Active Learning for Pipeline Models

Dan Roth and **Kevin Small**
Department of Computer Science
University of Illinois at Urbana-Champaign
{danr, ksmall}@uiuc.edu

Abstract

For many machine learning solutions to complex applications, there are significant performance advantages to decomposing the overall task into several simpler sequential stages, commonly referred to as a pipeline model. Typically, such scenarios are also characterized by high sample complexity, motivating the study of active learning for these situations. While most active learning research examines single predictions, we extend such work to applications which utilize pipelined predictions. Specifically, we present an adaptive strategy for combining local active learning strategies into one that minimizes the annotation requirements for the overall task. Empirical results for a three-stage entity and relation extraction system demonstrate a significant reduction in supervised data requirements when using the proposed method.

Introduction

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 (Chang, Do, & Roth 2006; Finkel, Manning, & Ng 2006). For example, consider the relation extraction subtask of information extraction where the goal is to extract named relationships between entities in a given text. In this situation, relation classification is often the final stage of a pipeline consisting of previous stages such as phrase segmentation and named entity classification as seen in Figure 1. Furthermore, these stages may be preceded by other simpler related natural language processing tasks such as part of speech tagging.

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 input text will result in a complex function that may be impossible to learn. A second aspect of such domains is the corresponding high cost associated with obtaining sufficient labeled data for good learning performance. The active learning protocol offers one promising solution to 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 (Cohn, Ghahramani, & Jordan 1996). While receiving significant recent attention, most active learning research focuses on new algorithms as they relate to a single classification task. This work instead assumes that an active learning algorithm exists for each stage of a pipelined learning model and develops a strategy that jointly minimizes the annotation requirements for the pipelined process.

This paper 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 reasonably error free input. Furthermore, we apply this method to a three stage named entity and relation extraction system, demonstrating significant reductions in annotation requirements.

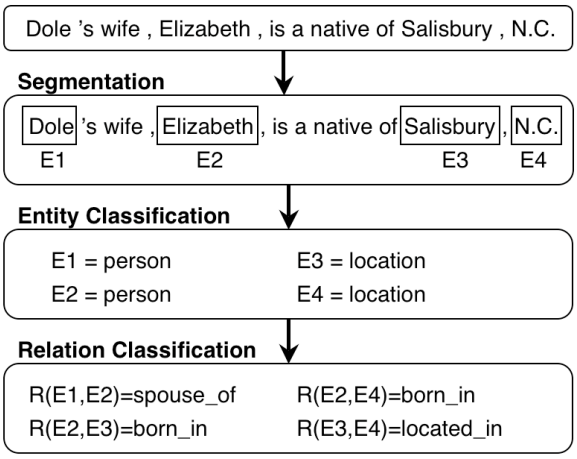

Figure 1: A three-stage pipeline model for named entity and relation extraction

Preliminaries

Learning Pipeline Models

Following the standard classification task, let $x \in \mathcal{X}$ represent members in an input domain and $y \in \mathcal{Y}$ represent members of an output domain where we require a prediction function $h : \mathcal{X} \rightarrow \mathcal{Y}$. This work 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 $\hat{y} = h(x) = \text{argmax}_{y' \in \mathcal{Y}} f(\mathbf{x}, y')$. In a pipeline model, each stage $d = 1, \ldots, D$ has access to the input instance in addition to the classifications from all previous stages, $\Phi^{(d)}(x, \hat{y}^{(0)}, \ldots, \hat{y}^{(d-1)}) \rightarrow \mathbf{x}^{(d)}$. Each stage of a pipelined learning process takes m training instances $\mathcal{S}^{(d)} = \left\{(\mathbf{x}_1^{(d)}, y_1^{(d)}), \ldots, (\mathbf{x}_m^{(d)}, y_m^{(d)})\right\}$ as input to a learning algorithm $\mathcal{A}^{(d)}$ and returns a classifier, $h^{(d)}$, which minimizes the respective loss function of the d^{th} stage. Note that each stage may vary in complexity from a single binary prediction, $y^{(d)} \in \{-1, 1\}$, to a multiclass prediction, $y^{(d)} \in \{\omega_1, \ldots, \omega_k\}$, to a structured output prediction, $y^{(d)} \in \mathcal{Y}_1^{(d)} \times \cdots \times \mathcal{Y}_{n_y}^{(d)}$. 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,

$$\hat{\mathbf{y}} = h(x) = \left\{ \underset{y' \in \mathcal{Y}^{(d)}}{\text{argmax}} f^{(d)}\left(\mathbf{x}^{(d)}, y'\right) \right\}_{d=1}^{D}. \quad (1)$$

Active Learning

Active learning describes the protocol where the learner maintains the ability to select examples from an unlabeled data source $\mathcal{S}_u$ with the goal of selecting instances which will most rapidly improve its hypothesis. The key difference between active learning and standard supervised learning is a querying function, $\mathcal{Q}$, which when provided with the data $\mathcal{S}$ and the learned classifier h returns a set of unlabeled instances $\mathcal{S}_{select} \subseteq \mathcal{S}_u$. These selected instances are labeled and added to the supervised training set $\mathcal{S}_l$ used to update the learned hypothesis. Example selection criteria used for single predictions with a single classifier include minimizing uncertainty (Cohn, Ghahramani, & Jordan 1996; Tong & Koller 2001) and maximizing expected error reduction (Roy & McCallum 2001).

The work in this paper 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,

$$\mathcal{Q} : x_\star = \underset{x \in \mathcal{S}_u}{\text{argmin}}\, q(x). \quad (2)$$

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 require that each $q^{(d)}$ be of similar range and shape such that the values may be effectively compared and combined.

Active Learning for Pipeline Models

Given a pipeline model and a query scoring function for each stage of the pipeline, $q^{(d)}$, 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

$$\mathcal{Q}_{pipeline} : x_\star = \underset{x \in \mathcal{S}_u}{\text{argmin}} \sum_{d=1}^{D} \beta^{(d)} \cdot q^{(d)}(x). \quad (3)$$

Based upon this formulation, the goal is to set the values of $\boldsymbol{\beta}_t$ for each querying phase of the active learning protocol by exploiting properties of pipeline models. Some observed properties of a well designed pipeline which most strongly affect selecting values for $\boldsymbol{\beta}_t$ include:

1. The results of earlier stages are useful, and often necessary, for later stages.
2. Earlier stages are easier to learn than later stages.
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 is stated by

$$\hat{\mathbf{y}} = \underset{\mathbf{y}' \in \mathcal{Y}^{(1)} \times \cdots \times \mathcal{Y}^{(D)}}{\text{argmax}} \sum_{d=1}^{D} \pi^{(d)} \cdot f^{(d)}\left(\mathbf{x}^{(d)}, \mathbf{y}'\right) \quad (4)$$

where $\boldsymbol{\pi}$ is used to determine the relative importance associated with correct predictions for each stage of the pipeline, noting that in most cases $\boldsymbol{\pi} = [0, \ldots, 0, 1]$. Comparing equation 4 to the pipelined prediction function of equation 1, 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 treated independently and only the predictions are required to propagate information between stages. 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 d possesses a certain degree of robustness to noise from the input $\Phi^{(d)}(x, \hat{y}^{(0)}, \ldots, \hat{y}^{(d-1)})$. If this tolerance is exceeded, stage d will no longer make reliable predictions and will lead to errors cascading to later stages. This notion results in the prime 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 d implies that stages $1, \ldots, (d-1)$ are performing sufficiently well and stage d should be emphasized.
3. Conversely, lack of performance improvement at stage d implies that stages $1, \ldots, (d-1)$ are not performing well and should be emphasized by the querying function.

The first criteria is trivial to satisfy by setting $\boldsymbol{\beta}_0 = [1, 0, \ldots, 0]$. The remaining criteria are more difficult as an estimate of querying function performance at each stage is required to update $\boldsymbol{\beta}$ without labeled data for cross-validation. (Donmez, Carbonell, & Bennett 2007) prescribe such a procedure in the setting 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, $\hat{\epsilon} = (\sum_{x \in \mathcal{S}_u} E[(\hat{y} - y)^2 | x])/|\mathcal{S}_u|$ where $E[(\hat{y} - y)^2 | x] = \sum_{y \in \mathcal{Y}} \mathcal{L}_{0/1}(\hat{y}, y) P(y|x)$ and $\mathcal{L}_{0/1}$ is the 0/1 loss function. Once the change in expected error is small, $\frac{\Delta \hat{\epsilon}}{\Delta t} < \delta$, the current configuration is deemed to be achieving diminishing returns and the second querying function should be used.

Our work derives an analogous method in the context of pipeline models, where operating regions correspond to the segment of the pipeline being emphasized. The first observation is that we cannot directly extend the aforementioned procedure 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 complex classifiers comprising each stage. Furthermore, intricate knowledge of these parameters is required to reasonably specify $\delta^{(d)}$. However, a second observation is that $\hat{\epsilon}$ is their query scoring function which we generalize to basing our method on the average of the query scoring function over the unlabeled data, $U_t^{(d)} = (\sum_{x \in \mathcal{S}_u} q^{(d)}(x))/|\mathcal{S}_u|$. The intuition is that $U_t^{(d)}$ represents the certainty of $f^{(d)}$ for each iteration of active learning and once this value stops increasing between iterations, $\mathcal{Q}^{(d)}$ is likely entering an operating region of diminishing returns and should be discounted. Since δ would be difficult to calibrate for multiple stages and irrevocable crossover points would be undesirable in the pipelined case, we opt for an algorithm where each stage competes with other stages based on the relative value changes in $U^{(d)}$, resulting in Algorithm 1.

Algorithm 1 begins by taking as input the seed labeled data $\mathcal{S}_l$, unlabeled data $\mathcal{S}_u$, the learning algorithm for each stage $\mathcal{A}^{(d)}$, a query scoring function for each stage $q^{(d)}$, an update rate parameter λ, and an active learning stopping criteria κ. Lines 2-7 initialize the algorithm by learning an initial hypothesis $h_0^{(d)}$ for each stage, calculating the initial average query scoring function value $U_0^{(d)}$ for each stage, and setting $\boldsymbol{\beta}_0 = [1, 0, \ldots, 0]$. Line 8 checks if active learning stopping criteria has been met. If not, lines 9-11 select instances $\mathcal{S}_{select}$ according to the current $\boldsymbol{\beta}$ which are removed from $\mathcal{S}_u$, labeled, and added to $\mathcal{S}_l$. Lines 12-16 update the hypothesis for each stage and calculate the new values of $U_t^{(d)}$ for each stage. After $\boldsymbol{\Delta}_t$ is normalized (line 16), we update the value of $\beta_t^{(d)}$ for each stage based on the relative improvements of $U_t^{(d)}$. Finally, $\boldsymbol{\beta}$ is normalized (line 19) and the process is repeated. Fundamentally, based upon earlier stated principles, Algorithm 1 assumes that $\boldsymbol{\beta} = [1, 0, \ldots, 0]$ is the optimal mixing parameter at $t = 0$ and tracks this non-stationary parameter over t based on the feedback provided by $(U_t^{(d)} - U_{t-1}^{(d)})$ at line 15.

Algorithm 1 Active Learning for Pipeline Models

1: **Input:** $\mathcal{S}_l, \mathcal{S}_u, \{\mathcal{A}^{(d)}\}_{d=1}^{D}, \{q^{(d)}\}_{d=1}^{D}, \lambda, \kappa$

2: **for** $d \leftarrow 1, \ldots, D$ **do** {initialize algorithm}
3: $\quad h_0^{(d)} \leftarrow \mathcal{A}^{(d)}(\mathcal{S}_l)$
4: $\quad U_0^{(d)} \leftarrow \frac{\sum_{x \in \mathcal{S}_u} q^{(d)}(x)}{|\mathcal{S}_u|}$
5: $\quad \beta_0^{(d)} \leftarrow 0$
6: $\beta_0^{(1)} \leftarrow 1$
7: $t \leftarrow 1$

8: **while** !κ **do** {query new examples}
9: $\quad \mathcal{S}_{select} \leftarrow \text{argmin}_{x \in \mathcal{S}_u} \sum_{d=1}^{D} \beta_{t-1}^{(d)} \cdot q^{(d)}(x)$
10: $\quad \mathcal{S}_u \leftarrow \mathcal{S}_u \backslash \mathcal{S}_{select}$
11: $\quad \mathcal{S}_l \leftarrow \mathcal{S}_l \cup \mathcal{S}_{select}$ {expert labels $\mathcal{S}_{select}$}

12: $\quad$ **for** $d \leftarrow 1, \ldots, D$ **do** {update hypothesis}
13: $\quad\quad h_t^{(d)} \leftarrow \mathcal{A}^{(d)}(\mathcal{S}_l)$
14: $\quad\quad U_t^{(d)} \leftarrow \frac{\sum_{x \in \mathcal{S}_u} q^{(d)}(x)}{|\mathcal{S}_u|}$
15: $\quad\quad \Delta_t^{(d)} \leftarrow U_t^{(d)} - U_{t-1}^{(d)}$
16: $\quad \boldsymbol{\Delta}_t \leftarrow \frac{\boldsymbol{\Delta}_t}{\|\boldsymbol{\Delta}_t\|}$

17: $\quad$ **for** $d \leftarrow 1, \ldots, D$ **do** {update $\boldsymbol{\beta}$}
18: $\quad\quad \beta_t^{(d)} \leftarrow \beta_{t-1}^{(d)} + \lambda \cdot \Delta_t^{(d)}$
19: $\quad \boldsymbol{\beta}_t \leftarrow \frac{\boldsymbol{\beta}_t}{\|\boldsymbol{\beta}_t\|}$
20: $\quad t \leftarrow t + 1$

21: **Output:** $\{h^{(d)}\}_{d=1}^{D}$

A Three-stage Discriminative Entity and Relation Extraction System

The experimental setting we explore with this protocol is the three-stage entity and relation extraction system shown in Figure 1. 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 of (Roth & Small 2006). Let $x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_{n_x}$ represent an input instance and $\mathbf{y} \in \mathcal{C}(\mathcal{Y}^{n_y})$ represent a structured assignment in the space of output variables $\mathcal{Y}_1 \times \cdots \times \mathcal{Y}_{n_y}$. $\mathcal{C} : 2^{\mathcal{Y}^*} \rightarrow 2^{\mathcal{Y}^*}$ represents a set of constraints that enforces structural consistency on $\mathbf{y}$, making the prediction function $\hat{\mathbf{y}}_{\mathcal{C}} = h(x) = \text{argmax}_{y' \in \mathcal{C}(\mathcal{Y}^{n_y})} f(\mathbf{x}, \mathbf{y}')$.

While active learning often relies upon the use of support vector machines (Tong & Koller 2001), good results have also been shown with a regularized version of the structured Perceptron algorithm (Collins 2002). The regularized structured Perceptron adds a large margin component heuristically, requiring thick multiclass separation between the class activations to determine hypothesis updates. In this case, $f(\mathbf{x}, \mathbf{y}) = \boldsymbol{\alpha} \cdot \Phi(\mathbf{x}, \mathbf{y})$ represents the global scoring function such that $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^{|\mathcal{Y}|})$ is a concatenation of the

Algorithm 2 Regularized Inference Based Training

Input: $\mathcal{S} \in \{\mathcal{X}^* \times \mathcal{Y}^*\}^m, \gamma, T$
$\boldsymbol{\alpha} \leftarrow 0$
for T iterations **do**
  for all $(\mathbf{x}, \mathbf{y}) \in \mathcal{S}$ **do**
    $\hat{\mathbf{y}}_{\mathcal{C}} \leftarrow \text{argmax}_{\mathbf{y} \in \mathcal{C}(\mathcal{Y}^{n_y})} \boldsymbol{\alpha} \cdot \Phi(\mathbf{x}, \mathbf{y})$
    for all $i = 1, \ldots, n_y$ **do**
      if $f_{y_i}(\mathbf{x}, i) - \gamma < f_{\hat{y}_{\mathcal{C},i}}(\mathbf{x}, i)$ **then**
        $\boldsymbol{\alpha}^{y_i} \leftarrow \boldsymbol{\alpha}^{y_i} + \Phi^{y_i}(\mathbf{x}, i)$
        $\boldsymbol{\alpha}^{\hat{y}_i} \leftarrow \boldsymbol{\alpha}^{\hat{y}_i} - \Phi^{\hat{y}_i}(\mathbf{x}, i)$
Output: $\{f_y\}_{y \in \mathcal{Y}} \in \mathcal{H}$

local $\boldsymbol{\alpha}^y$ vectors and $\Phi(\mathbf{x}, \mathbf{y}) = (\Phi^1(\mathbf{x}, \mathbf{y}), \ldots, \Phi^{|\mathcal{Y}|}(\mathbf{x}, \mathbf{y}))$ is a concatenation of the local feature vectors, $\Phi^y(\mathbf{x}, \mathbf{y})$. $f_y(\mathbf{x}, i) = \boldsymbol{\alpha}^y \cdot \Phi^y(\mathbf{x}, i)$ where $\boldsymbol{\alpha}^y \in \mathbb{R}^{d_y}$ is the learned weight vector and $\Phi^y(\mathbf{x}, i)$ is the local feature vector. Given that $\dot{y} = \text{argmax}_{y' \in \mathcal{Y} \setminus y} f_{y'}(\mathbf{x})$ corresponds to the highest activation value such that $\dot{y} \neq y$, the learning algorithm for each stage, $\mathcal{A}^{(d)}$, is described by Algorithm 2.

As a discriminative framework, performance is strongly correlated to the quality of $\Phi^{(d)}$. We extract features in a method similar to (Roth & Yih 2004) except segmentation is not assumed, but is 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 directly adjacent words. 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. 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, and professional titles compiled in advance. 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 (Roth & Yih 2004) (e.g. $\Phi_{arg1,prof,arg2}$(CNN reporter David McKinley) $= 1$).

Active Entity and Relation Extraction

As stated, this formulation for active learning with pipeline models requires that each stage of the pipeline has a predefined query scoring function $q^{(d)}$. To design $q^{(d)}$ for each stage of our system, we build upon previous work for active learning in structured output spaces (Roth & Small 2006). This previous work relies upon the decomposition of structured predictions into a vector of multiclass predictions and derive active learning querying functions based upon the expected multiclass margin. Defining $\tilde{y} = \text{argmax}_{y' \in \mathcal{Y} \setminus \hat{y}} f_{y'}(\mathbf{x})$ as the label corresponding to the second highest activation value, the multiclass classification querying function is $\mathcal{Q}_{multiclass} : x_\star = \text{argmin}_{x \in \mathcal{S}_u} [f_{\hat{y}}(\mathbf{x}) - f_{\tilde{y}}(\mathbf{x})]$.

To extend $\mathcal{Q}_{multiclass}$ 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 $\mathcal{Y} \in \{begin, inside, outside\}$. The constraints $\mathcal{C}$ enforce a valid structure by ensuring that *inside* only follows a *begin* label for BIO segmentation. We follow (Roth & Small 2006) 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

$$q_{segment} = \frac{\sum_{i=1}^{n_y} [f_{\hat{y}_{\mathcal{C}}}(\mathbf{x}, i) - f_{\tilde{y}_{\mathcal{C}}}(\mathbf{x}, i)]}{n_y}. \quad (5)$$

For entity classification, we begin with segmentation from the previous stage and classify these segments into $\mathcal{Y} \in \{person, location, organization\}$. In this case, there are a small number of entities per sentence and we empirically determined that the least certain entity best captures the uncertainty of the entire sentence. The resulting query scoring function is stated by

$$q_{NER} = \min_{i=1,\ldots,n_y} [f_{\hat{y}}(\mathbf{x}, i) - f_{\tilde{y}}(\mathbf{x}, i)]. \quad (6)$$

Finally, relation classification begins with named entity classifications and label each entity pair with $\mathcal{Y} \in \{located\_in, work\_for, org\_based\_in, live\_in, kill\} \times \{left, right\} + no\_relation$. Once again, we find that the least certain single instance works best for determining which sentence to annotate, but exploit the knowledge that the *no_relation* classification is by far the dominant class and will receive adequate annotation regardless of the querying function. Therefore, we define $\mathcal{Y}_+ = \mathcal{Y} \setminus no\_relation$ and do not consider this label when calculating the query scoring function,

$$q_{relation} = \min_{i=1,\ldots,n_y} \left[f_{\hat{y}_+}(\mathbf{x}, i) - f_{\tilde{y}_+}(\mathbf{x}, i)\right]. \quad (7)$$

Experimental Results

The data for our experiments was derived from (Roth & Yih 2007), which is an annotation of a set of sentences from TREC documents. In our data, there are 1,987 sentences which contain 4,645 entities, and 6,909 intrasentence pairs of entities. The entity labels include 1,648 *person* entities, 1,872 *location* entities, and 858 *organization* entities. The relation labels include 420 *located_in*, 394 *work_for*, 451 *org_based_in*, 529 *live_in*, and 270 *kill*.

For active learning experiments, we first selected 287 of the 1,436 sentences (20%) with at least one active relation for testing. From the training data, we constructed 10 different seed sets of labeled data such that each set contains four instances of each type of relation in $\mathcal{Y}^+ = \mathcal{Y} \setminus no\_relation$, ignoring direction. Each data point is an average of the ten different $\mathcal{S}_l$ as the initial seed. For each querying phase, $|\mathcal{S}_{select}| = 1$, and labeled instances are added to $\mathcal{S}_l$ until we meet the stopping criteria, κ, of the performance level of training on all sentences. We present results in terms on $F1 = (2 * precision * recall)/(precision + recall)$ 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, $\mathcal{Q}_{uniform}$, which sets $\boldsymbol{\beta} = \left[\frac{1}{D}, \ldots, \frac{1}{D}\right]$ for all iterations. This querying function can be viewed as a structured output active learning function that is not aware of the pipeline assumptions and treats all stages equally. Finally, we also compare the querying functions to $\mathcal{Q}_{random}$ which selects a random instance for each pass of Algorithm 1.

Segmentation

Figure 2 shows the segmentation task results when selecting sentences for complete annotation. Note that despite good results for segmentation, this is not the task that we are interested in directly, but only for its utility to downstream processes. The first important observation is that both $\mathcal{Q}_{uniform}$ and $\mathcal{Q}_{pipeline}$ perform better than $\mathcal{Q}_{random}$, although $\mathcal{Q}_{uniform}$ starts by performing worse. The more important observation is that $\mathcal{Q}_{pipeline}$ significantly outperforms $\mathcal{Q}_{uniform}$ and $\mathcal{Q}_{random}$ throughout all phases of the protocol. The explanation for this phenomena seems straightforward as $\mathcal{Q}_{pipeline}$ emphasizes $\mathcal{Q}_{segment}$ early in the procedure, to the point that they are virtually identical early in the process. Another interesting point is that $\mathcal{Q}_{segment}$ performs better than $\mathcal{Q}_{pipeline}$. Given that this is the first pipeline stage, this result is not particularly surprising as $\mathcal{Q}_{segment}$ selects sentences as if this was a single stage task, which we will see hurts performance of later stages. However, the final result for segmentation annotation with $\mathcal{Q}_{pipeline}$ is that the effort is reduced by 45%.

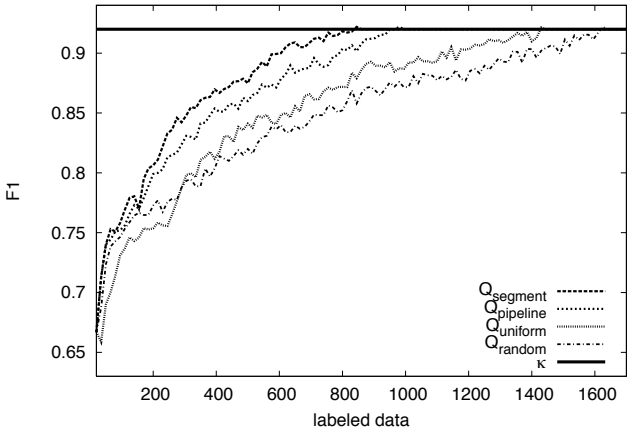

Figure 2: Experimental results for the segmentation stage of the pipeline. The proposed querying function $\mathcal{Q}_{pipeline}$ outperforms $\mathcal{Q}_{uniform}$ and $\mathcal{Q}_{random}$, reducing the annotation effort by 45%. While $\mathcal{Q}_{segment}$ performs better at this stage, it will be shown detrimental to downstream processes.

Entity Classification

Figure 3 presents results for the entity classification stage. For entity classification, once again both $\mathcal{Q}_{pipeline}$ and $\mathcal{Q}_{uniform}$ perform better than $\mathcal{Q}_{random}$ with $\mathcal{Q}_{pipeline}$ significantly outperforming $\mathcal{Q}_{uniform}$. A second observation is that we also included $\mathcal{Q}_{segment}$ to show that there is significant value in dynamically changing the query scoring function weighting as even though $\mathcal{Q}_{segment}$ 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 $\mathcal{Q}_{segment}$ still outperforms $\mathcal{Q}_{uniform}$, demonstrating the value of having earlier stages performing well before emphasizing later stages to reduce error propagation. The final result for entity classification is that by using $\mathcal{Q}_{pipeline}$, the annotation effort is reduced by 42%.

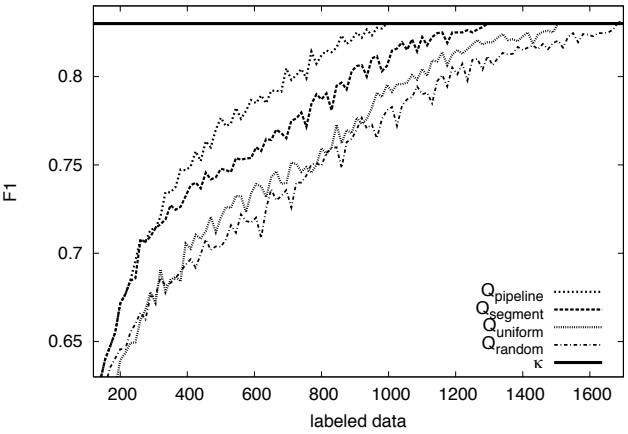

Figure 3: Experimental results for the entity classification pipeline stage. The proposed querying function $\mathcal{Q}_{pipeline}$ outperforms all other querying functions, including $\mathcal{Q}_{segment}$ and reduces the annotation effort by 42%.

Relation Classification

Figure 4 presents results for the relation classification stage of the pipeline, also measured by $F1$. As we see, both $\mathcal{Q}_{pipeline}$ and $\mathcal{Q}_{uniform}$ once again perform better than $\mathcal{Q}_{random}$ with $\mathcal{Q}_{pipeline}$ significantly outperforming $\mathcal{Q}_{uniform}$. Secondly, both $\mathcal{Q}_{uniform}$ and $\mathcal{Q}_{pipeline}$ require more queries early in the process than in other stages before they demonstrate significantly accelerated learning over $\mathcal{Q}_{random}$. 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 incidental. 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%. Note that as we move down the pipeline, we tend to require a greater annotation effort as $\mathcal{Q}_{pipeline}$ 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 note is a comparison of these results to (Roth & Yih 2007), 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 results. However, our system does not assume that segmentation is provided and thereby can be used directly with textual input.

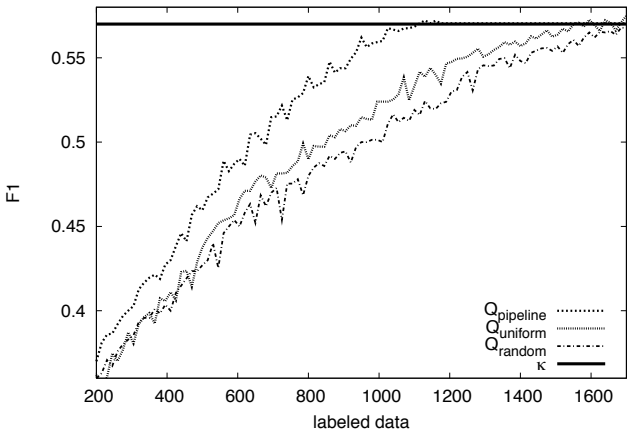

Figure 4: Experimental results for the relation classification pipeline stage. The proposed querying function $\mathcal{Q}_{pipeline}$ reduces the overall annotation effort by 35%.

Related Work

One of the earliest active learning works for more complex tasks is (Thompson, Califf, & Mooney 1999) which studies active learning for both natural language parsing and information extraction from the perspective of assigning uncertainty based on the reliability of application specific algorithms and is not a general classification framework. In the context of active learning for named entity classification, some recent works include (Becker *et al.* 2005; Jones 2005; Shen *et al.* 2004). However, in these works the task is primarily entity classification and the problem is not cast as a pipeline model. (Culotta & McCallum 2005) looks at information extraction framed as a structured output prediction and performs active learning based on conditional random fields. While not a pipeline model and not performing relation extraction, this work is interesting as it examines the benefits of a more sophisticated interaction with the user.

Conclusion

The pipeline model is a widely used paradigm for machine learning solutions to complex applications, where the overall task is decomposed into a sequence of predictions for which each pipeline stage uses previous predictions as input. For these applications, a second issue is often lack of sufficient annotated data. This paper prescribes a general method for combining active learning approaches for each separate pipeline stage into a joint active learning strategy that explicitly exploits properties of a pipeline. We demonstrate the effectiveness of the stated methods on a three stage named entity and relation extraction system, where we see a significant reduction in the need for annotated data.

Acknowledgments

The authors would like to thank Ming-Wei Chang, Alex Klementiev, Nick Rizzolo, and the reviewers for helpful comments regarding this work. This work was supported by NSF grant ITR IIS-0428472, DARPA funding under the Bootstrap Learning Program and by MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

References

Becker, M.; Hachey, B.; Alex, B.; and Grover, C. 2005. Optimising selective sampling for bootstrapping named entity recognition. In *ICML Workshop on Multiple Views*.

Chang, M.-W.; Do, Q.; and Roth, D. 2006. Multilingual dependency parsing: A pipeline approach. In *Recent Advances in Natural Language Processing*, 195–204.

Cohn, D. A.; Ghahramani, Z.; and Jordan, M. I. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research* 4:129–145.

Collins, M. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Culotta, A., and McCallum, A. 2005. Reducing labeling effort for structured prediction tasks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Donmez, P.; Carbonell, J. G.; and Bennett, P. N. 2007. Dual strategy active learning. In *Proc. of the European Conference on Machine Learning (ECML)*.

Finkel, J. R.; Manning, C. D.; and Ng, A. Y. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jones, R. 2005. *Learning to Extract Entities from Labeled and Unlabeled Text*. Ph.D. Dissertation, Carnegie Mellon.

Roth, D., and Small, K. 2006. Margin-based active learning for structured output spaces. In *Proc. of the European Conference on Machine Learning (ECML)*.

Roth, D., and Yih, W.-T. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*.

Roth, D., and Yih, W.-T. 2007. Global inference for entity and relation identification via a linear programming formulation. In *Introduction to Statistical Relational Learning*.

Roy, N., and McCallum, A. 2001. Toward optimal active learning through sampling estimation of error reduction. In *Proc. of the International Conference on Machine Learning (ICML)*, 441–448.

Shen, D.; Zhang, J.; Su, J.; Zhou, G.; and Tan, C.-L. 2004. Multi-criteria-based active learning for named entity recognition. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 589–596.

Thompson, C. A.; Califf, M. E.; and Mooney, R. J. 1999. Active learning for natural language parsing and information extraction. In *Proc. of the International Conference on Machine Learning (ICML)*, 406–414.

Tong, S., and Koller, D. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* 2:45–66.