

Margin-based active learning for structured predictions

Kevin Small · Dan Roth

Received: 14 May 2010 / Accepted: 3 July 2010 / Published online: 14 September 2010
© Springer-Verlag 2010

Abstract Margin-based active learning remains the most widely used active learning paradigm due to its simplicity and empirical successes. However, most works are limited to binary or multiclass prediction problems, thus restricting the applicability of these approaches to many complex prediction problems where active learning would be most useful. For example, machine learning techniques for natural language processing applications often require combining multiple interdependent prediction problems—generally referred to as *learning in structured output spaces*. In many such application domains, complexity is further managed by decomposing a complex prediction into a sequence of predictions where earlier predictions are used as input to later predictions—commonly referred to as a *pipeline model*. This work describes methods for extending existing margin-based active learning techniques to these two settings, thus increasing the scope of problems for which active learning can be applied. We empirically validate these proposed active learning techniques by reducing the annotated data requirements on multiple instances of synthetic data, a semantic role labeling task, and a named entity and relation extraction system.

Keywords Active learning · Structured predictions · Structured output spaces · Pipeline models

1 Introduction

As machine learning techniques become more widely adopted, there has been an increased interest in reducing the costs associated with deploying such systems. Although more robust and less expensive to develop than traditional expert system solutions (e.g. [77]) to similar problems, the successful application of machine learning techniques to practical scenarios is often predicated on procuring large labeled data sets. These issues are exacerbated when we require the system perform well over a wide range of data, such as designing an information extraction system that is trained primarily on newswire data knowing that it will be also utilized for financial documents amongst other domains.

This observation regarding the expense of labeled data has motivated copious recent work regarding reducing labeled data requirements. At one extreme of the spectrum is unsupervised learning [32, 40], where all available data is unlabeled and used to perform operations including density estimation, clustering, and model building. As unsupervised learning is often not directly applicable to prediction tasks, a related strategy is to pre-cluster the data and only require labels from representative points [46]. A particularly notable point along the continuum from unsupervised learning to supervised learning is semi-supervised learning (see [81] for a survey), where the learning algorithm is provided with a small amount of labeled data and a large amount of unlabeled data, exploiting regularities over both data sets; popular approaches in this vein include bootstrapping [1] and co-training [11]. The specific paradigm for reducing annotation through partial labeling studied in this work is

This paper extends work previously published as conference proceedings [54, 55]. Most of this work was completed while K. Small was at the University of Illinois at Urbana-Champaign.

K. Small (✉)
Department of Computer Science,
Tufts University, Medford, MA, USA
e-mail: kevin.small@tufts.edu

D. Roth
Department of Computer Science,
University of Illinois at Urbana-Champaign,
Urbana, IL, USA
e-mail: danr@illinois.edu

active learning (see [63] for a survey), where the learning algorithm again receives a small labeled training set and a large unlabeled training set. The innovation of active learning is that the learning algorithm maintains access to the annotator and is allowed to select additional instances to be labeled between training rounds, attempting to reduce costs by labeling exclusively what are expected to be the most useful instances for learning.

Even with large quantities of labeled data, for many applications such as named entity recognition (NER), semantic role labeling (SRL), or relation extraction (RE), it is infeasible to learn a single function which can accurately identify all of the named entities and relations within a sentence. For instance, consider the information extraction example shown in Fig. 1, where we wish to extract all of the {PEOPLE, LOCATION, ORGANIZATION} entities and label any existing relations from a predefined set (e.g. {LOCATEDIN, ORGANIZATIONBASEDIN, SUBSIDIARYOF, LIVESIN,...}).

In these scenarios, a more practical approach is to learn a model which decomposes the learning problem into several local subproblems and then reassembles them to return a predicted global annotation. We refer to a learning problem for which the global prediction task is decomposed into several subtasks which are composed into a global prediction as *learning in structured output spaces*. A second related strategy for managing complexity is to decompose the target prediction into a sequence of predictions where the output from one prediction stage is used as input to later stages, referred to as a *pipeline model*. For example, the system of Fig. 1 may be decomposed into the stages of entity identification, entity classification, and relation classification. For these two *structured prediction* settings, data is particularly expensive and the active learning protocol is particularly suitable for ameliorating the practical labeling costs required for effective learning.

This work describes margin-based active learning methods for both structured output spaces and pipeline models. First, Sect. 2 provides a brief overview of margin-based learning algorithms and margin-based active learning algorithms, which gives an intuition for some of the active learning principles used throughout this paper. Section 3 then introduces a method for active learning in structured output spaces where the interdependencies between output variables are described by a general set of constraints able to represent arbitrary structural forms.

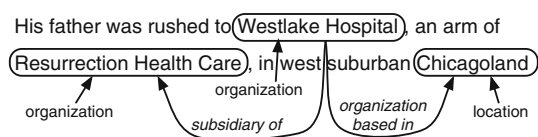


Fig. 1 Named entity and relation extraction from unstructured text

Specifically, we examine active learning querying functions which can query entire structures (querying complete labels) and querying functions which query substructures (querying partial labels). Section 4 extends this work to include pipeline predictions by deriving an active learning framework which looks to sufficiently learn each stage sequentially in order to limit error propagation between learned classifiers. Section 5 empirically validates our methods for active learning in structured output spaces on both synthetic data and the semantic role labeling (SRL) task. Section 6 proceeds by validating our active learning for pipeline models approach on a three-stage named entity and relation extraction system. Finally, Sect. 7 overviews related work and Sect. 8 provides some broad conclusions and future directions for this vein of study.

2 Preliminaries

This section provides background material for the remainder of this work regarding margin-based learning algorithms and margin-based active learning.

2.1 Margin-based learning algorithms

The most widely studied and well understood learning protocol is supervised learning, where a learning algorithm uses labeled instances to formulate a predictive model. More formally, a supervised learning algorithm $\mathcal{A} : \mathcal{S} \times \mathcal{H} \times \mathcal{L} \rightarrow h$ is minimally specified by the following variables:

- $\mathbf{x} \in \mathcal{X}$ represents members of an *input domain* \mathcal{X} .
- $y \in \mathcal{Y}$ represents members of an *output space*. The output space specification often defines the learning problem including regression, $\mathcal{Y} = \mathbb{R}$, binary classification, $\mathcal{Y} = \{-1, 1\}$, multiclass classification, $\mathcal{Y} = \{\omega_1, \omega_2, \dots, \omega_k\}$, amongst others. The form of \mathcal{Y} will be clear from the problem setting.
- $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$ represents a distribution over $\mathcal{X} \times \mathcal{Y}$ from which supervised data is drawn.
- A *training sample* $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ is drawn i.i.d from the probability distribution $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$.
- A *hypothesis space* $\mathcal{H} : \mathcal{X} \rightarrow \mathcal{Y}$ is a family of functions from which the learned hypothesis $h \in \mathcal{H}$ may be selected.
- A *loss function* $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ measures the disagreement between two output elements.

Using this terminology, a learning algorithm is formalized by the following definition:

Definition 1 (Learning algorithm) Given m training examples $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ drawn i.i.d. from a distribution $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$, a hypothesis space \mathcal{H} , and a loss function \mathcal{L} ,

a learning algorithm \mathcal{A} returns a hypothesis function $\hat{h} \in \mathcal{H}$ which minimizes the expected loss \mathcal{L} on a randomly drawn example from $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$, $\hat{h} = \operatorname{argmin}_{h' \in \mathcal{H}} E_{(\mathbf{x}, y) \sim \mathcal{D}_{\mathcal{X} \times \mathcal{Y}}} (\mathcal{L}(h'(\mathbf{x}), y))$.

While it is theoretically desirable to design a learning algorithm as stated in Definition 1 for classification, this is often not feasible in practice. Namely, since the distribution $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$ is unknown and only a finite set of training instances are provided, practical algorithms instead minimize the *empirical loss*, $\hat{h} = \operatorname{argmin}_{h' \in \mathcal{H}} \sum_{i=1}^m \mathcal{L}(h'(\mathbf{x}_i), y)$. Secondly, although minimization of zero-one loss is a meaningful goal as it generally serves as the basis of classifier evaluation, this problem is intractable in its direct form. Therefore, many learning algorithms instead minimize a differentiable function as a surrogate to the ideal loss function for a given task. One widely studied family of learning algorithms which does this are the *margin-based learning algorithms* [2]. To formulate a margin-based learning algorithm in these terms requires the specification of the following additional variables:

- A family of real-valued *hypothesis scoring functions* $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ is a surjective mapping onto \mathcal{H} such that $\hat{y} = h(\mathbf{x}) = \operatorname{argmax}_{y \in \mathcal{Y}} f_y(\mathbf{x})$.
- The *margin* of an instance $\rho : \mathcal{X} \times \mathcal{Y} \times \mathcal{F} \rightarrow \mathbb{R}^+$ is a non-negative real-valued function such that $\rho = 0$ iff $\hat{y} = y$ and its magnitude is associated with the confidence of a prediction \hat{y} for the given input \mathbf{x} relative to a specific hypothesis h .
- A margin-based loss function $\mathcal{L} : \rho \rightarrow \mathbb{R}^+$ measures the disagreement between the predicted output and true output based upon its margin relative to a specified hypothesis.

Based upon this additional terminology, a margin-based learning algorithm is defined as follows:

Definition 2 (Margin-based learning algorithm) Given m training examples $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ drawn i.i.d. from a distribution $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$, a hypothesis scoring function space \mathcal{F} , a definition of margin ρ , and a margin-based loss function \mathcal{L} , a margin-based learning algorithm \mathcal{A} returns a hypothesis scoring function $\hat{f} \in \mathcal{F}$ which minimizes the empirical loss over the training examples to select a hypothesis scoring function $\hat{f} = \operatorname{argmin}_{f' \in \mathcal{F}} \sum_{i=1}^m \mathcal{L}(\rho(\mathbf{x}_i, y_i, f'))$.

An example margin-based loss function which has received significant recent attention in the context of support vector machines (SVM) [75] is *hinge loss*, defined as $\mathcal{L}_{\text{hinge}} = \max\{0, 1 - \rho(\mathbf{x}, y, f)\}$. (1)

Many classic and more recently developed learning algorithms can be cast in this framework by defining an

appropriate margin-based loss function including regression, logistic regression, decision trees [51], and AdaBoost [34].

2.2 Margin-based active learning

Pool-based active learning [20] is a training scheme in which the learning algorithm has access to a pool of unlabeled examples and can request the labels for a number of them with the goal of minimizing the total number of labeled examples required for performance competitive to training with a completely labeled data set. In contrast to *passive learning*, where the learning algorithm receives a random sample of training data to all be processed in a single round of training, active learning allows the learner to incrementally select instances over multiple interactive rounds. During each round, the learner selects those instances which it believes will be most beneficial for later rounds of training. More formally, pool-based active learning begins with a passive learning algorithm \mathcal{A} , an initial data sample $\mathcal{S} = \mathcal{S}_l \cup \mathcal{S}_u$ where it is assumed that there are few labeled instances and many unlabeled instances (i.e. $|\mathcal{S}_l| \ll |\mathcal{S}_u|$), a querying function \mathcal{Q} to select instances for labeling, and access to a domain expert. The primary technical necessity for effective active learning is the querying function as specified by Definition 3.

Definition 3 (Querying function) Given a partially labeled set of instances $\mathcal{S} = \mathcal{S}_l \cup \mathcal{S}_u$, where \mathcal{S}_l and \mathcal{S}_u are the labeled and unlabeled data respectively, and a learned hypothesis $\hat{h} \in \mathcal{H}$, a querying function $\mathcal{Q} : \mathcal{S} \times \mathcal{H} \rightarrow \mathcal{S}_{\text{select}}$ returns a set of instances $\mathcal{S}_{\text{select}} \subseteq \mathcal{S}_u$ which will be labeled by the expert and added to \mathcal{S}_l .

The goal of \mathcal{Q} is to select the subset of \mathcal{S}_u such that when labeled and added to \mathcal{S}_l will generate the largest improvement in the performance of \mathcal{A} , the learning algorithm which generates \hat{h} . Given a querying function and access to a domain expert capable of labeling any instance in the input domain \mathcal{X} , active learning alternates between the three states of learning a new hypothesis \hat{h} based upon the current data \mathcal{S} and learning algorithm \mathcal{A} , using \hat{h} in coordination with \mathcal{Q} to select unlabeled instances $\mathcal{S}_{\text{select}}$, and having the expert label $\mathcal{S}_{\text{select}}$ which is then added to \mathcal{S}_l for the next round of training. Within the active learning framework, the primary research question is the design of the querying function for a particular learning algorithm and application setting. For settings which use a *margin-based learning algorithm*, the specification of margin can also be used to derive a suitable querying function, resulting in a *margin-based querying function*.

Denoting the margin of an example relative to the hypothesis scoring function as $\rho(\mathbf{x}, y, f)$, a margin-based

learning algorithm is a learning algorithm which selects a hypothesis by minimizing a loss function $\mathcal{L} : \mathbb{R} \rightarrow [0, \infty)$ using the margin of instances contained in \mathcal{S}_l . We correspondingly define an active learning algorithm with a querying function dependent on $\rho(\mathbf{x}, y, f)$ as a *margin-based active learning algorithm*. For example, for binary classification (i.e. $\mathcal{Y} \in \{-1, 1\}$), a common definition of margin is $\rho_{\text{binary}}(\mathbf{x}, y, f) = y \cdot f(\mathbf{x})$. As the magnitude of ρ indicates the confidence of the prediction of h on a given unlabeled instance, a commonly used querying function for binary classification is the *minimum margin instance* [14, 61, 72] stated as

$$\mathcal{Q}_{\text{binary}} : \mathbf{x}_\star = \underset{\mathbf{x} \in \mathcal{S}_u}{\operatorname{argmin}} |f(\mathbf{x})|. \quad (2)$$

For multiclass classification, a widely accepted definition for multiclass margin is $\rho_{\text{multiclass}}(\mathbf{x}, \mathbf{y}, f) = f_{\hat{y}}(\mathbf{x}) - f_{\tilde{y}}(\mathbf{x})$ where y represents the true label and $\hat{y} = \operatorname{argmax}_{y' \in \mathcal{Y}} f_{y'}(\mathbf{x})$ corresponds to the highest activation value such that $\hat{y} \neq y$ [39]. Previous work on multiclass active learning [12, 78] advocates a querying function closely related to this definition of multiclass margin where $\hat{y} = \operatorname{argmax}_{y' \in \mathcal{Y}} f_{y'}(\mathbf{x})$ represents the predicted label and $\tilde{y} = \operatorname{argmax}_{y' \in \mathcal{Y}, y' \neq \hat{y}} f_{y'}(\mathbf{x})$ represents the label corresponding to the second highest activation value,

$$\mathcal{Q}_{\text{multiclass}} : \mathbf{x}_\star = \underset{\mathbf{x} \in \mathcal{S}_u}{\operatorname{argmin}} [f_{\hat{y}}(\mathbf{x}) - f_{\tilde{y}}(\mathbf{x})]. \quad (3)$$

By using the active learning protocol with appropriately defined querying functions, the costs associated with obtaining labeled data can often be dramatically reduced, thus facilitating the application of machine learning protocols to more complex application domains.

3 Active learning for structured output spaces

One important framework for learning complex models is *learning in structured output spaces*, where multiple local learners are trained to return predictions which are combined into a global coherent structure through an inference procedure. A classic example of a structured output classifier is the Hidden Markov Model (HMM) [52], which describes a generative model for learning sequential structures. More recently, many conditional structured models have been introduced including Conditional Random Fields (CRF) [43], structured Support Vector Machines [73], structured Perceptron [22], and Max-Margin Markov Networks (M^3N) [68]. The particular framework we study in this work is the Constrained Conditional Model (CCM) [19, 57], which can also be used to frame many of these other structured output formulations [58].

As an illustrative example of structured output, consider the named entity recognition (NER) task shown in Fig. 2.

The NER task requires that, given a sentence, to identify all of the entities that belong to a specified set of word classes [62]. For this particular example, we are considering the classes {PEOPLE, LOCATION, ORGANIZATION}. Therefore, the output space for local predictions would be $\mathcal{Y}_i = \{B, I\} \times \{\text{PEOPLE, LOCATION, ORGANIZATION}\} + O$, where B represents the beginning word of a candidate named entity, I represents a non-beginning (inside) word of a candidate named entity, and O represents a word that is not part of (outside) a candidate named entity. This formulation accounts for both segmentation of the words into entities and labeling the resulting segments. For this particular sentence, we want to annotate *Michael Jordan* as a PEOPLE and *Chicago Bulls* as an ORGANIZATION. In Fig. 2, the histograms at the top represent the local hypothesis scoring functions without structural dependencies. When only considering local context, situations may result in both *Michael* and *Jordan* being first names (i.e. B -PEOPLE) or that *Chicago* would most likely be considered a LOCATION. However, the histograms at the bottom of Fig. 2 represent local predictions which consider structural dependencies. In this hypothetical case, *Michael Jordan* would be considered a single entity comprised of two words due to sequential labeling features and *Chicago Bulls* would be considered a single entity comprised of two words due to output constraints, namely that I -ORGANIZATION must follow a B -ORGANIZATION.

Naturally, complex applications which benefit from a structured output formulation also require more labeled data in addition to each label being of higher annotation cost than for simpler settings. This section describes a margin-based method for active learning in structured output spaces where the interdependencies between output variables are described by a general set of constraints able to represent arbitrary structural form. Specifically, we study two different querying protocols and propose novel querying functions for active learning in structured output space scenarios: *querying complete labels* and *querying*

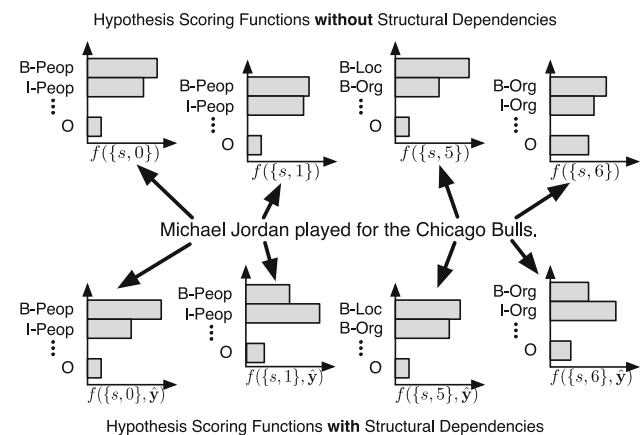


Fig. 2 Learning in structured output spaces

partial labels. In the NER example, these two protocols correspond to requiring the learner to request the labels for entire sentences during the instance selection process or single terms, such as *Bulls*, respectively. We then describe a specific algorithmic implementation of the developed machinery based on the structured Perceptron algorithm and derive a mistake-driven intuition for the relative performance of the querying functions. Finally, we provide empirical evidence on both synthetic data and the semantic role labeling (SRL) task to demonstrate the effectiveness of the proposed methods in Sect. 5.

3.1 Constrained conditional models

The constrained conditional model (CCM) framework for learning in structured output spaces is described in terms of the following variables:

- The input $\mathbf{x} \in \mathbf{X} \subseteq \mathbb{R}^{\sum_{i=1}^{n_x} d_i}$ which is derived by concatenating several local feature vectors $\mathbf{x}(i) \in \mathbf{X}_i \subseteq \mathbb{R}^{d_i}$ such that $i = 1, \dots, n_x$ where n_x is the number of input components for an input structure and d_i is the resulting dimensionality of the i th local input component.
- $y \in \mathcal{Y}$ represents elements of a *structured output space* when \mathcal{Y} can be decomposed into several local output variables $\mathcal{Y} = \mathcal{Y}_1 \times \dots \times \mathcal{Y}_{n_y}$ where n_y is the number of local predictions with respect to a particular instance and $\mathcal{Y}_i = \{\omega_1, \omega_2, \dots, \omega_{k_i}\}$.
- $\tau : \mathcal{Y} \rightarrow \mathbf{Y}$ represents a deterministic transformation function which converts the output structure into a vector of local predictions $\mathbf{y} \in \mathbf{Y}$. Conversely, $\tau^{-1} : \mathbf{Y} \rightarrow \mathcal{Y}$ converts a vector of predictions into an output structure. In a slight abuse of notation, a single transformed output vector is represented by $\mathbf{y} = \langle y(1), y(2), \dots, y(n_y) \rangle$.
- A global hypothesis scoring function, $\mathcal{F} : \mathbf{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, which is a sum over local hypothesis scoring functions, $\mathcal{F}_i : \mathbf{X}_i \times \mathcal{Y}_i \rightarrow \mathbb{R}$, resulting in the global score $f(\mathbf{x}, y) = \sum_{i=1}^{n_y} f_{y(i)}(\mathbf{x}(i))$ where $y(i)$ is the i th element of \mathbf{y} and $\mathbf{x}(i)$ is the local input vector component used to predict $\hat{y}(i)$.
- A set of constraints $\mathcal{C} : 2^{\mathcal{Y}} \rightarrow \{0, 1\}$ enforces global consistency on \mathcal{Y} to ensure that only coherent output structures are generated. To enforce constraints, we require an *inference* procedure which restricts the output space as per the constraints, which we denote by $\mathcal{C}(\mathcal{Y})$.

Using this terminology, learning in structured output spaces can be defined as follows:

Definition 4 (Learning algorithm for structured output spaces) Given a set of structural constraints \mathcal{C} , m training examples $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ drawn i.i.d. from a distribution $\mathcal{D}_{\mathbf{X} \times \mathcal{C}(\mathcal{Y})}$, a hypothesis space \mathcal{H} , and a loss function \mathcal{L} , a structured learning algorithm \mathcal{A} returns a hypothesis

function $\hat{h} \in \mathcal{H}$ which minimizes the expected loss \mathcal{L} on a randomly drawn example from $\mathcal{D}_{\mathbf{X} \times \mathcal{C}(\mathcal{Y})}$, $\hat{h} = \operatorname{argmin}_{h' \in \mathcal{H}} E_{(\mathbf{x}, y) \sim \mathcal{D}_{\mathbf{X} \times \mathcal{C}(\mathcal{Y})}}(\mathcal{L}(h'(\mathbf{x}), y))$.

Given an instance \mathbf{x} , the resulting classification is given by

$$\hat{y}_{\mathcal{C}} = h(\mathbf{x}) = \operatorname{argmax}_{y' \in \mathcal{C}(\mathcal{Y})} f(\mathbf{x}, y'). \tag{4}$$

The output variable assignments are determined by a global hypothesis scoring function $f(\mathbf{x}, y)$ which can be decomposed into local scoring functions $f_{y(i)}(\mathbf{x}(i))$ such that $f(\mathbf{x}, y) = \sum_{i=1}^{n_y} f_{y(i)}(\mathbf{x}(i))$. When structural consistency is not enforced, the global scoring function will output the value $f(\mathbf{x}, \hat{y})$ resulting in assignments given by $\hat{y} = \operatorname{argmax}_{y' \in \mathcal{Y}} f(\mathbf{x}, y')$. An inference mechanism takes the scoring function $f(\mathbf{x}, y)$, an instance (\mathbf{x}, y) , and a set of constraints \mathcal{C} , returning an optimal structurally coherent assignment $\hat{y}_{\mathcal{C}}$ based on the global score $f(\mathbf{x}, \hat{y}_{\mathcal{C}})$ consistent with the defined output structure. Specifically, we will use general constraints with the ability to represent any structure and return a top- k ranking of structures, thereby require a general search mechanism for inference to enforce structural consistency [28]. As active learning querying functions are designed to select instances by exploiting specific structural properties, we define the notions of *locally learnable instances* and *globally learnable instances* for exposition purposes. Informally, locally learnable instances are those instances which can be learned without structural constraints, whereas exclusively globally learnable instances require structural constraints to learn an accurate classifier.

Definition 5 (Locally learnable instance) Given a classifier, $f \in \mathcal{F}$, an instance (\mathbf{x}, y) is locally learnable if $f_{y(i)}(\mathbf{x}(i)) > f_{y'(i)}(\mathbf{x}(i))$ for all $y'(i) \in \mathcal{Y}_i \setminus y(i)$. In this situation, $\hat{y} = \hat{y}_{\mathcal{C}} = y$.

Definition 6 (Exclusively globally learnable instance) Given a classifier, $f \in \mathcal{F}$, an instance (\mathbf{x}, y) is globally learnable if $f(\mathbf{x}, y) > f(\mathbf{x}, y')$ for all $y' \in \mathcal{Y} \setminus y$. We will refer to instances that are globally learnable, but not locally learnable as *exclusively globally learnable* in which case $\hat{y} \neq \hat{y}_{\mathcal{C}} = y$.

3.2 Querying complete labels

The first scenario we examine is the more straightforward case of *querying complete labels*. The task of a querying function for complete labels requires selecting instances \mathbf{x} during each round of active learning such that all output labels associated with the specified instance will be provided by the domain expert. Consider the word alignment example shown in Fig. 3 (example from [29]). Generally accepted as an important subtask of machine translation,



Fig. 3 Bitext word alignment

word alignment is the problem where the input is two parallel strings of text (i.e. a bitext) and the desired output is a bipartite matching between the words of the two strings [47]. Easily modeled as a structured output problem, querying complete labels entails querying the entire bitext. When querying complete labels, the question from the learner to the expert would be, “Can you provide a word alignment for the following bitext?” From one perspective, querying complete labels is the more natural active learning scenario as the annotator often makes annotations based upon the entire text, and designing an appropriate cost function for anything other than the entire structure may require modeling the subtleties of an expert who performs an analysis for the entire structure but labels only a substructure versus simply labeling the entire structure.

Following the margin-based approach for designing querying functions in the multiclass setting (i.e. $\mathcal{Q}_{\text{multiclass}}$), an analogous definition of margin for structured output spaces is stated as

$$\rho_{\text{global}}(\mathbf{x}, y, f) = f(\mathbf{x}, y) - f(\mathbf{x}, \hat{y}_C),$$

where $\hat{y}_C = \operatorname{argmax}_{y' \in \mathcal{C}(\mathcal{Y}) \setminus y} f(\mathbf{x}, y')$. This definition of margin corresponds to the score associated with the correct structure minus the second highest scoring structure (which is not the correct structure). In the ideal case, the querying function would be aware of the correct label, selecting the instance for which the prediction most strongly disagrees with the correct label. However, since the hypothesis cannot guarantee the correct label for unlabeled instances, the predicted label is assumed correct as a proxy for the correct label in the querying function. The semantics of this querying function is selecting instances for the which the learner believed there were many plausible predictions. The corresponding querying function for a structured learner that incorporates the constraints into the learning model for strictly global predictions is defined by

$$\mathcal{Q}_{\text{global}} : x_{\star} = \operatorname{argmin}_{x \in \mathcal{S}_u} [f(\mathbf{x}, \hat{y}_C) - f(\mathbf{x}, y_C)], \quad (5)$$

where $\tilde{y}_C = \operatorname{argmax}_{y' \in \mathcal{C}(\mathcal{Y}) \setminus \hat{y}_C} f(\mathbf{x}, y')$, the second highest scoring structure (which is thereby not the predicted structure). It should be noted that $\mathcal{Q}_{\text{global}}$ does not require $f(\mathbf{x}, y)$ to be decomposable, thereby allowing its use with arbitrary loss functions. The only requirement is that the inference mechanism must be capable of calculating $f(\mathbf{x}, \hat{y}_C)$ and $f(\mathbf{x}, \tilde{y}_C)$ for a given structured instance.

One of the stronger arguments for margin-based active learning is the principle of selecting instances which attempt to halve the version space with each selection [4, 72]. It is straightforward to see that structured output is a special case of multiclass classification where the output space is exponential in the number of variables. For structured output, the version space is stated as

$$\mathcal{V}_{\text{global}} = \{f \in \mathcal{F} | f(\mathbf{x}, y) > f(\mathbf{x}, \tilde{y}); \forall (\mathbf{x}, y) \in \mathcal{S}_l\}.$$

Analogously to the multiclass case in Eq. 3, if we wish to select instances which are most likely to halve the hypothesis space, we should look for instances which are closest to this decision boundary as shown by $\mathcal{Q}_{\text{global}}$ in Eq. 5.

However, for many structured learning settings the scoring function and consequently the loss function is decomposable into local classification problems. Furthermore, it has been observed that when the local classification problems are easy to learn without regard for structural constraints during training, directly optimizing these local functions often leads to a lower sample complexity [49]. As these findings are predicated on making concurrent local updates during learning, selecting structured examples that make as many local updates as possible will generally be desirable for such situations. This observation motivates a querying function which selects instances based on local predictions, resulting in the margin-based strategy of selecting examples with a small average local multiclass margin,

$$\mathcal{Q}_{\text{local}(\mathcal{C})} : x_{\star} = \operatorname{argmin}_{x \in \mathcal{S}_u} \frac{\sum_{i=1}^{n_y} [f_{\hat{y}_C(i)}(\mathbf{x}(i)) - f_{\tilde{y}_C(i)}(\mathbf{x}(i))]}{n_y},$$

where $\hat{y}_C(i) = \operatorname{argmax}_{y' \in Y_i} f_{y'(i)}(\mathbf{x}(i))$, the local prediction after inference has applied the constraints, and $\tilde{y}_C(i) = \operatorname{argmax}_{y' \in Y_i \setminus \hat{y}_C(i)} f_{y'(i)}(\mathbf{x}(i))$, the second highest prediction after constraints have been applied to ensure a coherent structure. We will observe in Sect. 5 that $\mathcal{Q}_{\text{global}}$ is appropriate when the problem output space is highly constrained and $\mathcal{Q}_{\text{local}(\mathcal{C})}$ is more appropriate when the problem can be learned effectively by learning local variables.

3.3 Querying partial labels

We noted that $\mathcal{Q}_{\text{global}}$ makes no assumptions regarding decomposability of the scoring function and $\mathcal{Q}_{\text{local}(\mathcal{C})}$ requires only that the scoring function be decomposable in accordance with the output variables such that the individual local scores may be considered. We now consider active learning in settings where $f(\mathbf{x}, y)$ is decomposable and the local output variables can be queried independently, defined as *querying partial labels*. The intuitive advantage of querying partial labels is that we are no longer subject to cases where a structured instance has one output variable with a very informative label, but the other output variables of the

same instance are minimally useful and yet add cost to the labeling effort. In the word alignment example of Fig. 3, *querying partial labels* would be akin to having the learner ask the question, “Which of the French words does *likes* align with?” This is an appealing model in many cases as the learner may be very confident regarding a subset of the predictions, but is unsure about a few substructures. While this configuration is not immediately usable for applications with a scoring function not easily decomposable into local output variables that can be independently queried (e.g. parsing), we will see this approach is very beneficial in scenarios where such capabilities are available (e.g. semantic role labeling, chunking, word alignment).

Observing that querying partial labels requires requesting a single multiclass classification, the naive querying function for this case is to simply ignore the structural information and use $\mathcal{Q}_{\text{multiclass}}$, resulting in the querying function

$$\mathcal{Q}_{\text{local}} : \mathbf{x}(i)_{\star} = \underset{\substack{\mathbf{x}(i) \in \mathcal{S}_u \\ i=1, \dots, n_y}}{\text{argmin}} [f_{\hat{y}(i)}(\mathbf{x}(i)) - f_{\bar{y}(i)}(\mathbf{x}(i))]. \tag{6}$$

Continuing with the argument that it is desirable to select instances which attempt to halve the version space with each instance selection, we design an active learning querying function which performs this on the basis of a per local level prediction. A local classifier which either ignores or is ignorant of the structural constraints maintains a version space described by

$$\mathcal{V}_{\text{local}} = \{f \in \mathcal{F} | f_{y(i)}(\mathbf{x}(i)) > f_{\bar{y}(i)}(\mathbf{x}(i)); \forall (\mathbf{x}, y) \in \mathcal{S}_l\}.$$

If the learning algorithm has access to an inference mechanism that maintains structural consistency, the version space is only dependent on the subset of possible output variable assignments that are consistent with the global structure,

$$\mathcal{V}_{\text{local}(C)} = \{f \in \mathcal{F} | f_{y(i)}(\mathbf{x}(i)) > f_{\bar{y}(i)}(\mathbf{x}(i)); \forall (\mathbf{x}, y) \in \mathcal{S}_l\}$$

where $\bar{y}(i) = \text{argmax}_{y'(i) \in \mathcal{C}(y) \setminus y(i)} f_{y'(i)}(\mathbf{x}(i))$. Therefore, if the learning algorithm enforces structural consistency within the learning model, we advocate also utilizing this information to augment $\mathcal{Q}_{\text{local}}$, resulting in the querying function

$$\mathcal{Q}_{\text{local}(C)} : \mathbf{x}(i)_{\star} = \underset{\substack{\mathbf{x}(i) \in \mathcal{S}_u \\ i=1, \dots, n_y}}{\text{argmin}} [f_{\hat{y}(i)}(\mathbf{x}(i)) - f_{\bar{y}(i)}(\mathbf{x}(i))]$$

While the version space argument provides some theoretical justification for $\mathcal{Q}_{\text{local}(C)}$, there is also a practical argument related to *correction propagation* [23]. Again consider the example shown in Fig. 3, and assume that the classifier is very confident in its alignment predictions for *the* and *fish*. At this point, any single prediction significantly constrains the predictions of the remaining words by reducing the coherent output structures.

In many cases, this will dramatically reduce the feasible output space and thus reduce the need for additional partial queries at the cost of only the inference procedure. We empirically demonstrate in Sect. 5 that when possible, partial queries will often outperform complete queries under the same local prediction driven annotation cost model.

3.4 Active learning with structured perceptron

Now that we have established potentially appropriate querying functions for structured output scenarios, we apply it to learning within the hypothesis space of linear classifiers. This work specifically utilizes classifiers of a linear representation with parameters learned using the structured (Collins) Perceptron algorithm [22]. In this case, $f(\mathbf{x}, \mathbf{y}) = \boldsymbol{\alpha} \cdot \Phi(x, \mathbf{y})$ represents the global scoring function such that $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_{y(1)}, \dots, \boldsymbol{\alpha}_{y(n_y)})$ is a concatenation of the local $\boldsymbol{\alpha}_{y(i)}$ vectors and $\mathbf{x} = \Phi(x, \mathbf{y}) = (\Phi_1(x, \mathbf{y}), \dots, \Phi_{n_y}(x, \mathbf{y}))$ is a concatenation of the local feature vectors, $\mathbf{x}(i) = \Phi_i(x, \mathbf{y})$. Utilizing this notation, $f_{y(i)}(\mathbf{x}(i)) = \boldsymbol{\alpha}_{y(i)} \cdot \Phi_i(x, \mathbf{y})$ where $\boldsymbol{\alpha}_{y(i)} \in \mathbb{R}^{d_i}$ is the learned local weight vector and $\Phi_i(x, \mathbf{y}) \in \mathbb{R}^{d_i}$ is the feature vector for local classifications.

Margin-based active learning generally relies upon the use of support vector machines [72, 78]. While this would naturally extend to existing work on SVM for structured output [73], the incremental nature of active learning over large data sets associated with structured output makes these algorithms impractical for such uses. As in the binary and multiclass scenarios, we require a learning algorithm that can maintain selection batch sizes such that the domain expert is always labeling instances which improve the most current hypothesis. Along this vein, this work builds upon the *inference based training* (IBT) learning strategy [22, 49] shown in Algorithm 1, which incorporates the structural knowledge into the learning procedure. The basic premise of IBT is to make a sequence of local predictions, perform inference to obtain a coherent structure, and then make additive updates relative to the output vector associated with the coherent output.

However, to facilitate use with active learning, we make a few important modifications. We first modify the IBT algorithm for partial labels by updating only local components which have been labeled, thus requiring an inference procedure capable of handling unassigned output variables. Secondly, we add a notion of large margin to the IBT algorithm heuristically by requiring thick separation γ between class activations. Finally, after training is complete, we use the averaged Perceptron [35] method where the final hypothesis is a weighted average of hypotheses used for updates as weighted by their survival counts. This adds stability to the learning procedure and results in a substantial performance improvement.

Algorithm 1 Inference Based Training (IBT) with Thick Separation

1: **Input:** Labeled training sample $\mathcal{S} \in \{\mathcal{X} \times \mathcal{Y}\}^m$, estimated thickness of linear separation γ , number of training rounds T

2: $\alpha \leftarrow 0$ {Initialize weight vector}
3: **for** T iterations **do**
4: **for all** $(\mathbf{x}, y) \in \mathcal{S}$ **do**
5: $\hat{y}_C \leftarrow \operatorname{argmax}_{y \in \mathcal{C}(\mathcal{Y})} \alpha \cdot \Phi(x, \mathbf{y})$
6: **for** $i \leftarrow 1$ to n_y such that $(x, y(i)) \in \mathcal{S}_l$ **do**
7: **if** $\hat{y}_C(i) \neq y(i)$ **then**
8: $\alpha_{y(i)} \leftarrow \alpha_{y(i)} + \Phi_{y(i)}(x, \mathbf{y})$ {Promote weight vector associated with correct label}
9: $\alpha_{\hat{y}(i)} \leftarrow \alpha_{\hat{y}(i)} - \Phi_{\hat{y}(i)}(x, \mathbf{y})$ {Demote weight vector associated with incorrect prediction}
10: **else if** $f_{\hat{y}(i)}(\mathbf{x}(i)) - \gamma < f_{\hat{y}_C(i)}(\mathbf{x}(i))$ **then**
11: $\alpha_{\hat{y}(i)} \leftarrow \alpha_{\hat{y}(i)} + \Phi_{\hat{y}(i)}(x, \mathbf{y})$ {Promote weight vector associated with predicted/correct label}
12: $\alpha_{\hat{y}(i)} \leftarrow \alpha_{\hat{y}(i)} - \Phi_{\hat{y}(i)}(x, \mathbf{y})$ {Demote weight vector associated with second highest activation}
13: **end if**
14: **end for**
15: **end for**
16: **end for**

17: **Output:** Learned hypothesis \hat{h}

3.4.1 Mistake-driven active learning

A greedy criteria for active learning querying functions looks to make the most immediate progress towards learning the target function with each requested label. For the mistake-driven Perceptron algorithm, a suitable measurement for progress is to track the number of additive updates for each query. In the structured setting, this intuition motivates two metrics to explain the performance results of a given querying function, *average Hamming error per query*, $\mathcal{M}_{\text{Hamming}}$, and *average global error per query*, $\mathcal{M}_{\text{global}}$. For a specific round of active learning, the current hypothesis is used to select a set of instances $\mathcal{S}_{\text{select}}$ for labeling. Once the labels are received, we calculate the Hamming loss,

$$\text{Hamming}(h, \mathbf{x}) = \sum_{\substack{i=1 \\ (\mathbf{x}(i), y(i)) \in \mathcal{S}_l}}^{n_y} \mathbb{I}[\hat{y}_C(i) \neq y(i)]$$

and the global loss

$$\mathcal{G}(h, \mathbf{x}) = \mathbb{I}[\hat{y}_C \neq y]$$

at the time when the instance is first labeled.¹ In Sect. 5, we will compare and observe a correlation between the quality of a querying function and the average of the number of updates induced by all queries up to the specified round of active learning. To successfully utilize active learning in a mistake-bound algorithm, we require querying functions which can select examples which update the current hypothesis during each round of active learning.

¹ $\mathbb{I}[p]$ is an indicator function such that $\mathbb{I}[p]$ if p is true and 0 otherwise.

Noting that *Hamming*(h, \mathbf{x}) is only useful for partial labels, we hypothesize that for partial label queries or cases of complete label queries where the data sample \mathcal{S} is largely locally separable, the relative magnitude of $\mathcal{M}_{\text{Hamming}}$ will determine the relative performance of the querying functions as this will increase the number of local updates. Alternatively, for complete queries where a significant portion of the data is exclusively globally separable, $\mathcal{M}_{\text{global}}$ will be more strongly correlated with querying function performance as it will ensure at least one update, relying on the correction propagation principle to constrain the other variables. We support this hypothesis empirically in Sect. 5.

4 Active learning with pipeline models

In addition to structured output scenarios, pipeline models are a second important formalism for successfully applying machine learning approaches to complex problems. In many such cases, there are substantial performance advantages to decomposing the overall task into a sequence of several simpler sequential stages where each stage solves a progressively more difficult problem based on the output of previous stages. Similar to structured output scenarios, a distinguishing feature of applications requiring pipeline models is that they often require significant quantities of labeled data to learn accurately, motivating the study of active learning in such scenarios. This section presents a novel strategy for combining local active learning instance querying strategies into a global strategy which minimizes the annotation requirements for the overall task. Using this method, we present promising empirical results for a three-stage entity and relation

extraction system which significantly reduces supervised data requirements in Sect. 6.

Decomposing complex classification tasks into a series of sequential stages, where the local classifier at a specified stage is explicitly dependent on the predictions from the previous stages, is a common practice in many engineering disciplines. In the machine learning and natural language processing communities, this widely used paradigm is commonly referred to as a *pipeline model* [13, 18, 33]. For example, again consider the named entity extraction (NER) task shown in Fig. 2. In this case, instead of making several local predictions regarding both segmentation and classification for each word and assembling them into a global prediction, a pipeline model would first learn an entity identification (segmentation) classifier and use this as input into an entity labeling classifier, which is then assembled into a two stage pipeline NER system as shown in Fig. 4. More formally, we first want to learn a segmentation classifier where each local prediction is within the output space $\mathcal{Y}_i = \{B, I, O\}$ where, as before, B represents the beginning word of a named entity, I represents an inside word of a named entity, O represents a word outside any named entity, and an I label can only follow a B label. In addition to our segmentation classifier, we also learn a classifier that makes predictions over already segmented text in the output space $\mathcal{Y}_i = \{\text{PEOPLE, LOCATION, ORGANIZATION}\}$. Furthermore, these stages may be preceded by other simpler related natural language processing tasks such as part of speech (POS) tagging or word sense disambiguation (WSD).

The primary motivation for modeling complex tasks as a pipelined process is the difficulty of solving such applications with a single monolithic classifier; that expressing a problem such as relation extraction directly in terms of unprocessed input text will result in a complex function that may be impossible to learn with reasonable labeled data constraints. A second relevant aspect of such domains

is the corresponding high cost associated with obtaining sufficient labeled data for good learning performance. As previously stated, active learning often mitigates this dilemma by allowing the learning algorithm to incrementally select unlabeled examples for labeling by the domain expert with the goal of maximizing performance while minimizing the labeling effort [21]. This section extends active learning for a single classifier by assuming that an active learning querying function exists for each stage of a pipelined learning model and develops a strategy that jointly minimizes the annotation requirements for the pipelined process.

This section presents a general method for combining separate active learning strategies from multiple pipelined stages into a single strategy that exploits properties particular to pipeline models. Specifically, we propose a criteria that begins by preferring instances which most benefit early pipeline stages until they are performing sufficiently well, at which point instances are selected which target later stages of the pipeline. This method attempts to reduce error propagation and supply all pipeline stages with sufficiently error free input for effective learning. Furthermore, we instantiate this method for a three stage named entity and relation extraction system in Sect. 6, demonstrating significant reductions in annotation requirements.

4.1 Learning pipeline models

Our pipeline model formulation specifically utilizes classifiers based upon a feature vector generating procedure $\Phi(x) \rightarrow \mathbf{x}$ and generates the output assignment using a scoring function $f : \Phi(\mathcal{X}) \times \mathcal{Y} \rightarrow \mathbb{R}$ such that the prediction is stated as $\hat{y} = h(x) = \operatorname{argmax}_{y' \in \mathcal{Y}} f_{y'}(\mathbf{x})$. In a pipeline model, each stage $j = 1, \dots, J$ has access to the input instance in addition to the classifications from all previous stages, $\Phi^{(j)}(x, \hat{y}^{(0)}, \dots, \hat{y}^{(j-1)}) \rightarrow \mathbf{x}^{(j)}$. Each stage of a pipelined learning process takes m training instances $\mathcal{S}^{(j)} = \{(\mathbf{x}_1^{(j)}, y_1^{(j)}), \dots, (\mathbf{x}_m^{(j)}, y_m^{(j)})\}$ as input to a learning algorithm $\mathcal{A}^{(j)}$ and returns a classifier, $h^{(j)}$, which minimizes the respective loss function of the j th stage. Note that each stage may vary in complexity from a single binary prediction, $y^{(j)} \in \{-1, 1\}$, to a multiclass prediction, $y^{(j)} \in \{\omega_1, \dots, \omega_k\}$, to a structured output prediction, $y^{(j)} \in \mathcal{Y}_1^{(j)} \times \dots \times \mathcal{Y}_{n_y}^{(j)}$. Once each stage of the pipeline model classifier is learned, global predictions are made sequentially with the expressed goal of maximizing performance on the overall task,

$$\hat{\mathbf{y}} = h(x) = \left\langle \operatorname{argmax}_{y' \in \mathcal{Y}^{(j)}} f_{y'}^{(j)}(\mathbf{x}^{(j)}) \right\rangle_{j=1}^J. \tag{7}$$

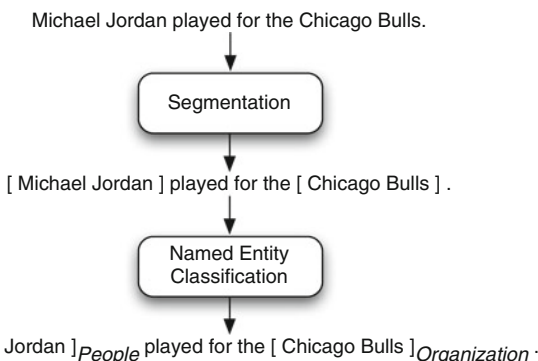


Fig. 4 Pipeline model for named entity recognition (NER)

4.2 Querying functions for pipelines

As previously stated, the key difference between active learning and standard supervised learning is a querying function, Q , which when provided with the data S and the learned classifier h returns a set of unlabeled instances $S_{\text{select}} \subseteq S_u$. These selected instances are labeled and added to the supervised training set S_l used to update the learned hypothesis. This work requires that all querying functions (one for each stage) determine instance selection using an underlying *query scoring function* $Q : x \rightarrow \mathbb{R}$ such that instances with smaller scoring function values are selected,

$$Q : x_{\star} = \operatorname{argmin}_{x \in S_u} Q(x). \tag{8}$$

For notational convenience, we assume that the query scoring function only requires the instance x to return a score and implicitly has access to facilities required to make this determination (e.g. f, h, Φ , properties of \mathcal{Y} , etc.). Furthermore, in the pipeline setting, we enforce that each $Q^{(j)}$ be of similar range and shape such that the values may be effectively compared and combined. This restriction is not difficult to obey in practice as the classifiers at each stage generally use similar features and classification algorithms; however, there may be circumstances where this would be a technical issue requiring further investigation.

Given a pipeline model and a query scoring function for each stage of the pipeline, $Q^{(j)}$, this work develops a general strategy for combining local query scoring functions into a joint querying function for the global pipeline task of the form

$$Q_{\text{pipeline}} : x_{\star} = \operatorname{argmin}_{x \in S_u} \sum_{j=1}^J \beta^{(j)} \cdot Q^{(j)}(x). \tag{9}$$

Based upon this formulation, the goal is to set the values of β_t for each querying phase of the active learning protocol by exploiting properties of pipeline models. By varying the relative magnitude of each component of β , we are able to alter the relative influence of the query scoring function, thereby emphasizing different stages throughout the procedure as appropriate. Some observed properties of a well designed pipeline which most strongly affect selecting values for β_t include the following desiderata:

1. The results of earlier stages are useful, and often necessary, for later stages [18].
2. Earlier stages should be easier to learn than later stages [56].
3. Errors from early stages will propagate to later stages.

To design a global querying function for such architectures, examination of the pipeline model assumptions is required. Given a sequence of pipelined functions, the idealized global prediction function for a pipeline model is stated by

$$\hat{y} = \operatorname{argmax}_{y' \in \mathcal{Y}^{(1)} \times \dots \times \mathcal{Y}^{(J)}} \sum_{j=1}^J \pi^{(j)} \cdot f_{y^{(j)'}}^{(j)}(\mathbf{x}^{(j)}) \tag{10}$$

where π is used to determine the relative importance associated with correct predictions for each stage of the pipeline, noting that in most cases $\pi = [0, \dots, 0, 1]$ and the goal is to maximize the performance exclusively in regards to the final stage. Comparing Equation 10 to the pipelined prediction function of Eq. 7, we see that the pipeline model assumption is essentially that the learned function for each stage abstracts sufficient information such that each stage can be learned independently and only the predictions are required to propagate information between stages as opposed to a joint learning model. Naturally, this alleviates the need to predict joint output vectors with interdependent variables and will result in a much lower sample complexity if the assumption is true. However, to satisfy the pipeline model assumption, we first observe that each stage j possesses a certain degree of robustness to noise from the input $\Phi^{(j)}(x, \hat{y}^{(0)}, \dots, \hat{y}^{(j-1)})$. If this tolerance is exceeded, stage j will no longer make reliable predictions and will lead to errors cascading to later stages. This notion results in the primary criteria for designing a querying function for pipeline models, that early stages must be performing sufficiently well before later stages influence the combined querying function decision. Therefore, the global querying function should possess the following properties:

1. Early stages should be emphasized for earlier iterations of active learning, ideally until learned perfectly.
2. Significant performance improvement at stage j implies that stages $1, \dots, (j - 1)$ are performing sufficiently well and stage j should be emphasized.
3. Conversely, lack of performance improvement at stage j either implies that stages $1, \dots, (j - 1)$ are not performing well and should be emphasized by the querying function or stage j has plateaued in performance and later stages should be emphasized.

The first criteria is trivial to satisfy by setting $\beta_0 = [1, 0, \dots, 0]$. The remaining criteria are more difficult as an estimate of querying function performance at each stage is required to update β without labeled data for cross-validation. Based upon work regarding the *early stopping* of active learning algorithms, [31] prescribes such a procedure

in the context of determining crossover points with querying functions specifically suitable for two different operating regions of the active learning protocol for a single binary prediction. This method calculates the average expected error over \mathcal{S}_u after each iteration,

$$\hat{\epsilon} = \frac{\sum_{x \in \mathcal{S}_u} E[(\hat{y} - y)^2 | x]}{|\mathcal{S}_u|}$$

where

$$E[(\hat{y} - y)^2 | x] = \sum_{y \in \mathcal{Y}} \mathcal{L}_{0/1}(\hat{y}, y) P(y | x) \tag{11}$$

and $\mathcal{L}_{0/1}$ is the 0/1 loss function. Once the change in expected error is small from the current round t to the next round $t + 1$ for a specified prediction, $\Delta \hat{\epsilon} < \delta$, the current configuration is deemed to be achieving diminishing returns and the second querying function more appropriate for different operating regions should be used. By using this method, the appropriate querying functions are used for each operating range, thus improving the overall result of active learning.

This work derives an analogous method in the context of pipeline models, where operating regions correspond to the segment of the pipeline being emphasized when querying instances. The first observation is that we cannot directly extend the aforementioned procedure and maintain sufficient generality as the loss function at each stage is not necessarily $\mathcal{L}_{0/1}$ and it is difficult to accurately estimate $P(y|x)$ for the arbitrarily complex classifiers comprising each stage, in contrast with the earlier work discussed above. Furthermore, relatively precise knowledge of these parameters is required to reasonably specify $\delta^{(j)}$ (i.e. the crossover criterion for different operating regions). However, an important observation is that Eq. 11 is their query scoring function which we generalize to basing the method for determining crossover points on the average of the query scoring function over the unlabeled data,

$$U_t^{(j)} = \frac{\sum_{x \in \mathcal{S}_u} Q^{(j)}(x)}{|\mathcal{S}_u|}.$$

The intuition is that $U_t^{(j)}$ represents the certainty of $f^{(j)}$ for each iteration of active learning and once this value stops increasing between iterations, $Q^{(j)}$ is likely entering an operating region of diminishing returns and should be discounted. However, since δ would be difficult to calibrate for multiple stages and irrevocable crossover points would be undesirable in the pipeline model case, we opt for an algorithm where each stage competes with other stages for relative influence on the global querying function of Eq. 9 based on the relative value changes in $U^{(j)}$. This reasoning leads to Algorithm 2.

Algorithm 2 Active Learning for Pipeline Models

```

1: Input:  $\mathcal{S}_l, \mathcal{S}_u, \{\mathcal{A}^{(j)}\}_{j=1}^J, \{Q^{(j)}\}_{j=1}^J, \lambda, \mathcal{K}$ 
2: for  $j \leftarrow 1, \dots, J$  do {initialize algorithm}
3:    $h_0^{(j)} \leftarrow \mathcal{A}^{(j)}(\mathcal{S}_l)$ 
4:    $U_0^{(j)} \leftarrow \frac{\sum_{x \in \mathcal{S}_u} Q^{(j)}(x)}{|\mathcal{S}_u|}$ 
5:    $\beta_0^{(j)} \leftarrow 0$ 
6: end for
7:  $\beta_0^{(1)} \leftarrow 1$ 
8:  $t \leftarrow 1$ 
9: while  $\mathcal{P}(\hat{h}) < \mathcal{K}$  do {query new examples}
10:   $\mathcal{S}_{select} \leftarrow \operatorname{argmin}_{x \in \mathcal{S}_u} \sum_{j=1}^J \beta_{t-1}^{(j)} \cdot Q^{(j)}(x)$ 
11:   $\mathcal{S}_u \leftarrow \mathcal{S}_u \setminus \mathcal{S}_{select}$ 
12:   $\mathcal{S}_l \leftarrow \mathcal{S}_l \cup \mathcal{S}_{select}$  {expert labels  $\mathcal{S}_{select}$ }
13: for  $j \leftarrow 1, \dots, D$  do {update hypothesis}
14:    $h_t^{(j)} \leftarrow \mathcal{A}^{(j)}(\mathcal{S}_l)$ 
15:    $U_t^{(j)} \leftarrow \frac{\sum_{x \in \mathcal{S}_u} Q^{(j)}(x)}{|\mathcal{S}_u|}$ 
16:    $\Delta_t^{(j)} \leftarrow U_t^{(j)} - U_{t-1}^{(j)}$ 
17: end for
18:  $\Delta_t \leftarrow \frac{\Delta_t}{\|\Delta_t\|}$ 
19: for  $j \leftarrow 1, \dots, J$  do {update  $\beta$ }
20:    $\beta_t^{(j)} \leftarrow \beta_{t-1}^{(j)} + \lambda \cdot \Delta_t^{(j)}$ 
21: end for
22:  $\beta_t \leftarrow \frac{\beta_t}{\|\beta_t\|}$ 
23:  $t \leftarrow t + 1$ 
24: end while
25: Output:  $\{h^{(j)}\}_{j=1}^J$ 

```

Algorithm 2 begins by taking as input the seed labeled data \mathcal{S}_l , unlabeled data \mathcal{S}_u , the learning algorithm for each stage $\mathcal{A}^{(j)}$, a query scoring function for each stage $Q^{(j)}$, an update rate parameter λ , and an active learning stopping criteria \mathcal{K} . Lines 2–8 initialize the algorithm by learning an initial hypothesis $h_0^{(j)}$ for each stage, calculating the initial average query scoring function value $U_0^{(j)}$ for each stage, and setting $\beta_0 = [1, 0, \dots, 0]$. Line 9 checks if active learning stopping criteria has been met. If not, lines 10–12 select instances \mathcal{S}_{select} according to the current β which are removed from \mathcal{S}_u , labeled, and added to \mathcal{S}_l . Lines 13–17 update the hypothesis for each stage and calculate the new values of $U_t^{(j)}$ for each stage. After Δ_t is normalized (line 18), we update the value of $\beta_t^{(j)}$ for each stage based on the relative improvements of $U_t^{(j)}$. Finally, β is normalized (line 22) and the process is repeated. Fundamentally, based upon earlier stated principles, Algorithm 2 assumes that $\beta = [1, 0, \dots, 0]$ is the optimal mixing parameter at $t = 0$ and tracks this non-stationary parameter over t based on the feedback provided by $(U_t^{(j)} - U_{t-1}^{(j)})$ at line 16 to set the values of β for each round of active learning with pipeline models. We empirically validate this algorithm on a three-stage entity and relation extraction system in Sect. 6.

Fig. 5 Constraints used for generating synthetic data

1. $\mathcal{C}_1 : [h_2(\mathbf{x}) \neq \omega_3] \wedge [h_5(\mathbf{x}) \neq \omega_1]$
2. $\mathcal{C}_2 : \text{At most one } h_i(\mathbf{x}) \text{ can output } \omega_2.$
3. $\mathcal{C}_3 : \text{For one or more } h_i(\mathbf{x}) \text{ to output } \omega_3, \text{ at least one } h_i(\mathbf{x}) \text{ must output } \omega_1.$
4. $\mathcal{C}_4 : h_i(\mathbf{x}(i)) \text{ can output } \omega_4 \text{ if and only if } h_{i-1}(\mathbf{x}) = \omega_1 \text{ and } h_{i-2}(\mathbf{x}) = \omega_2.$

5 Empirical results for structured output spaces

We demonstrate particular properties of the proposed querying functions of Sect. 3 by first conducting several active learning simulations on synthetic data each intended to elucidate performance characteristics on different structured learning scenarios. We then verify practical use for actual applications by performing experiments on a restricted for of the semantic role labeling (SRL) task.²

5.1 Synthetic data

Our synthetic structured output problem is comprised of five multiclass classifiers, h_1, \dots, h_5 , each with the output space $Y_i = \omega_1, \dots, \omega_4$. In addition, we define the output structure using the practical constraints shown in Fig. 5. To generate the synthetic data, we first create four linear functions of the form $\alpha_{y'(i)} \cdot \mathbf{x}(i) + b_{y'(i)}$ such that $\alpha_{y'(i)} \in [-1, 1]^{100}$ and $b_{y'(i)} \in [-1, 1]$ are generated at random from a uniform distribution for each $h_{y'(i)}$ where $y'(i) \in \{\omega_1, \dots, \omega_4\}$. These weight vectors are held constant throughout the data generation process. To generate data, we first produce five local examples $\mathbf{x}(i) \in \{0, 1\}^{100}$ where the normal distribution $\mathcal{N}(20, 5)$ determines the number of features assigned the value 1, distributed uniformly over the feature vector. Each vector is then labeled according to the function $\text{argmax}_{i=1, \dots, k} [\alpha_{y(i)} \cdot \mathbf{x} + b_{y(i)}]$ resulting in the label vector $\mathbf{y}_{\text{local}} = (h_1(\mathbf{x}(1)), \dots, h_5(\mathbf{x}(5)))$. We then use the inference procedure to obtain the final labeling y of the instance \mathbf{x} to ensure that the output $y \in \mathcal{C}(\mathcal{Y})$. If $\hat{y}_{\mathcal{C}} \neq \hat{y}$, then the data is exclusively globally separable. We control the total amount of such data with the parameter κ which represents the fraction of exclusively globally separable data in \mathcal{S} . We further filter the difficulty of the data such that all exclusively globally separable instances have a Hamming error drawn from a stated normal distribution $\mathcal{N}(\mu, \sigma)$. We generate 10000 structured instances for each configuration, or equivalently 50000 local instances, in this for each set of data parameters we use. This process is outlined in Fig. 6.

² Empirical discrepancies between the performance reported in this work and that of [54] is accounted for by the use of averaged Perceptron and smaller batch sizes during instance selection.

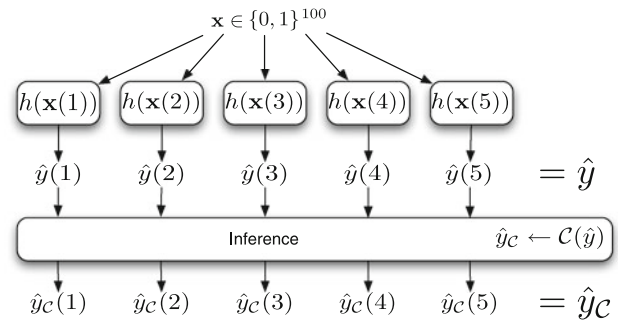


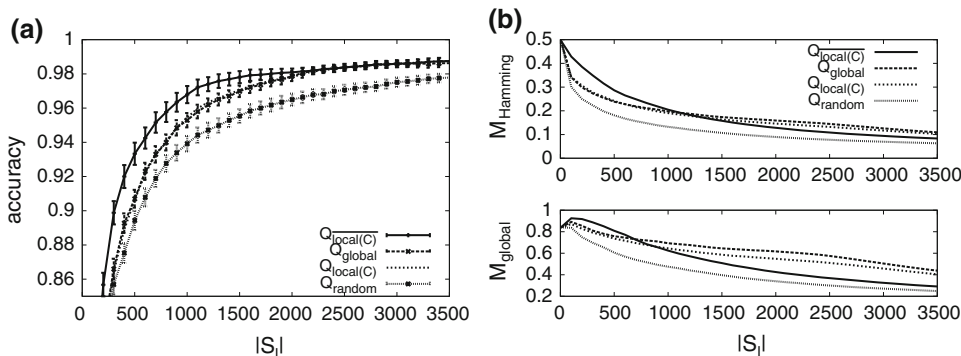
Fig. 6 Procedure used to generate synthetic data

When conducting complete label active learning experiments with synthetic data, we report results for $\mathcal{Q}_{\text{local}(\mathcal{C})}$, the querying function based upon the averaging of local output predictions, $\mathcal{Q}_{\text{global}}$, the querying function based upon the global output scores, $\mathcal{Q}_{\text{local}(\mathcal{C})}$, the querying function which bases its score on the output of the single local prediction of minimum certainty, and $\mathcal{Q}_{\text{random}}$, the querying function which selects instances randomly from a uniform distribution over the unlabeled data \mathcal{S}_u at each step. For experiments using complete queries, we use a querying schedule which begins with five labeled examples and selects five instances during each round of active learning, $|\mathcal{S}_l| = 5, 10, \dots, 8000$. For all synthetic experiments, $T = 7$ and $\gamma = 0.5$ for Algorithm 1 and fivefold cross validation is performed with error bars indicating 95% confidence intervals.

5.1.1 Complete queries with locally separable data

The first case we consider for complete queries is where $\kappa = 0$, the situation where the data is completely locally learnable and the constraints are not necessary to learn the target function with linear classifiers as shown in Fig. 7. As hypothesized, $\mathcal{Q}_{\text{local}(\mathcal{C})}$ performs better in this setting than $\mathcal{Q}_{\text{global}}$. By using $\mathcal{Q}_{\text{local}(\mathcal{C})}$, we achieve a performance level equivalent to training on all examples at $|\mathcal{S}_l| \approx 1500$, which is approximately a 81% reduction in labeled data. When using $\mathcal{Q}_{\text{global}}$, we achieve a labeled data reduction of approximately 73%. This difference is also reflected in the plot of $\mathcal{M}_{\text{Hamming}}$, where $\mathcal{Q}_{\text{local}(\mathcal{C})}$ induces significantly more local updates on average. Finally, we note that

Fig. 7 Complete label querying for $\kappa = 0.0$. **a** Relative performance of complete label querying functions. **b** Relative values of $\mathcal{M}_{\text{Hamming}}$ and $\mathcal{M}_{\text{global}}$ for complete label querying functions



$Q_{\text{local}(C)}$ performs nearly identically to Q_{global} as the differences between two structures in this scenario often will rely on a single local prediction. Note that we omitted error bars for $Q_{\text{local}(C)}$ for clarity as they were very similar to Q_{global} .

5.1.2 Complete queries with exclusively globally separable data

The second case we consider is the setting where $\kappa = 0.3$ and the Hamming error of the generated data is drawn from $\mathcal{N}(3, 1)$, with results shown in Fig. 8. For this case, the output space is much more heavily constrained and as hypothesized, Q_{global} works better than $Q_{\text{local}(C)}$. By using Q_{global} , we achieve a performance level equivalent to training on all 8000 examples at $S_l \approx 3000$, which is approximately a 63% reduction in labeled data requirements. Conversely, when using $Q_{\text{local}(C)}$, we achieve a labeled data reduction of approximately 25% (not shown on this graph). This difference is also reflected in the plot of $\mathcal{M}_{\text{global}}$, relative to $\mathcal{M}_{\text{global}}$ where $Q_{\text{local}(C)}$ may induce more local updates on average, but the number of global updates does not correspond accordingly. This implies that $Q_{\text{local}(C)}$ is making local update which do not affect the global prediction. Note that we omitted error bars for $Q_{\text{local}(C)}$ for clarity as its performance is clearly statistically worse than Q_{global} .

5.1.3 Partial queries on synthetic data

We continue our study with synthetic data by conducting experiments which make partial queries during each round of active learning. For the partial query setting, we report results using the two partial querying functions Q_{local} and $Q_{\text{local}(C)}$ in addition to Q_{random} on three sets of data. For partial queries, the querying schedule starts by querying 10 partial labels at a time from $|S_l| = 10, 20, \dots, 40000$, once

again performing fivefold cross validation to demonstrate statistical significance with 95% confidence intervals.

The first data set for partial queries is when $\kappa = 0.0$ and the data is completely locally separable as in our experiment for complete queries. The results for this configuration are displayed in Fig. 9, demonstrating a savings in labeled data requirements of approximately 80% for $Q_{\text{local}(C)}$. In this case, active learning for both Q_{local} and $Q_{\text{local}(C)}$ perform better than Q_{random} . Somewhat more surprising is the result that $Q_{\text{local}(C)}$ performs noticeably better than Q_{local} even though they often query similar points for $\kappa = 0.0$. However, we hypothesize is that when there is only a small quantity of labeled data, the constraints provide additional information which guides the learned hypothesis toward the target hypothesis by querying instances which will make more updates to the current hypothesis. We see this phenomena to some degree in the plot of $\mathcal{M}_{\text{Hamming}}$ as $Q_{\text{local}(C)}$ results in more updates in the IBT algorithm than Q_{local} .

The second partial querying experiment we conduct is for the synthetic data set $\kappa = 0.3; \mathcal{N}(3, 1)$ as shown in Fig. 10. This configuration demonstrates a similar performance ordering as when $\kappa = 0.0$, where $Q_{\text{local}(C)}$ outperforms Q_{local} which in turn outperforms Q_{random} with $Q_{\text{local}(C)}$ achieving a label savings of approximately 65%. Furthermore, early in the active learning process, we see that Q_{local} performs worse than Q_{random} . The plot of the number of updates in combination with this fact implies that the number of updates is not the sole indicator of performance, but that when constraints are used, the selected instances must make updates as targeted by the correct version space.

Finally, we perform a partial querying experiment with synthetic data set where $\kappa = 1.0; \mathcal{N}(5, 1)$, meaning that the data is completely exclusively globally separable and the difference between $Q_{\text{local}(C)}$ and Q_{local} will be different at all predictions for most examples. These results are shown in Fig. 11. In this configuration, $Q_{\text{local}(C)}$ performs significantly better than Q_{local} demonstrating the benefit of

Fig. 8 Complete label querying for $\kappa = 0.3$ and Hamming error drawn from $\mathcal{N}(3, 1)$. **a** Relative performance of complete label querying functions. **b** Relative values of $\mathcal{M}_{\text{Hamming}}$ and $\mathcal{M}_{\text{global}}$ for complete label querying functions

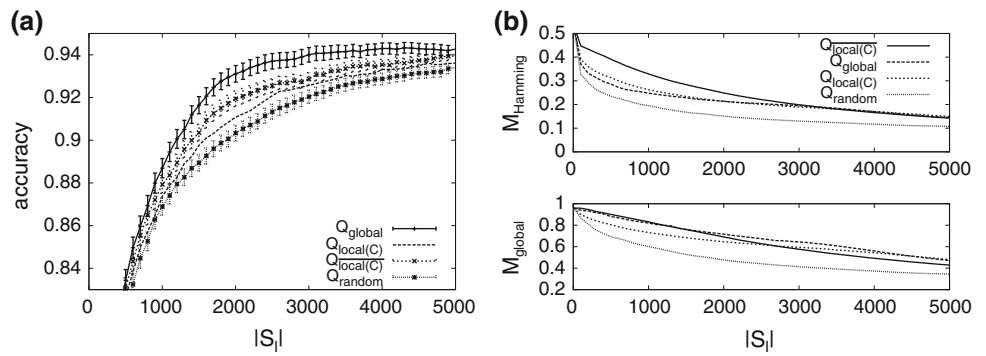


Fig. 9 Partial label querying for $\kappa = 0.0$. **a** Relative performance of partial label querying functions. **b** Relative values of $\mathcal{M}_{\text{Hamming}}$ for partial label querying functions

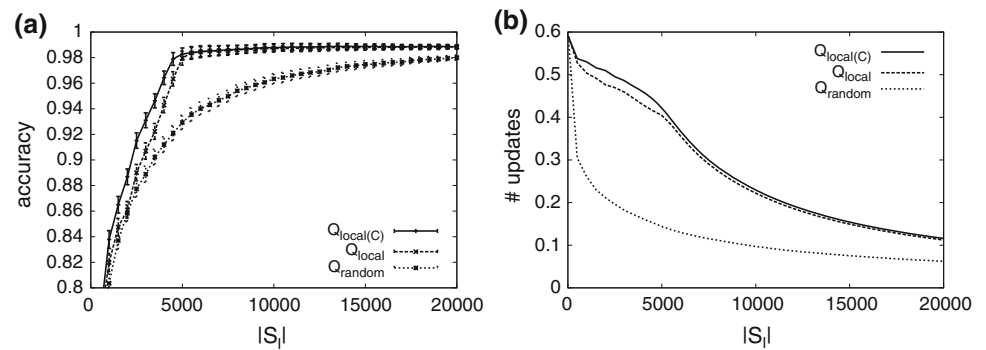


Fig. 10 Partial label querying for $\kappa = 0.3$ and Hamming error drawn from $\mathcal{N}(3, 1)$. **a** Relative performance of partial label querying functions. **b** Relative values of $\mathcal{M}_{\text{Hamming}}$ for partial label querying functions

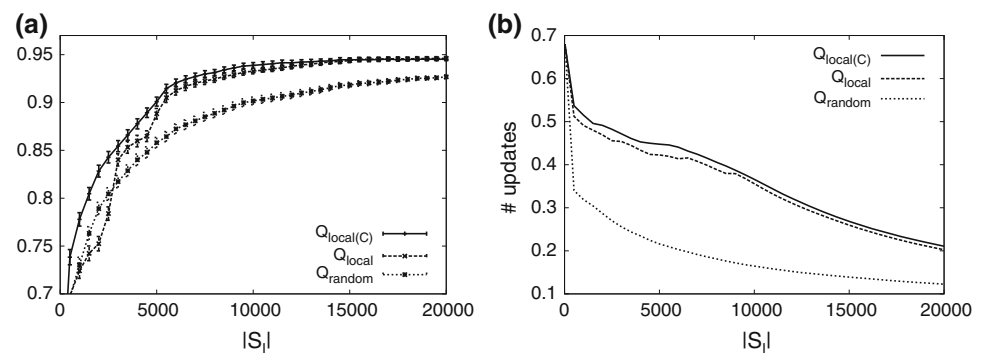
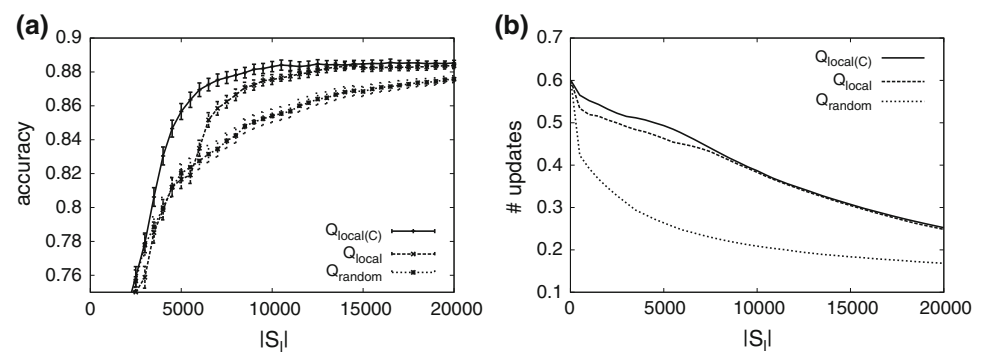


Fig. 11 Partial label querying for $\kappa = 1.0$ and Hamming error drawn from $\mathcal{N}(5, 1)$. **a** Relative performance of partial label querying functions. **b** Relative values of $\mathcal{M}_{\text{Hamming}}$ for partial label querying functions



considering the structural constraints when requesting labels for unlabeled instances. Specifically, $Q_{\text{local}(C)}$ achieves a reduction in labeled data requirements of approximately 75% while Q_{local} achieves a savings of approximately 63%, although this actually understates the

case as $Q_{\text{local}(C)}$ performs much better at earlier data points. In this case, we once again observe that although the learning algorithm is making many updates, they are seemingly not updates which assist the learning algorithm in learning the target hypothesis.

5.2 Semantic role labeling

As a practical application, we also perform experiments on the (SRL) task as described in the CoNLL-2004 shared task [15]. For SRL, the goal is given a sentence to identify for each verb in the sentence which constituents fill a semantic role and determine the type of the specified argument as shown in Fig. 12. For this particular example, A0 represents the *leaver*, A1 represents the *item left*, A2 represents the *benefactor*, and AM-LOC is an adjunct indicating the location where the action occurs. Examples of specifying structural relationships to ensure coherence include declarative statements such as *every sentence must contain exactly one verb, certain arguments may only attach to specific verbs, or no arguments may overlap*.

To model this problem, we essentially follow the model described in [49] where linear classifiers f_{A0}, f_{A1}, \dots are used to map constituent candidates to one of 45 different classes. For a given argument/predicate pair, the multiclass classifier returns a set of scores which are used to produce the output \hat{y}_c consistent with the structural constraints associated with other arguments relative to the same predicate. We simplify the task by assuming that the constituent boundaries are given, making this an argument classification task. We use the CoNLL-2004 shared task data, but restrict our experiments to sentences that have greater than five arguments to increase the number of instances with interdependent variables and take a random subset of this to get 1500 structured examples comprised of 9327 local predictions. For our testing data, we also restrict ourself to sentences with greater than five arguments, resulting in 301 structured instances comprised of 1862 local predictions. We use the features in Table 1 and the applicable subset of families of constraints which do not concern segmentation shown in Fig. 13 as described by [50].

Figure 14a shows the empirical results for the SRL experiments when querying complete labels. For the complete labels querying scenario, we start with a querying schedule of $|S_l| = 50, 80, \dots, 150$ and slowly increase the step size until ending with $|S_l| = 1000, 1100, \dots, 1500$. When using Algorithm 1, we set $\gamma = 1.0$ and $T = 5$. In this setting, we observe that $Q_{\text{local}(\mathcal{C})}$ performs better than Q_{global} , implying that the data is largely locally separable which is consistent with the findings of [49] for this high dimensional feature space. Essentially, when using such a

high dimensional feature space, the efforts of the querying function are best spent inducing as many local updates as possible as the performance of the local classifiers independent of the global constraints will most significantly affect performance. However, we observe that both $Q_{\text{local}(\mathcal{C})}$ and Q_{global} perform better than Q_{random} with approximately a 35% reduction in labeling effort requirements with $Q_{\text{local}(\mathcal{C})}$.

For partial labels, we used a similar experimental setup with a querying schedule that starts at $|S_l| = 100, 200, \dots, 500$ and increases step size until ending at $|S_l| = 6000, 7000, \dots, 9327$. In this case, $Q_{\text{local}(\mathcal{C})}$ performs better than Q_{local} and Q_{random} , requiring approximately 45% of the data to be labeled. While both $Q_{\text{local}(\mathcal{C})}$ and Q_{local} performs better than Q_{random} , $Q_{\text{local}(\mathcal{C})}$ likely selects instance components for which the constraints do not provides sufficient information to make a confident prediction and thus is using information based upon the hypothesis space of the target function. A final observation is that $Q_{\text{local}(\mathcal{C})}$ requires approximately 4000 instances to achieve the performance of the final state of Q_{random} . Conservatively estimating all instances to have exactly six components, this corresponds to ~ 670 complete instances, which we observe in Fig. 14b is not the final performance level. This reaffirms our contention that partial queries should be used whenever possible and a realistic cost model can be specified.

6 Experiments on a three-stage pipelined entity and relation extraction system

The experimental setting we explore with our pipeline active learning protocol is the three-stage entity and relation extraction system shown in Fig. 15. For each pipeline stage, sentences comprise the instance space of the learning problem which when selected are labeled for all pipeline stages. Secondly, each stage requires multiple predictions, thereby being a structured prediction problem for which we follow the active learning framework for structured output scenarios as presented in Sect. 3. Let $\mathbf{x} \in \mathbf{X}_1 \times \dots \times \mathbf{X}_{n_x}$ represent an input instance and $\mathbf{y} \in \mathcal{C}(\mathcal{Y})$ represent a structured assignment in the space of output variables $\mathbf{Y}_1 \times \dots \times \mathbf{Y}_{n_y}$. \mathcal{C} represents a set of constraints that enforces structural consistency on y , making the prediction function $\hat{y}_c = h(x) = \text{argmax}_{y' \in \mathcal{C}(\mathcal{Y})} f(\mathbf{x}, y')$ for the structured output space.

While active learning often relies upon the use of complex algorithms with high running times such as support vector machines [72] or conditional random fields [23, 64], Sect. 5 demonstrated good results with a regularized version of the structured Perceptron algorithm [22]. In

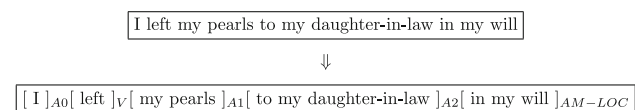


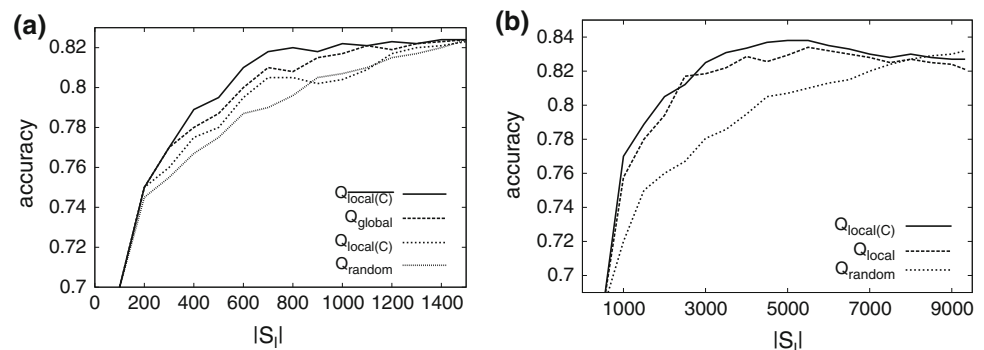
Fig. 12 Semantic role labeling (SRL)

Table 1 Features for semantic role labeling classifier

Name	Description
Predicate lemma and POS tag	A conjunction of the lemma and POS tag of predicate
Voice	Indicates voice (active/passive) of predicate
Position	Indicates if the target argument is before or after the predicate
Clause position	Indicates position of the target argument relative to the predicate in the parse tree
Clause path	The path formed by the parse tree containing only clauses and chunks
Chunk pattern	Encodes the sequence of chunks from the current argument to the predicate
Word and POS tag	Word and POS tags of the first word, last word, and head word
Named entity	Indicates if any element of the predicate is an element of a named entity
Chunk	Indicates if any element of the predicate is an element of a noun phrase
Length	The length of the target phrase both in number of words and number of chunks
Verb class	The class of the active predicate described in the frame files
Phrase type	The target argument phrase type (e.g. VP, PP, NP)
Sub-categorization	Sequence of phrase types of predicate chunk and segments around the predicate
Baseline	Features based on heuristics for identifying AM-NEG and AM-LOC
Clause coverage	Indicates how much of the predicate clause is covered by the argument
Chunk pattern length	Counts the number of chunk patterns in the argument phrase
Conjunctions	A pairwise conjunction of all pairs of features above
Boundary words and POS tags	The words and tags within a window of size 2
Bigrams	Bigrams of words and POS tags within a window of size 2
Sparse collocation	A sequence of one word/tag from window of two before argument, first word/tag of argument, last word/tag of argument, and one word/tag from window of two after argument

Fig. 13 Constraints for semantic role labeling classifier

1. C_1 : At most one argument may output $\{A0, A1, A2, A3, A4, A5\}$.
2. C_2 : Exactly one argument must output V.
3. C_3 : For a classifier to output C-V, it must be preceded by V, A1.
4. C_4 : For a classifier to output R-XXX, a different classifier must output XXX.
5. C_5 : For a classifier to output C-XXX, a preceding classifier must output XXX.
6. C_6 : Given the predicate, some argument classes are illegal as given by the PropBank Frames.

Fig. 14 **a** Active learning for semantic role labeling with complete label queries. **b** Active learning for semantic role labeling with partial label queries

designing an active learning algorithm with complete queries for the stated system, the learning algorithm for each stage, $\mathcal{A}^{(j)}$, is an instance of the IBT algorithm as

described by Algorithm 1. As a discriminative framework, performance is strongly correlated to the quality of the feature vector generating procedure $\Phi^{(j)}$. We extract

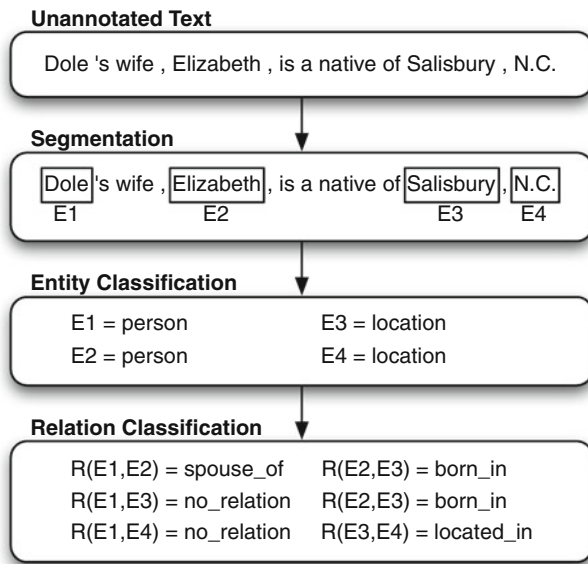


Fig. 15 A three-stage pipeline model for named entity and relation extraction

features in a method similar to [57] except segmentation is not assumed, but the first stage in our pipeline. For segmentation, each target word and its context extracts a feature set including words from a window of size 3 on each side of the target, bigrams within a window of size 2, and the capitalization of words within a window of size one. Furthermore, we check if either of the previous two words have membership in a list of male and female names taken from U.S. census data. Finally for segmentation, we also check membership in a list of months, days, and cities compiled in advance. These features are summarized by Table 2. For entity classification, we extract features including the words of the segment, words within a window of size 2 around the segment, the segment length, and a capitalization pattern. Secondly, we check if any word is in a list of cities, countries, names, and professional titles compiled in advance. These features are summarized in Table 3. Finally, for relation classification, we first extract a conjunction of the features used for the two entities, the labels of the two entities, the length the entities, the distance between them, and membership in a set of extraction patterns [57] (e.g. $\Phi_{arg1,prof,arg2}$ (CNN reporter David McKinley) = 1) as summarized in Table 4.

6.1 Active entity and relation extraction

As stated, this formulation for active learning with pipeline models requires that each stage of the pipeline has a pre-defined query scoring function $Q^{(j)}$. To design $Q^{(j)}$ for each stage of our system, we build upon the active learning for structured output as described in Sect. 3, based upon the

Table 2 Feature generation functions for segmentation

Name	Description
Word	The lowercase version of the target word
Context words	Lowercase version of words within a window of size 3 from the target word
Context bigrams	Lowercase version of bigrams within a window of size 2 from target word
Capitalized	Returns if target word is capitalized
Capitalized context	Returns in words within window of size 1 are capitalized
Male name	Checks if the target word is a male name (from census data)
Female name	Checks if the target word is a female name (from census data)
Month	Checks if the target word is a month
Day	Checks if the target word is a day
City	Checks if the target word is a city
Context names	Checks if previous two words are members of male name or female name

Table 3 Feature generation functions for named entity recognition

Name	Description
Word	The lowercase version of the word in the target segment
Phrase	The n -gram of lowercase words in the segment
Capitalization	Capitalization pattern of phrase (e.g. Duke of Earl \rightarrow XxX)
Length	Target segment length
Context word	Words surrounding target segment within window size 2
Male name	Checks if any target word is a male name (from census data)
Female name	Checks if any target word is a female name (from census data)
Country	Checks if any target word is a country
Title	Checks if any target word is a professional title
City	Checks if the any target word is a city

Table 4 Feature generation functions for relation extraction

Name	Description
Entity Features	A position sensitive conjunction of all features from Table 3
Entity labels	The labels of the two features in sequence
Distance	The number of words between the two entities in a sentence
Pattern	Extraction patterns taken from [57]

decomposition of structured predictions into a vector of multiclass predictions and deriving active learning querying functions based upon the expected multiclass margin.

As previously stated, to extend $\mathcal{Q}_{\text{multiclass}}$ to structured predictions, we must consider the types of predictions made by each stage of the pipeline. For segmentation, the local scoring function f_{segment} outputs an estimate of $P(y | x_i)$ for each word in the input sentence over $\mathcal{Y} \in \{B, I, O\}$. The constraints \mathcal{C} enforce a valid structure by ensuring that *inside* only follows a *begin* label for BIO segmentation. We follow the principles for locally learnable instances and use a variant of the average margin where we do not include high frequency words contained in a stoplist and emphasize capitalized words. This results in the segmentation query scoring function

$$Q_{\text{segment}} = \frac{\sum_{i=1}^{n_y} [f_{\hat{y}_c(i)}(\mathbf{x}(i)) - f_{\tilde{y}_c(i)}(\mathbf{x}(i))]}{n_y}. \quad (12)$$

For entity classification, we begin with segmentation from the previous stage and classify these segments into $\mathcal{Y} \in \{\text{PEOPLE}, \text{LOCATION}, \text{ORGANIZATION}\}$. In this case, there are a small number of entities per sentence and we empirically determined that the least certain entity (i.e. $\mathcal{Q}_{\text{local}(\mathcal{C})}$ from Sect. 3) best captures the uncertainty of the entire sentence. The resulting query scoring function is stated by

$$Q_{\text{NER}} = \min_{i=1, \dots, n_y} [f_{\hat{y}(i)}(\mathbf{x}(i)) - f_{\tilde{y}(i)}(\mathbf{x}(i))]. \quad (13)$$

Finally, relation classification begins with named entity classifications and label each entity pair with $\mathcal{Y} \in \{\text{LOCATEDIN}, \text{WORKFOR}, \text{ORGBASEDIN}, \text{LIVEIN}, \text{KILL}\} \times \{\text{left}, \text{right}\} + \text{NoRELATION}$. Once again, we find that the least certain single local instance works best for determining which sentence to annotate, but exploit the knowledge that the NoRELATION classification is by far the dominant class and will receive adequate annotation regardless of the querying function. Therefore, we define $\mathcal{Y}_+ = \mathcal{Y} \setminus \text{NoRELATION}$ and do not consider this label when calculating the query scoring function,

$$Q_{\text{relation}} = \min_{i=1, \dots, n_y} [f_{\hat{y}_+(i)}(\mathbf{x}(i)) - f_{\tilde{y}_+(i)}(\mathbf{x}(i))]. \quad (14)$$

The data for our experiments was derived from [59], which is an annotation of a set of sentences from TREC documents. In our data, there are 1987 sentences which contain 4645 entities, and 6909 intrasentence pairs of entities (including NoRELATION). The entity labels include 1648 PEOPLE entities, 1872 LOCATION entities, and 858 ORGANIZATION entities. The relation labels include 420 LOCATEDIN, 394 WORKFOR, 451 ORGBASEDIN, 529 LIVEIN, and 270 KILL. These data properties are summarized in Table 5.

For active learning experiments, we first selected 287 of the 1436 sentences (20%) with at least one active relation for testing. From the training data, we constructed 10

Table 5 Data properties for entity and relation extraction task

Sentences	1987
Total entities	4645
PEOPLE	1648
LOCATION	1872
ORGANIZATION	858
MISCELLANEOUS	267
Total (binary) relations	2064
LOCATEDIN	420
WORKFOR	394
ORGANIZATIONBASEDIN	451
LIVEIN	529
KILL	270

different seed sets of labeled data such that each set contains four instances of each type of relation in $\mathcal{Y}^+ = \mathcal{Y} \setminus \text{NoRELATION}$, ignoring direction. Each data point is an average of the ten different \mathcal{S}_l as the initial seed. For each querying phase, $|\mathcal{S}_{\text{select}}| = 1$, and labeled instances are added to \mathcal{S}_l until we meet the stopping criteria, \mathcal{K} , of the performance level of training on all sentences. We present results in terms of the commonly used F1 measure,

$$F1 = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (15)$$

and plot every fifteenth point to improve clarity.

In addition to previously defined querying functions, we also compare the results to a non-adaptive pipeline querying function, $\mathcal{Q}_{\text{uniform}}$, which sets $\beta = [\frac{1}{J}, \dots, \frac{1}{J}]$ for all iterations. This querying function can be viewed as a structured output active learning querying function from Sect. 3 that is not aware of the pipeline assumptions and treats all stages equally. Finally, we also compare the querying functions to $\mathcal{Q}_{\text{random}}$ which selects a random instance for each round of active learning in Algorithm 2.

6.2 Segmentation

The first experiment conducted is active learning for complete queries for the segmentation task, with results shown in Fig. 16. Note that despite good results for active learning on segmentation, this is not the task that we are interested in directly, but only for its utility to downstream processes. However, this specific stage can be viewed as a further affirmation of the framework introduced in for locally learnable instances. The first important observation is that both $\mathcal{Q}_{\text{uniform}}$ and $\mathcal{Q}_{\text{pipeline}}$ perform better than $\mathcal{Q}_{\text{random}}$, although $\mathcal{Q}_{\text{uniform}}$ begins by performing worse earlier in the process. The more important observation is that $\mathcal{Q}_{\text{pipeline}}$ significantly outperforms $\mathcal{Q}_{\text{uniform}}$ and $\mathcal{Q}_{\text{random}}$ throughout all phases of the protocol. The explanation for

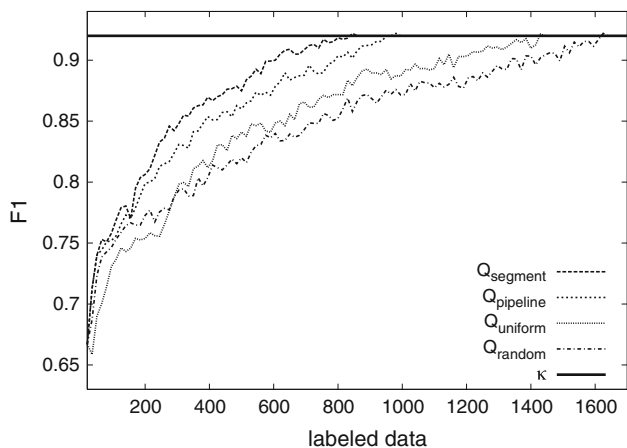


Fig. 16 Experimental results for the segmentation stage of the pipeline. The proposed querying function Q_{pipeline} outperforms Q_{uniform} and Q_{random} , reducing the annotation effort by 45%

this phenomena seems straightforward as Q_{pipeline} emphasizes Q_{segment} early in the procedure, to the point that they are virtually identical for early rounds of active learning. Another interesting point is that Q_{segment} performs better than Q_{pipeline} . Given that this is the first pipeline stage, this result is not particularly surprising as Q_{segment} selects sentences as if this was a single stage task, which we will see hurts performance when later stages should receive greater consideration. However, the final result for segmentation annotation with Q_{pipeline} is that the effort is reduced by 45% as Q_{pipeline} reaches the \mathcal{K} performance level when the quantity of labeled data, $|S_l|$ is approximately 950 instances.

6.3 Entity classification

Figure 17 presents results for active learning on the entity classification stage. For entity classification, once again both Q_{pipeline} and Q_{uniform} perform better than Q_{random} with Q_{pipeline} significantly outperforming Q_{uniform} . A second observation is that we also included Q_{segment} to show that there is significant value in dynamically changing the query scoring function weighting, as even though Q_{segment} does well initially, eventually it reaches a point of diminishing returns and is discounted in favor of later stages. However, it is also interesting to note that Q_{segment} still outperforms Q_{uniform} , demonstrating the value of having earlier stages performing well before emphasizing later stages to reduce error propagation. This experiment is the first to demonstrate that querying instances based on earlier stages ameliorates issues associated with error propagation by ensuring early stages are performing sufficiently well before emphasizing later stages, while also demonstrating that later stages should be emphasized once earlier stages

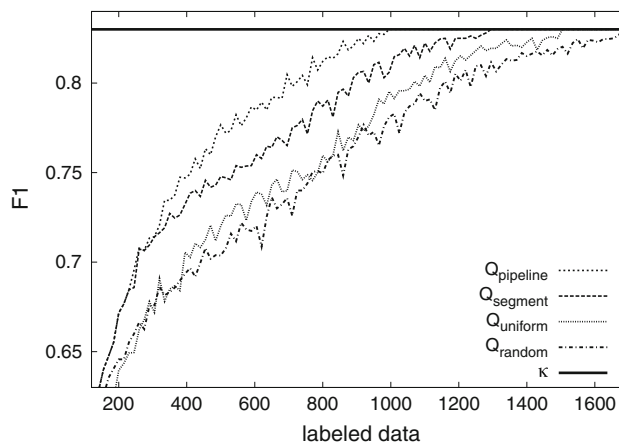


Fig. 17 Experimental results for the entity classification pipeline stage. The proposed querying function Q_{pipeline} outperforms all other querying functions, including Q_{segment} and reduces the annotation effort by 42%

are performing at a level where error propagation is manageable. The final result for entity classification with the proposed methods is that by using Q_{pipeline} , the annotation effort is reduced by 42%.

6.4 Relation classification

Figure 18 presents active learning results for the relation classification stage of the pipeline, also measured by $F1$ as given in Eq. 15. As we see, both Q_{pipeline} and Q_{uniform} once again perform better than Q_{random} with Q_{pipeline} significantly outperforming Q_{uniform} . Secondly, both Q_{uniform} and Q_{pipeline} require more queries early in the process than in other stages before they demonstrate significantly accelerated learning over Q_{random} . This should likely be attributed to the examples that are selected early in the process are being used to learn previous stages and improvements for relation classification is largely a byproduct of reduction in error propagation. This delay is reflected in the overall annotation effort, where we require more examples relative to the segmentation or entity classifications tasks to achieve the same performance as learning with all of the data. However, we still achieve an overall savings of 35%. At the point where Q_{pipeline} returns a hypothesis such that $\mathcal{P}(\hat{h}_t) \geq \mathcal{K}$, Q_{pipeline} outperforms Q_{random} by 5.8 $F1$ score (a relative improvement of 11.4%) at this stage of learning. Note that as we move down the pipeline, we tend to require a greater annotation effort as Q_{pipeline} has to ensure that previous stages are learned adequately before continuing to the present stage as each successive stage builds upon the results of previous stages. A final observation is a comparison of these results to [59], where our final $F1$ score of 0.57 for the relation extraction task and 0.83 for the entity extraction task are competitive with previously reported

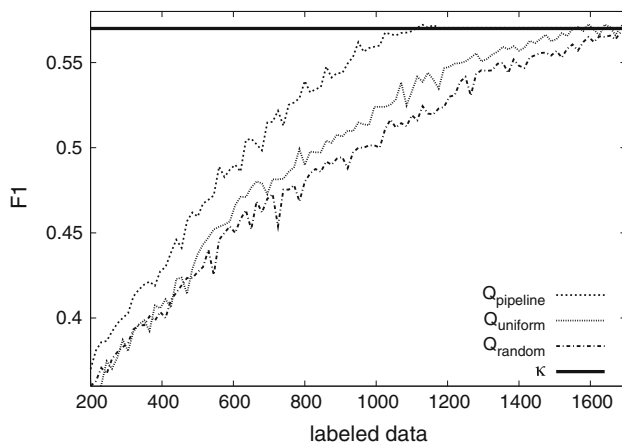


Fig. 18 Experimental results for the relation classification pipeline stage. The proposed querying function Q_{pipeline} reduces the overall annotation effort by 35%

results. However, our system is capable of using raw text as input.

7 Related work

Active learning has gained significant research attention recently, making it nearly impossible to succinctly summarize all of the various strategies; however, several recent theses and surveys do this admirably [48, 63, 66]. One of the earliest active learning works for complex prediction tasks is [69] which studies active learning for both natural language parsing and information extraction from the perspective of *unreliability sampling* [10]. Since then, there has been additional research on active learning in the context of syntactic parsing [41, 67], named entity recognition [65], and other information extraction subtasks [23, 64].

From the perspective of active learning frameworks for structured predictions, the earliest work was for sequence labeling in the context of HMMs [3, 24, 60] although there have been extensions for CRFs [23, 64]. Recently, some of these principles for active learning in structured spaces have been extended to settings where both semi-supervised and active learning are combined by automatically labeling certain local predictions and querying uncertain substructures [70]. More application targeted active learning strategy which emit general principles capable of being extended to other settings include active learning for probabilistic context free grammars (PCFGs) [8, 41]. Pre-dating many of these works in structured output spaces, although relying on similar margin-based principles, include studies regarding active learning for multiclass classification [12, 36, 45, 78] and active learning for ranked data [12, 30].

Our pipeline model framework builds on work regarding methods for autonomously determining the progress of active learning. Works exist which attempt to determine operating regions of the active learning cycle to switch between more exploration driven cycles to more exploitation driven cycles [9, 31]. A closely related concept is attempting to indicate when active learning has achieved its maximum performance without cross-validation data, referred to as deriving a *stopping criteria* [14, 44, 61, 71, 76, 79, 80].

While margin-based active learning remains by far the most popular formalism despite a lack of strong performance guarantees, there have been several recent works examining active learning based upon the PAC learning model [74] for realizable concept classes [7, 16, 25, 27, 37] and the agnostic learning model [42] for broader concept classes [5, 6, 26, 38]. However, while these results are important, it should also be noted that they make assumptions which render them generally less applicable to the complex applications where active learning in most useful, although this is certainly a direction for future study.

8 Conclusions and future directions

This work first describes a margin-based active learning approach for structured output spaces building upon the constrained conditional model (CCM) framework. This facilitates applying active learning to a very general family of structured output scenarios. We first examine the setting of querying complete labels, defining Q_{global} , which is most appropriate when the scoring function $f(\mathbf{x}, \mathbf{y})$ is not decomposable or the data is expected to be exclusively globally learnable. Furthermore, a querying function based on the average margin over component predictions, $Q_{\text{local}(C)}$, is shown to perform better when the scoring function is decomposable and the data is expected to be locally learnable. We further demonstrate that in cases where the local classifications can be queried independently, the labeling effort is most drastically reduced using partial label queries with the querying function $Q_{\text{local}(C)}$. These findings are supported empirically on both synthetic data and the semantic role labeling (SRL) task.

We extend this work to develop a framework for active learning with pipeline models, a widely used paradigm for complex applications where the global task is decomposed into a sequence of predictions where each pipeline stage uses the output of earlier stages as input. This work specifically prescribes a general method for combing the querying functions associated with each pipeline stage into a joint active learning strategy which explicitly exploits properties of a pipeline model. We demonstrate the

effectiveness of this approach on a three-stage named entity and relation extraction system, where we observe a significant reduction in labeled data requirements.

There appears to be many dimensions for future work within both of these frameworks. From the perspective of structured output spaces, an interesting direction would be to examine scenarios where subsets of the output variables can be queried, providing a continuum between single and complete labels. Furthermore, developing a more realistic model of annotation cost along this continuum would likely facilitate the application of this work to a wider range of structured output applications. Finally, there has been recent work regarding combining active and semi-supervised learning for sequence labeling [70] that seems directly applicable to this work, particularly if substructures can be queried. From the perspective of pipelines, the most immediate direction for future work would be to examine both deeper pipelines and more general feedforward network structures (e.g. POS and WSD being parallel input to segmentation). Finally, combining active learning with domain adaptation [17, 53] is particularly appropriate for cases where structural information may inform the querying function by allowing structural similarities between domains to be specified and possibly lead to further reductions in annotation requirements.

Acknowledgments The authors would like to thank Ming-Wei Chang, Alex Klementiev, Vasin Punyakanok, Nick Rizzolo, and the reviewers for their helpful comments regarding this work. This work has been partially funded by NSF grant ITR IIS-0428472, a research grant from Motorola Labs, DARPA funding under the Bootstrap Learning Program, and by MIAS, a DHS funded Center for Multimodal Information Access and Synthesis at UIUC.

References

- Abney S (2002) Bootstrapping. In: Proceedings of the annual meeting of the association for computational linguistics (ACL), pp 360–367
- Allwein EL, Schapire RE, Singer Y (2000) Reducing multiclass to binary: a unifying approach for margin classifiers. *J Mach Learn Res* 1:113–141
- Anderson B, Moore A (2005) Active learning for hidden Markov models: objective functions and algorithms. In: Proceedings of the international conference on machine learning (ICML), pp 9–16
- Angluin D (1988) Queries and concept learning. *Mach Learn* 2(4):319–342
- Balcan M-F, Beygelzimer A, Langford J (2006) Agnostic active learning. In: Proceedings of the international conference on machine learning (ICML), pp 65–72
- Balcan M-F, Broder A, Zhang T (2007) Margin-based active learning. In: Proceedings of the annual ACM workshop on computational learning theory (COLT), pp 35–50
- Balcan MF, Hanneke S, Wortman J (2008) The true sample complexity of active learning. In: Proceedings of the annual ACM workshop on computational learning theory (COLT), pp 45–56
- Baldrige J, Osborne M (2004) Active learning and the total cost of annotation. In: Proceedings of the conference on empirical methods for natural language processing (EMNLP), pp 9–16
- Baram Y, El-Yaniv R, Luz K (2004) Online choice of active learning algorithms. *J Mach Learn Res* 5:255–291
- Becker M (2008) Active learning: an explicit treatment of unreliable parameters. PhD thesis, University of Edinburgh
- Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the annual ACM workshop on computational learning theory (COLT), pp 92–100
- Brinker K (2004) Active learning of label ranking functions. In: Proceedings of the international conference on machine learning (ICML), pp 129–136
- Bunescu RC (2008) Learning with probabilistic features for improved pipeline models. In: Proceedings of the conference on empirical methods for natural language processing (EMNLP), pp 670–679
- Campbell C, Cristianini N, Smola A (2000) Query learning with large margin classifiers. In: Proceedings of the international conference on machine learning (ICML), pp 111–118
- Carreras X, Marquez L (2004) Introduction to the conll-2004 shared tasks: semantic role labeling. In: Proceedings of the annual conference on computational natural language learning (CoNLL)
- Castro RM, Nowak RD (2007) Minimax bounds for active learning. In: Proceedings of the Annual ACM workshop on computational learning theory (COLT), pp 5–19
- Chan YS, Ng HT (2007) Domain adaptation with active learning for word sense disambiguation. In: Proceedings of the annual meeting of the association for computational linguistics (ACL), pp 49–56
- Chang M-W, Do Q, Roth D (2006) Multilingual dependency parsing: a pipeline approach. In: Recent advances in natural language processing. Springer, Berlin, pp 195–204
- Chang M-W, Ratnikov L, Rizzolo N, Roth D (2008) Learning and inference with constraints. In: Proceedings of the national conference on artificial intelligence (AAAI), pp 1513–1518
- Cohn D, Atlas L, Ladner R (1994) Improving generalization with active learning. *Mach Learn* 15(2):201–222
- Cohn DA, Ghahramani Z, Jordan MI (1996) Active learning with statistical models. *J Artif Intell Res* 4:129–145
- Collins M (2002) Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In: Proceedings of the conference on empirical methods for natural language processing (EMNLP), pp 1–8
- Culotta A, McCallum A (2005) Reducing labeling effort for structured prediction tasks. In: Proceedings of the national conference on artificial intelligence (AAAI), pp 746–751
- Dagan I, Engelson SP (1995) Committee-based sampling for training probabilistic classifiers. In: Proceedings of the international conference on machine learning (ICML), pp 150–157
- Dasgupta S (2004) Analysis of a greedy active learning strategy. In: The conference on advances in neural information processing systems (NIPS), pp 337–344
- Dasgupta S, Hsu D, Monteleoni C (2007) A general agnostic active learning algorithm. In: The conference on advances in neural information processing systems (NIPS), vol 20, pp 353–360
- Dasgupta S, Kalai AT, Monteleoni C (2005) Analysis of perceptron-based active learning. In: Proceedings of the annual ACM workshop on computational learning theory (COLT), pp 249–263
- Daumé III H, Langford J, Marcu D (2009) Search-based structured prediction. *Mach Learn* 75(3):297–325
- Davis PC (2002) Stone soup translation: the linked automata model. PhD thesis, Ohio State University
- Donmez P, Carbonell J (2008) Optimizing estimated loss reduction for active sampling in rank learning. In: Proceedings of

- the international conference on machine learning (ICML), pp 248–255
31. Donmez P, Carbonell JG, Bennett PN (2007) Dual strategy active learning. In: Proceedings of the European conference on machine learning (ECML), pp 116–127
 32. Duda RO, Hart PE, Stork DG (2001) Pattern classification, 2nd edn. Wiley-Interscience, New York
 33. Finkel JR, Manning CD, Ng AY (2006) Solving the problem of cascading errors: approximate bayesian inference for linguistic annotation pipelines. In: Proceedings of the conference on empirical methods for natural language processing (EMNLP), pp 618–626
 34. Freund Y, Schapire RE (1997) An decision-theoretic generalization of on-line learning and an application to boosting. *J Comput Syst Sci* 55(1):119–139
 35. Freund Y, Schapire RE (1999) Large margin classification using the perceptron algorithm. *Mach Learn* 37(3):277–296
 36. Godbole S, Harpale A, Sarawagi S, Chakrabarti S (2004) Document classification through interactive supervision of document and term labels. In: Proceedings of the European conference on principles and practice of knowledge discovery in databases (PKDD), pp 185–196
 37. Hanneke S (2007) A bound on the label complexity of agnostic active learning. In: Proceedings of the international conference on machine learning (ICML), pp 353–360
 38. Hanneke S (2007) Teaching dimension and the complexity of active learning. In: Proceedings of the annual ACM workshop on computational learning theory (COLT), pp 66–81
 39. Har-Peled S, Roth D, Zimak D (2002) Constraint classification for multiclass classification and ranking. In: The conference on advances in neural information processing systems (NIPS), pp 785–792
 40. Hinton G, Sejnowski TJ (1999) Unsupervised learning: foundations of neural computation. MIT Press, Cambridge
 41. Hwa R (2004) Sample selection for statistical parsing. *Comput Linguist* 30(3):253–276
 42. Kearns MJ, Schapire RE, Sellie LM (1994) Toward efficient agnostic learning. *Mach Learn* 17(2–3):115–141
 43. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the international conference on machine learning (ICML), pp 282–289
 44. Laws F, Schütze H (2008) Stopping criteria for active learning of named entity recognition. In: Proceedings of the international conference on computational linguistics (COLING), pp 465–472
 45. Luo T, Kramer K, Goldfog DB, Hall LO, Samson S, Remsen A, Hopkins T (2005) Active learning to recognize multiple types of plankton. *J Mach Learn Res* 6:589–613
 46. Nguyen HT, Smeulders A (2004) Active learning using pre-clustering. In: Proceedings of the international conference on machine learning (ICML), pp 623–630
 47. Och FJ, Tillmann C, Ney H (1999) Improved alignment models for statistical machine translation. In: Proceedings of the conference on empirical methods for natural language processing (EMNLP), pp 20–28
 48. Olsson F (2009) A literature survey of active machine learning in the context of natural language processing. Technical report, Swedish Institute of Computer Science
 49. Punyakanok V, Roth D, Yih W, Zimak D (2005) Learning and inference over constrained output. In: Proceedings of the international joint conference on artificial intelligence (IJCAI), pp 1124–1129
 50. Punyakanok V, Roth D, Yih W, Zimak D (2004) Semantic role labeling via integer linear programming inference. In: Proceedings of the international conference on computational linguistics (COLING)
 51. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Francisco
 52. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *IEEE* 77(2):257–286
 53. Rai P, Saha A, Hal Daume III HD, Venkatasubramanian S (2010) Domain adaptation meets active learning. In: NAACL workshop on active learning for NLP (ALNLP)
 54. Roth D, Small K (2006) Margin-based active learning for structured output spaces. In: Proceedings of the European conference on machine learning (ECML), pp 413–424
 55. Roth D, Small K (2008) Active learning for pipeline models. In: Proceedings of the national conference on artificial intelligence (AAAI), pp 683–688
 56. Roth D, Small K, Titov I (2009) Sequential learning of classifiers for structured prediction problems. In: Proceedings of the international conference on artificial intelligence and statistics (AI-STATS), pp 440–447
 57. Roth D, Yih W-T (2004) A linear programming formulation for global inference in natural language tasks. In: Proceedings of the annual conference on computational natural language learning (CoNLL), pp 1–8
 58. Roth D, Yih W-T (2005) Integer linear programming inference for conditional random fields. In: Proceedings of the international conference on machine learning (ICML), pp 737–744
 59. Roth D, Yih W-T (2007) Global inference for entity and relation identification via a linear programming formulation. In: Introduction to statistical relational learning
 60. Scheffer T, Wrobel S (2001) Active learning of partially hidden Markov models. In: Proceedings of the ECML/PKDD workshop on instance selection
 61. Schohn G, Cohn D (2000) Less is more: active learning with support vector machines. In: Proceedings of the international conference on machine learning (ICML), pp 839–846
 62. Sekine S, Sudo K, Nobata C (2002) Extended named entity hierarchy. In: Proceedings of the international conference on language resources and evaluation (LREC), pp 1818–1824
 63. Settles B (2009) Active learning literature survey. Technical Report 1648, University of Wisconsin-Madison
 64. Settles B, Craven M (2008) An analysis of active learning strategies for sequence labeling tasks. In: Proceedings of the conference on empirical methods for natural language processing (EMNLP), pp 1069–1078
 65. Shen D, Zhang J, Su J, Zhou G, Tan C-L (2004) Multi-criteria-based active learning for named entity recognition. In: Proceedings of the annual meeting of the association for computational linguistics (ACL), pp 589–596
 66. Small K (2005) Interactive learning protocols for natural language applications. PhD thesis, University of Illinois at Urbana-Champaign
 67. Tang M, Luo X, Roukos S (2002) Active learning for statistical natural language parsing. In: Proceedings of the annual meeting of the association for computational linguistics (ACL), pp 120–127
 68. Taskar B, Guestrin C, Koller D (2003) Max-margin Markov networks. In: The conference on advances in neural information processing systems (NIPS)
 69. Thompson CA, Califf ME, Mooney RJ (1999) Active learning for natural language parsing and information extraction. In: Proceedings of the international conference on machine learning (ICML), pp 406–414
 70. Tomanek K, Hahn U (2009) Semi-supervised active learning for sequence labeling. In: Proceedings of the annual meeting of the association for computational linguistics (ACL), pp 1039–1047
 71. Tomanek K, Wermter J, Hahn U (2007) An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In: Proceedings of the conference

- on empirical methods for natural language processing (EMNLP), pp 486–495
72. Tong S, Koller D (2001) Support vector machine active learning with applications to text classification. *J Mach Learn Res* 2:45–66
 73. Tschantz I, Hofmann T, Joachims T, Altun Y (2004) Support vector machine learning for interdependent and structured output spaces. In: Proceedings of the international conference on machine learning (ICML), pp 823–830
 74. Valiant LG (1984) A theory of the learnable. *Commun ACM*, pp 1134–1142
 75. Vapnik VN (1999) The nature of statistical learning theory, 2nd edn. Springer, Berlin
 76. Vlachos A (2008) A stopping criterion for active learning. *Comput Speech Lang* 22(3):295–312
 77. Waterman DA (1986) A guide to expert systems. Addison-Wesley, Reading
 78. Yan R, Yang J, Hauptmann A (2003) Automatically labeling video data using multiclass active learning. In: Proceedings of the international conference on computer vision (ICCV), pp 516–523
 79. Zhu J, Wang H, Hovy EH (2008) Learning a stopping criterion for active learning for word sense disambiguation and text classification. In: Proceedings of the international joint conference on natural language processing (IJCNLP), pp 366–372
 80. Zhu J, Wang H, Hovy EH (2008) Multi-criteria-based strategy to stop active learning for data annotation. In: Proceedings of the international conference on computational linguistics (COLING), pp 1129–1136
 81. Zhu X (2005) Semi-supervised learning literature survey. Computer Sciences 1530, University of Wisconsin-Madison