782–792, 2011.

[18] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *AAAI*, pages 326–331, 2002.

[19] A. Kimmig, S. Bach, M. Broecheler, B. Huang, and L. Getoor. A short introduction to probabilistic soft logic. In *NIPS Workshop on Probabilistic Programming: Foundations and Applications*, pages 1–4, 2012.

[20] L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, 35(3):48–60, 2014.

[21] A. Krishnamurthy, A. Agarwal, and J. Langford. PAC reinforcement learning with rich observations. In *NIPS*, pages 1840–1848, 2016.

[22] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[23] S. Lange, T. Gabel, and M. Riedmiller. *Batch Reinforcement Learning*, pages 45–73. Springer Berlin Heidelberg, 2012.

[24] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[25] L. Li, M. L. Littman, and C. R. Mansley. Online exploration in least-squares policy iteration. In *AAMAS*, pages 733–739, 2009.

[26] K. Liu and Q. Zhao. Distributed learning in multi-armed bandit with multiple players. *Trans. Sig. Proc.*, 58(11):5667–5681, 2010.

[27] F. S. Melo and M. I. Ribeiro. Q-learning with linear function approximation. In *COLT*, pages 308–322, 2007.

[28] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: shedding light on the web of documents. In *I-SEMANTICS*, pages 1–8, 2011.

[29] A. Platanios, A. Blum, and T. M. Mitchell. Estimating accuracy from unlabeled data. In *In Proceedings of UAI*, 2014.

[30] E. A. Platanios, A. Dubey, and T. M. Mitchell. Estimating accuracy from unlabeled data: A bayesian approach. In *ICML*, pages 1416–1425, 2016.

[31] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[32] J. R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[33] P. Ruiz and T. Poibeau. Combining open source annotators for entity linking through weighted voting. In *SEM*, 2015.

[34] R. Speck and A. N. Ngomo. Ensemble learning for named entity recognition. In *In ISWC*, pages 519–534, 2014.

[35] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman. PAC model-free reinforcement learning. In *ICML*, pages 881–888, 2006.

[36] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[37] C. Thornton, F. Hutter, H. H. Hoos, and

K. Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *SIGKDD*, pages 847–855. ACM, 2013.

[38] R. Usbeck, A. N. Ngomo, M. Röder, D. Gerber, S. A. Coelho, S. Auer, and A. Both. AGDISTIS - graph-based disambiguation of named entities using linked data. In *ISWC*, pages 457–471, 2014.

[39] R. Usbeck, M. Röder, A. N. Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann, P. Ferragina, C. Lemke, A. Moro, R. Navigli, F. Piccinno, G. Rizzo, H. Sack, R. Speck, R. Troncy, J. Waitelonis, and L. Wesemann. GERBIL: general entity annotator benchmarking framework. In *WWW*, pages 1133–1143, 2015.

[40] T. J. Walsh, I. Szita, C. Diuk, and M. L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *UAI*, pages 591–598, 2009.

[41] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[42] P. Xuan, V. R. Lesser, and S. Zilberstein. Communication decisions in multi-agent cooperation: model and experiments. In *Agents*, pages 616–623, 2001.

[43] Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.