Online Stacked Graphical Learning

Zhenzhen Kou
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
zkou@andrew.cmu.edu

Vitor R. Carvalho
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
vitor@cs.cmu.edu

William W. Cohen
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

ABSTRACT

Collective classification predicts class labels simultaneously for a group of related instances, rather than predicting a class for each instance separately. The existing collective classification methods are usually expensive due to the iterative inference in graphical models and their learning procedures based on iterative optimization. When the dataset is large, the cost of maintaining large graphs or related instances in memory becomes a problem as well. *Stacked graphical learning* has been proposed for collective classification with efficient inference. However, the memory and time cost of standard stacked graphical learning is still expensive since it requires cross-validation-like predictions to be constructed during training. In this paper, we integrate recently-developed single-pass online learning with stacked learning, to save training time and to handle large streaming datasets with minimal memory overhead. Experimentally we will show that online stacked graphical learning gives accurate results on eleven sample problems from three domains, with less time and memory cost.

Keywords

Collective Classification, Stacked Graphical Learning, Relational Graphical Models, Online Learning Algorithms

1. INTRODUCTION

There are many relational datasets, such as hyperlinked webpages, scientific literature with dependencies among citations, and social networks. *Collective classification* has been widely used for classification on relational datasets. Collective classification predicts class labels simultaneously for a group of related instances, rather than predicting a class for each instance separately. Recently there have been studies on relational models for collective inference, such as relational dependency networks[1], relational Markov networks[2], Markov logic networks[3], and stacked graphical learning[4]. The existing collective classification methods are usually expensive due to the iterative inference in graphical models and their learning procedures based on iterative optimization. Also for large dataset, the cost of maintaining large graphs or related instances in memory becomes a problem.

Stacked graphical learning was proposed in the previous work[4]. Stacked graphical learning is a meta-learning method, which augments a base learner by providing the predicted labels of related instances. One advantage of stacked graphical learning is that the inference is very efficient. The previous work[4] shows that stacked graphical learning is 40 to 80 times faster than Gibbs sampling during inference. However, the time and memory cost during training for standard stacked graphical learning can be expensive since it applies a base learner to the training data in a cross-validation-like way to make predictions.

In this paper, we proposed a scheme to integrate recently-developed single-pass online learning with stacked learning, to save training time and to handle large streaming datasets with minimal memory overhead. During the learning procedure of an online learner, the intermediate predictions for training data are generated to learn the online model. Thus the predictions for training data can be obtained naturally and there is no need to apply the base learner several times to the training data to obtain the predictions. Therefore online stacked graphical models will save training time. Also the single-pass online learning provides reliable predictions and the learner needs to maintain only the classifiers and does not need to store all the examples in memory. Thus online stacked graphical learning will save training time and memory.

2. ONLINE STACKED GRAPHICAL LEARNING

2.1 Single-Pass Online Learning

Compared to batch methods, online learning methods are often simpler to implement, faster, and require considerably less memory. For such reasons, these techniques are natural ones to consider for large-scale learning problems. Online learning methods, such as Perceptron or Winnow, are also naturally suited to stream processing; however, in practice multiple passes over the same training data are required to achieve accuracy comparable to state-of-the-art batch learners.

In order to address this problem, Carvalho & Cohen [9] investigated the performance of different algorithms in the *single-pass online learning* setting, i.e., online learning algo-

KDD '07 San Jose, CA USA

rithms restricted to a single training pass over the available data. This setting is particularly relevant when the system cannot afford several passes throughout the training set: for instance, when dealing with massive amounts of data, or when memory or processing resources are restricted, or when data is not stored but presented in a stream.

Their work revealed that some single-pass online learning algorithms can provide batch-level performance on a variety of tasks. More specifically, it was observed that in classification tasks for datasets with sparse features (very common in Natural Language Processing tasks), a modification of the Balanced Winnow algorithm (MBW or Modified Balanced Winnow) [9] presented excellent performance - even comparable to batch learners. They also observed that a variation on the Perceptron algorithm called *Voted Perceptron* [23] presented fairly good results on classification tasks when the feature is not sparse.

Voting (a.k.a. averaging) an online classifier is a technique that, instead of using the best hypothesis learned so far, uses a weighted average of all hypotheses learned during a training procedure. The averaging procedure is expected to produce more stable models, which leads to less overfitting [24]. Averaging techniques have been successfully used with the Perceptron algorithm [23] as well as with several other online learning algorithms, including MBW [9].

2.2 MBW

MBW is a modification of the Balanced Winnow algorithm, which in turn is an extension of the Winnow algorithm [25, 26]. It is based on multiplicative updates and it assumes the incoming example x_t is a vector of positive weights, i.e., $x_{t,j} \geq 0$, $\forall t$ and $\forall j$, where $x_{t,j}$ denotes the j^{th} feature of x_t. This assumption is usually satisfied in NLP tasks, where the $x_{t,j}$ values are typically the frequency of a term, presence of a feature, TFIDF value of a term, etc. The learning algorithm is detailed in Table 1.

In general terms, for each new example x_t presented, the current model will make a prediction $\widehat{y}_t \in \{-1, 1\}$ and compare it to the true class $y_t \in \{-1, 1\}$. The prediction will be based on the score function f, on the example x_t and on the current hypothesis. MBW is mistake-driven, i.e., only in the case of a prediction mistake the hypothesis (or model) will be updated.

Like Balanced Winnow, MBW has a promotion parameter $\alpha > 1$, a demotion parameter β, where $0 < \beta < 1$ and a threshold parameter $\theta_{th} > 0$. It also has a margin parameter M, where $M \geq 0$.

After the algorithm is initialized, an *augmentation* and a *normalization* preprocessing step is applied to each incoming example x_t. When learning, the algorithm receives a new example x_t with m features, and it initially *augments* the example with an additional feature(the $(m+1)^{th}$ feature), whose value is permanently set to 1. This additional feature is typically known as "bias" feature. After *augmentation*, the algorithm then *normalizes* the sum of the weights of the augmented example to 1, therefore restricting all feature weights to $0 \leq x_{t,j} \leq 1$.

In MBW, the hypothesis is a combination of two parts: a positive model u_t and a negative model v_t. After *normalization*, the score function is calculated as $score = \langle x_t, u_i \rangle - \langle x_t, v_i \rangle - \theta_{th}$, where $\langle x_t, w_i \rangle$ denote the intermediate product of vectors x_t and w_i.

If the prediction is mistaken, i.e., $(score \cdot y_t) \leq M$, then the models are updated. The update rule will be based on multiplicative operations on the two models, taking into consideration the promotion and demotion parameters (α and β), as well as the particular feature weight of the incoming example.

Table 1: Modified Balanced Winnow (MBW).

1. Initialize $i = 0$, and models u_0 and v_0.
2. For $t = 1, 2, ..., T$:
   (a) Receive new example x_t.
   (b) Augmentation: add "bias" feature to x_t.
   (c) Normalize x_t to 1.
   (d) Calculate $score = \langle x_t, u_i \rangle - \langle x_t, v_i \rangle - \theta_{th}$.
   (e) Receive true class y_t.
   (f) If prediction was mistaken, i.e., $(score \cdot y_t) \leq M$:
      i. Update models. For all feature j s.t. $x_t > 0$:

$$u_{i+1,j} = \begin{cases} u_{i,j} \cdot \alpha \cdot (1 + x_{t,j}) & \text{, if } y_t > 0 \\ u_{i,j} \cdot \beta \cdot (1 - x_{t,j}) & \text{, if } y_t < 0 \end{cases}$$

$$v_{i+1,j} = \begin{cases} v_{i,j} \cdot \beta \cdot (1 - x_{t,j}) & \text{, if } y_t > 0 \\ v_{i,j} \cdot \alpha \cdot (1 + x_{t,j}) & \text{, if } y_t < 0 \end{cases}$$

Following the parameters suggested by by Carvalho & Cohen [9], our implementation sets the promotion parameter $\alpha = 1.5$, the demotion parameter $\beta = 0.5$, the threshold $\theta_{th} = 1.0$, the "margin" M was set to 1.0, and the initial weights were $\theta_0^+ = 2.0$ and $\theta_0^- = 1.0$.

In testing mode, the *augmentation* step in MBW is the same, but there is a small modification in the *normalization*. Before the normalization of the incoming instance, the algorithm checks each feature in the instance to see if it is already present in the current models (u_i and v_i). The features not present in the current model are then removed from the incoming instance before the normalization takes place.

2.3 Stacked Graphical Learning

We consider here collective classification tasks, in which the goal is to "collectively" classify some set of instances. In our notation, a dataset is $D = \{(x_1, y_1), ..., (x_n, y_n)\}$. An example is a pair of (x_i, y_i). $\mathbf{x}$ denote a set of instances and $\mathbf{y}$ is the corresponding labels for $\mathbf{x}$. For example, in a dataset of linked webpages, x_i can be a bag-of-word representation of a webpage and y_i is the category of x_i.

Stacked graphical learning (SGL) captures the dependency by expanding the feature of an instance x_i with "predicted" labels for the related instances. In SGL, a *relational template* C finds the related instances. A relational template is a procedure that finds all the examples related to a given example and returns their indices[4]. For example, in a collection of linked webpages, given a webpage x_i, that in the dataset webpages $x_{i_1}, ..., x_{i_L}$ are related to x_i (i.e., either link-to or link-from x_i), and predictions $\hat{\mathbf{y}}$ for the set of webpages $\mathbf{x}$, $C(x_i, \hat{\mathbf{y}})$ returns the predictions of the related webpages, i.e., $\hat{y}_{i_1}, ..., \hat{y}_{i_L}$.

Since the relation between x_i and x_j might be one-to-many, for example, webpages link to different numbers of

- Parameters: a relational template C.
- Learning algorithm: Given a training set $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ and a base learner A:
  - Learn the local model, i.e., when $k = 0$:
    Let $f^0 = A(D^0)$. Please note that $D^0 = D$.
  - Learn the stacked models, for $k = 1...K$:
    1. Construct predictions $\hat{y}_i^{k-1}$ for $x_i^{k-1} \in D^{k-1}$ in a cross-validation-like way, as shown in Figure 2.
    2. Construct an extended dataset $D^k = \{(x_1^k, y_1), ..., (x_n^k, y_n)\}$ by converting each instance x_i to x_i^k as follows: $x_i^k = (x_i, C(x_i, \hat{\mathbf{y}}^{k-1}))$, where $C(x_i, \hat{\mathbf{y}}^{k-1})$ will return the predictions for examples related to x_i such that $x_i^k = (x_i, \hat{y}_{i_1}^{k-1}, ..., \hat{y}_{i_L}^{k-1})$.
    3. Let $f^k = A(D^k)$.
- Inference algorithm: given a set of testing instances $\mathbf{x}$:

  1. $\hat{\mathbf{y}}^0 = f^0(\mathbf{x})$.

  For $k = 1...K$,

  2. Carry out Step 2 above to produce $\mathbf{x}^k$.
  3. $\mathbf{y}^k = f^k(\mathbf{x}^k)$.

  Return $\mathbf{y}^K$.

Figure 1: Standard Stacked Graphical Learning and Inference. k: the level of stacking. x_i^k: the instance expanded from x_i, $\hat{y}_i^k$: the prediction of x_i, and f^k: the learned classifier, at level k^{th} stacking.

Given a training set $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ and a base learner A, construct cross-validated predictions as follows:

1. Split D into J equal-sized disjoint subsets $D_1...D_J$.
2. For $j = 1...J$, let $f_j = A(D - D_j)$. That is, train a classifier f_j, based all the data from D except the subset D_j.
3. For $x \in D_j, \hat{y} = f_j(x)$. That is, for data in D_j, apply the classifier f_j to obtain its prediction.

Figure 2: A cross-validation-like technique to obtain predictions for training examples

webpages, we allow aggregation functions to combine predictions on a set of related instances into a single feature[4].

In standard SGL, we apply a cross-validation-like technique suggested by a meta-learning scheme, *stacking* [6], to obtain the predictions for training data. The procedure to obtain the predictions for training examples is shown in Figure 2. Figure 1 shows the learning and inference procedure in standard stacked graphical learning. In our notation, k denotes the level of stacking, x_i^k denotes the instance expanded from x_i at stacking of level k, $\hat{y}_i^k$ denotes the prediction of x_i at stacking of level k, and f^k denotes the classifier learned at level k^{th} stacking.

The cross-validation parameter J is set to 5 by default. The previous work has shown that SGL converges quickly[4] and usually we choose $K = 1$, i.e., one iteration of stacking.

The previous work[4] has shown that stacked graphical learning is very efficient during inference. However, the standard training scheme can be expensive since it applies the base learner to the training set several times, in a cross-validation-like way. In this paper, we proposed to integrate single-pass online learning with stacked graphical learning to construct online stacked graphical models to save training time and memory.

2.4 Online Stacked Graphical Learning

2.4.1 The algorithm

During the learning procedure of an online learner, the intermediate predictions for training data are generated to learn the online model. Thus the predictions for training data can be obtained naturally and there is no need to apply the base learner many times to the training data in a cross-validation-like procedure to obtain the predictions. Therefore combining the online learning scheme with stacked graphical models can save training time.

One practical difficulty is that, while online learning methods produce satisfactory predictions after learning on the whole training set, the intermediate predictions for the training data in the starting stage can be quite inaccurate. Thus, to obtain fair "predictions" for training examples, we define a burn-in data size b. That is, after training on b examples, we start recording intermediate predictions from the online learner and expanding features with the predictions. The learning procedure of online stacked learning is shown in Figure 3. Figure 3 shows that in the learning procedure of online stacked graphical models, f^0 is trained on the whole training dataset. After training on b examples, we start recording the intermediate predictions $\hat{y}_b^0, ..., \hat{y}_n^0$, which are generated naturally during the learning of f^0. For the first level of stacking, i.e., $k = 1$, we apply the relational template to expand features (i.e., $x_i^1 = (x_i, \hat{y}_{i_1}^0, ..., \hat{y}_{i_L}^0)$), and train f^1 with expanded examples $(x_b^1, y_b)..., (x_n^1, y_n)$. Similarly, for the k^{th} level of stacking, intermediate predictions $\hat{y}_{kb}^{k-1}, ..., \hat{y}_n^{k-1}$ (which are generated naturally during the learning of f^{k-1}) are recorded to expand features and the k^{th} stacked model is trained with expanded instances $x_{kb}^k, ..., x_n^k$.

One thing we would like to point out is that, in stacked

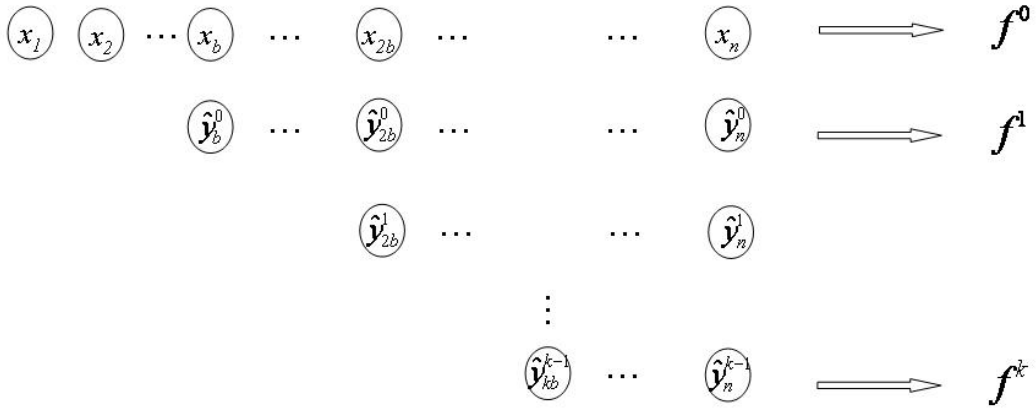

Figure 3: Online Stacked Graphical Learning

graphical learning for collective classification, given an instance x_i, we need to apply the relational template to retrieve the predicted labels for the related instances to extend features. Assume x_i and its neighbors are contained in a subset, we provide the instances in a subset to the online learner as a group and extend the features after the predictions for instances in the whole subset are made. Therefore in general, we provide the instances in groups to the base learner and the burn-in data size b will be chosen to include a few subsets of instances. In practise, the dataset might not be able to be split into disjoint subsets. In Section 3 we will demonstrate how to split the dataset into subsets empirically.

2.4.2 *Efficiency Analysis*

Theoretically, when there are infinitely many training examples, i.e., $kb << n$, applying the online stacked graphical learning shown in Figure 3 only requires single-pass training over the training set. We do not need to apply the cross-validation-like trick shown in Figure 2 to get the predictions for training examples. Therefore, online stacked graphical learning can save training time. In Section 3 we will show the speed-up experimentally as well.

In online stacked graphical learning, there are reliable predictions at level k after $(k+1)b$ examples have streamed by, and the learner needs to maintain only k classifiers and does not need to store examples. Therefore, the algorithm can save memory. This becomes extremely important when the size of training data is huge. Also this feature allows online stacked graphical learning to be applied to streaming data.

2.4.3 *An Implementation With Limited Data*

Theoretically, we assume $kb << n$ and online stacked graphical learning only requires single-pass training over the training set. In practice, the assumption $kb << n$ may not hold. An implementation with limited training data is to let $b = n/2$ and apply a one-and-half-pass procedure shown in Figure 4, to obtain the predictions for training examples. Using the procedure shown in Figure 4 to obtain the predictions, we end up with a learning and inference method similar to the procedure shown in Figure 1, except that the predictions are no longer obtained in a cross-validation-like way.

Given a training set $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ and an online learner A, construct predictions as follows:

1. Give the training data,$(x_1, y_1), ..., (x_n, y_n)$, to the online learner, train a classifier f_1, and record the intermediate predictions from online learning on x_j for $j = n/2, ..., n$.
2. While $x_j, j = n/2, ..., n$, streaming by, train another online learner f_2 with $(x_{n/2}, y_{n/2}), ..., (x_n, y_n)$, go back to $x_1, ..., x_{n/2}$, keep learning f_2 and record the intermediate predictions for $x_j, j = 1, ..., n/2 - 1$.

Figure 4: The procedure to obtain predictions for training examples via an online base learner, with limited data

3. EXPERIMENTAL RESULTS

We evaluated stacked graphical learning on tasks from three domains - collective classification over relational datasets, sequential partitioning[20], and named entity extraction.

3.1 Relational Datasets

The relational datasets we consider here include text region detection in Subcellular Location Image Finder (SLIF) [7, 8] and document classification.

SLIF is a system which extracts information from both figures and the associated captions in biological journal articles. The text region detection dataset contains candidate regions found in 1396 panels from 207 figures. The dataset contains 4129 connections among the examples. The problem studied in this paper is to classify if the candidate regions are text regions or not. More details about the dataset and the dependencies defined in the data can be found in the previous paper[4].

We use MBW as the base online learner for SLIF. The features are the same as in the previous work[4]. In the SLIF text region detection task, the candidate regions can be naturally grouped into disjoint subgraphs, i.e., candidate regions from the same figure construct a subgraph. There-

fore as long as the prediction for candidate regions from the same figure is obtained, we can apply the relational template to expand features. The relational template is the same as the previous paper[4].

The document classification includes the webpage classification on the WebKB dataset[10] and paper classification on the Cora dataset and the CiteSeer dataset[11]. The WebKB data contains aproximately 3800 webpages labelled from 6 categories and 8000 hyperlinks. The Cora data[12] contains 2708 papers labelled from seven categories and 5429 citations. The Citeseer data[13] contains 3312 papers labelled from six categories and 4732 citations. Our current implementation of MBW only supports binary labels, so we considered the task corresponding to the most common label. The relational template for document classification is the same as the previous paper[4].

We use MBW as the base online learner for document classification. The feature sets and relational templates are the same as the previous paper[4]. The WebKB dataset contains webpages from four computer science departments. Thus we split them into groups according to departments. We group the papers in Cora dataset by the year of publishing. There is no such year-of-publishing information available for the Citeseer dataset, thus we only applied the implementation shown in Figure 4 to Citeseer data.

3.2 Sequential Partitioning Datasets

Sequential partitioning tasks are sequential classification tasks characterized by long runs of identical labels: examples of these tasks include document analysis, video segmentation, and gene finding[20]. In this paper we consider three datasets.

The *signature* dataset is originated from the problem of recognizing the "signature" section of an email message. Each line of an email message [19] is labels as either *positive* or *negative*. A *positive* label indicates that a particular line in the message was part of a signature section, and *negative* otherwise. This dataset contains 33,013 labeled lines from 617 email messages. About 10% of the lines are labeled "positive". We used the "basic" feature set from Carvalho & Cohen [19].

One set of tasks involved classifying lines from FAQ documents with labels like "header", "question", "answer", and "trailer". We used the features adopted by McCallum *et al* [21] and the ai-general task adopted by Dietterich *et al* [22]. The data consists of 7 long sequences, each sequence corresponding to a single FAQ document; the task contains 10909 labeled lines. Our current implementation only supports binary labels, so we considered the label "answer" (A) for the FAQ dataset.

Another task was video segmentation task, in which the goal is to take a sequence of video "shots" (a sequence of adjacent frames taken from one camera) and classify them into categories such as "anchor", "news" and "weather". This dataset contains 12 sequences, each corresponding to a single video clip. There are a total of 406 shots, and about 700 features, which are produced by applying LDA to a 5x5, 125-bin RGB color histogram of the central frame of the shot. We constructed a video partitioning task, corresponding to the most common label[20].

We use a Modified Balance Winnow learner[9] as the base online learner in stacked graphical learning for sequential partitioning. In the sequential partitioning task, the instance is naturally grouped into sequences. Therefore as long as the prediction for a sequence is obtained, we can apply the relational template to expand features. The relational templates returns the predictions of ten adjacent examples (five preceding examples and five following examples).

3.3 Named Entity Extraction Datasets

We applied stacked graphical learning to named entity extraction from Medline abstracts and emails. We used three datasets to evaluate our method for protein name extractions. The University of Texas, Austin dataset contains 748 labeled abstracts[14]; the GENIA dataset contains 2000 labeled abstracts[15]; and the YAPEX dataset contains 200 labeled abstracts[16]. We also study person name extraction from the email message corpus. The CSpace corpus we used in this paper contains 216 email messages collected from a management course at Carnegie Mellon University[17].

The feature sets and relational templates for named entity extraction are the same as the previous work [4]. The relational template will retrieve the predictions for the adjacent words (with window size 5) and for the same word appearing in one abstract, apply the COUNT aggregator, and return the number of words in each category, given one word. That is, let w_i be the word in a document. For words $w_j = w_i$ in the same document, we count the number of times w_j appearing with label y and use it as one of the stacked features for w_i.

In addition to this relational template, we applied another relational template which just retrieves the predictions for the adjacent words (with window size 5).

3.4 Accuracy of Stacked Graphical Learning with efficient training

To evaluate the effectiveness of online stacked graphical learning on the collective classification task, in Table 2 we compare local models, stacked models, and a state-of-art competitive model. We evaluated two local models, MaxEnt and MBW. We considered a standard stacked model based on MaxEnt (with two-fold-cross-validation predictions), a standard stacked model based on MBW (with two-fold-cross-validation predictions), and an online stacked graphical model based on MBW. We also compared our stacked graphical model to a state-of-art relational graphical model, *relational dependency networks*[1].

Relational dependency network (RDN) uses the same features as the stacked model, but learns via a pseudo-likelihood method, and does inference with Gibbs sampling. Jensen's package, *PROXIMITY*[1], provides an implementation of RDNs, which takes Relational Probability Trees (RPT) for the conditional probability distribution (CPD) component [1]. We implemented RDN with MaxEnt for the CPD component, using the same model graph as SGL and the same aggregations as the relational template. With our implementation, it is easier to compare the running time.

Table 2 shows that on all of the four relational datasets, stacked graphical learning improves the performance of the base learner significantly. The two local models achieved performance of the same level, so did the stacked graphical models based on them. Our comparison to relational dependency networks shows that stacked models can achieve

[1]http://kdl.cs.umass.edu/software/

Table 2: Performance of online stacked graphical learning for relational datasets: accuracy for "Document classification" and F1-accuracy for "SLIF" are reported. We evaluated two local models: MaxEnt and MBW. We also compared to a competitive relational model - relational dependency networks. The standard stacked model used two-fold-cross-validation predictions. The online stacked graphical model is based on MBW. We used 1 level of stacking, i.e., K=1.

	SLIF	Document classification		
		WebKB	Cora	CiteSeer
Local model				
MaxEnt	77.2	58.3	63.9	55.3
MBW	79.5	58.6	63.7	56.1
Competitive relational model				
Relational Dependency Networks	86.7	**74.2**	72.9	58.7
Stacked model				
Standard Stacked model (with MaxEnt, k=1)	90.1	73.2	73.8	59.8
Standard Stacked model (with MBW, k=1)	92.1	**74.2**	**73.5**	**60.3**
Online Stacked model (k=1)	**92.3**	74.1	71.3	-

Table 3: Accuracy comparison of online stacked graphical learning for sequential partitioning. We evaluated two local models: MaxEnt and MBW. We compared to a competitive graphical model - conditional random fields. The standard stacked model used two-fold-cross-validation predictions. The online stacked graphical model is based on MBW. We used 1 level of stacking.

	Sequential Partitioning		
	FAQ	signature	video
Local model			
MaxEnt	67.3	96.3	80.9
MBW	64.9	96.5	78.4
Competitive relational model			
CRFs	85.6	98.1	83.0
Stacked model			
Standard Stacked model (with MaxEnt, k=1)	**87.1**	98.1	**85.8**
Standard Stacked model (with MBW, k=1)	84.1	**98.3**	85.5
Online Stacked model (k=1)	86.3	**98.3**	85.7

Table 4: Performance of online stacked graphical learning for Named Entity Extraction, F1 accuracy is reported. "Relational template 1" returns predictions of adjacent tokens only, "relational template 2" returns predictions of adjacent and repeated tokens.

	Named Entity Extraction			
	UT	Yapex	Genia	CSpace
Local model				
MaxEnt	69.1	62.1	66.5	74.2
MBW	67.9	62.3	66.9	75.1
Competitive relational model				
CRFs	73.1	65.7	72.0	80.3
Stacked model				
With relational template 1				
Standard Stacked model (with MaxEnt, k=1)	70.1	63.7	70.8	77.9
Standard Stacked model (with MBW, k=1)	72.1	63.9	71.3	79.9
Online Stacked model (with MBW, k=1)	72.6	64.6	72.3	80.0
With relational template 2				
Standard Stacked model (with MaxEnt, k=1)	77.3	68.2	78.5	82.1
Standard Stacked model (with MBW, k=1)	**76.6**	68.9	**78.9**	83.3
Online Stacked model (with MBW, k=1)	**76.6**	**69.1**	**78.9**	**83.4**

competitive results to the state-of-art model. However, the online stacked graphical model requires much less training time, which will be discussed later.

One thing we want to point out is that, due to the lack of information on the year of publication, we can not implement online stacked model to Citeseer data. And the performance of online stacked model for Cora data is not as good as the standard stacked graphical models. The reason for the performance drop is that providing papers in the order of years of publication to the online learner can only provide the predictions of papers that were published before the current timestamp and were cited by the current paper, i.e., the predictions available sofar can only provide information on the papers cited by the current paper, while in reality, there is also information contained in the paper that would be published and would cite the current paper.

Table 3 shows the performance of online stacked models on sequence partitioning. The state-of-art models we consider here are conditional random fields (CRFs). CRFs are sequential models that can capture the sequential dependency. On all of the three datasets, stacked graphical learning improves the performance of the base learner significantly. The MaxEnt model did better than MBW on two of three tasks, yet the stacked graphical models based on them achieved performance of the same level.

Table 4 reported the F1-accuracy of online stacked graphical learning for Named Entity Extraction. In Section 3.3 we described two relational templates for named entity extraction. One relational template captures sequential dependency only(denoted as relational template 1 in Table 4), the other one can also capture the dependency among the adjacent and repeated tokens(denoted as relational template 2 in Table 4).

Table 4 shows that on all of the four named entity extraction tasks, stacked graphical learning improves the performance of the base learner. With relational template 1, the stacked graphical models can capture the sequential dependency and achieved comparable results to CRFs. With relational template 2, the stacked graphical models achieved better performance than CRFs. Moreover, the online stacked graphical model requires much less training time.

3.5 Efficiency of the Training for Stacked Graphical Learning

One big success of online stacked graphical learning is that the learning is an online procedure and thus very efficient. We compared the training time of online stacked graphical models (with one iteration) to that of competitive relational models and the baseline standard stacked graphical model. The baseline algorithm we compare to is the best algorithm in previous work[4], the standard stacked graphical model based on MaxEnt, with 5-fold-cross-validation to obtain predictions during training. We compare the baseline algorithm to the online stacked graphical learning with implementation shown in Figure 4.

Table 5 shows the speedup, i.e., in the table "38.1" means the training in standard stacked graphical learning is 38.1 times slower than that of online stacked graphical learning. Table 5 shows that compared to online stacked graphical learning, standard stacked graphical learning based on MaxEnt is approximately 57 times slower in training.

We also compared online stacked graphical learning with the competitive relational models. Table 5 shows that online stacked graphical learning is approximately 14 times faster in training. Moreover, in the previous work[4], it has been shown that during inference stacked graphical learning is 40 to 80 times faster than Gibbs sampling in relational dependency networks[2].

Therefore, online stacked graphical models can achieve high accuracy with efficient training and testing.

Table 5: Comparison on training time.

	Standard SGM vs Online SGM	Competitive relational model vs Online SGM
SLIF	38.1	7.9
WebKB	50.0	10.1
Cora	49.7	9.9
Signature	67.4	13.6
FAQ	69.0	14.0
Video	45.0	9.7
UT	68.7	20.3
Yapex	60.6	17.1
Genia	69.4	22.4
CSpace	52.0	15.3
Average speed-up	57.0	14.0

4. CONCLUSIONS

Collective classification has been widely studied for classification on relational datasets. The existing relational graphical models are usually expensive due to the iterative inference and their learning procedures based on iterative optimization. Also for large datasets, the cost of maintaining large graphs or related instances in memory becomes a problem. Stacked graphical learning was proposed in the previous work and the inference of stacked graphical learning was shown to be very efficient - approximately 40 to 80 times faster than Gibbs sampling.

In this paper we presented an online training scheme for stacked graphical learning. Integrating single-pass online learning algorithm with stacked graphical learning can save the time and memory cost during training. With fast training and inference, stacked graphical learning is very competitive in applications where an efficient algorithm is extremely important. Experimentally we demonstrated that the approach gives accurate results on eleven sample problems from three domains. In addition to the improvement on efficiency, with the online learning scheme, stacked graphical learning is also able to be applied to streaming data.

Future work will compare stacked models to more graphical models such as relational Markov networks, and further explore relational template design and base learner selection. We are also considering more applications of stacked graphical learning, such as the application to streaming data.

5. REFERENCES

[1] D. Jensen and J. Neville , *Dependency Networks for Relational Data*, Proceedings of ICDM-04, Brighton, UK 2004.

[2]We also compared the training time of online stacking to that of Jensen's software, PROXIMITY. Due to the difficulty of isolating the pure training time of PROXIMITY, we only obtained a rough ratio of over 100, comparing the training time (including data loading) of PROXIMITY to that of our online stacked graphical learning.

[2] B. Taskar and P. Abbeel and D. Koller , *Discriminative Probabilistic Models for Relational Data*, Proceedings of UAI-02, Edmonton, Canada, 2002.

[3] M. Richardson and P. Domingos, *Markov Logic Networks*, Machine Learning, 62, pp107–136, 2006.

[4] Z. Kou and W. W. Cohen, *Stacked Graphical Learning for Efficient Inference in Markov Random Fields*, to appear in Proceedings of SDM-07, 2007.

[5] A. McCallum and C. Sutton, *Piecewise Training of Undirected Models*, Proceedings of UAI 2005.

[6] D. H. Wolpert, *Stacked generalization*, Neural Networks, vol. 5, pp241–259, 1992.

[7] Z. Kou, W. W. Cohen and R. F. Murphy, *Extracting Information from Text and Images for Location Proteomics*, Proceedings of the BIOKDD 2003.

[8] R. F. Murphy, Z. Kou, J. Hua, M. Joffe and W. W. Cohen, *Extracting and Structuring Subcellular Location Information from on-line Journal Articles: the Subcellular Location Image Finder*, Proceedings of KSCE-04, St. Thomas, US Virgin Islands, 2004.

[9] V. R. Carvalho and W. W. Cohen, *Single-Pass Online Learning: Performance, Voting Scheme and Online Feature Selection*, Proceedings of KDD-2006, Philadelphia, PA, 2006.

[10] M. Craven, et al., *Learning to Extract Symbolic Knowledge from the World Wide Web*, Proceedings of AAAI-98, Madison, WI, 1998.

[11] Q. Lu and L. Getoor, *Link-based Classification*, Proceedings of ICML-03, Washington, DC, 2003.

[12] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, *Automating the construction of internet portals with machine learning*, Information Retrieval, 3, pp127-63.

[13] C. L. Giles, K. Bollacker, and S. Lawrence, *Cite- Seer: An automatic citation indexing system*, ACM Digital Libraries 98.

[14] R. Bunescu, et al., *Comparative Experiments on Learning Information Extractors for Proteins and their Interactions*, Artificial Intelligence in Medicine, 33, 2 (2005), pp 139-155.

[15] N. Collier, et al., *The GENIA project: Corpus-based knowledge acquisition and information extraction from genome research papers*, Proceedings of EACL-99, pp271-272.

[16] K. Franzé, et al., *Protein names and how to find them*, International Journal of Medical Informatics, 2002, 67(1-3), pp49-61.

[17] W. W. Cohen and S. Sarawagi, *Exploiting Dictionaries in Named Entity Extraction: Combining Semi-Markov Extraction Processes and Data Integration Methods*, Proceedings of KDD 2004.

[18] Z. Kou, W. W. Cohen, R. F. Murphy, *A Stacked Graphical Model for Associating Sub-Images with Sub-Captions*, Proceedings of PSB 2007.

[19] Vitor R. Carvalho and William W. Cohen, *Learning to Extract Signature and Reply Lines from Email*, Proceedings of CEAS 2004 - First Conference on Email and Anti-Spam, Mountain View, CA, 2004.

[20] William W. Cohen and Vitor R. Carvalho,*Stacked Sequential Learning*, Proceedings of the IJCAI 2005.

[21] Andrew McCallum and Dayne Freitag and Fernando Pereira, *Maximum Entropy Markov Models for Information Extraction and Se gmentation*, Proceedings of the International Conference on Machine Lear ning (ICML-2000), Palo Alto, CA, 2000.

[22] Thomas G. Dietterich and Adam Ashenfelter and Yaroslav Bulatov, *Training conditional random fields via gradient tree boosting*, Proceedings of the Twenty-first International Conference (ICML-04), Banff, Alberta, Canada, 2004.

[23] Yoav Freund and Robert E. Schapire, *Large Margin Classification Using the Perceptron Algorithm*, Machine Learning,37(3), pp277–296, 1999.

[24] *Why Averaging Classifiers Can Protect Against Overfitting*, Yoav Freund and Yishay Mansour and Robert E. Schapire, AISTATS, 2001.

[25] Nick Littlestone, *Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm*, Machine Learning, 2(4), 1988.

[26] *Mistake-Driven Learning in Text Categorization*, I. Dagan and Y. Karov and D. Roth, EMNLP, 1997.