Combining Static Classifiers and Class Syntax Models for Logical Entity Recognition in Scanned Historical Documents

Song Mao[†] Praveer Mansukhani[‡] George R. Thoma[†]

[†]U.S. National Library of Medicine
Bethesda, Maryland 20894, USA
{smao,gthoma}@mail.nih.gov

[‡]Computer Science and Engineering Department,
University of Buffalo, Buffalo, NY, 14260, USA
pdm5@buffalo.edu

Abstract

Class syntax can be used to 1) model temporal or locational evolvement of class labels of feature observation sequences, 2) correct classification errors of static classifiers if feature observations from different classes overlap in feature space, and 3) eliminate redundant features whose discriminative information is already represented in the class syntax. In this paper, we describe a novel method that combines static classifiers with class syntax models for supervised feature subset selection and classification in unified algorithms. Posterior class probabilities given feature observations are first estimated from the output of static classifiers, and then integrated into a parsing algorithm to find an optimal class label sequence for the given feature observation sequence. Finally, both static classifiers and class syntax models are used to search for an optimal subset of features. An optimal feature subset, associated static classifiers, and class syntax models are all learned from training data. We apply this method to logical entity recognition in scanned historical U.S. Food and Drug Administration (FDA) documents containing court case Notices of Judgments (NJs) of different layout styles, and show that the use of class syntax models not only corrects most classification errors of static classifiers, but also significantly reduces the dimensionality of feature observations with negligible impact on classification performance.

1. Introduction and Prior Work

Temporal or locational relationship among objects and features extracted from the objects usually provide complementary information about their class labels. Temporal or locational relationship represented by class syntax models could place strong constraints on allowable class labels for a given sequence of objects. On the other hand, features extracted from objects could provide strong discriminative information for classifying the objects into different class labels. When class syntax models provide strong constraints on allowable class labels for a given sequence of objects, fewer features may be required to achieve satisfactory classification performance. Otherwise, more features may have to be used to achieve similar classification performance. Therefore, it is desirable to combine the power of both machineries for optimal classification and feature subset selection.

Class syntax models have been used in feature classification tasks in many applications. A hybrid SVM/HMM system has been used to combine the power of static classifiers (SVM) and class syntax models (HMM) for speech recognition [6]. However, SVM and HMM are used in separate steps and errors made in the first step cannot be corrected in the second step. Zheng et al. [15] proposed a machine printed text and handwritten text identification method in noisy images. They used Markov Random Field (MRF) to represent contextual information to improve feature classification accuracy. Heuristic rules are used to define MRFs in their approach.

Numerous algorithms have been proposed for logical labeling of documents [9]. Some are based on rules and others are based on deterministic grammatical models. Machine learning methods based on stochastic grammars have been proposed more recently. Shilman and Viola [13] proposed a document layout analysis method in which document layout is modeled by a non-generative grammar. Document layout is extracted by a global search of optimal parse based on a grammatical cost function. For each distinctive document class, feature subset and values of parameters are learned from corresponding training set. Chidlovskii and Fuselier [4] represent document structures by probabilistic context-free grammars and used a generalized probabilistic context-free parsing algorithm to annotate documents in a tree-like manner. However, grammatical models themselves are not learned in these methods and need to be manually specified for each document class. Bikel et al. [2] used a hidden

Markov model to learn to recognize and classify names, dates, times, and numerical quantities. The use of static classifiers for classification and feature subset selection are not addressed in this method. Conditional Random Field [8] has been proposed to relax independence assumptions and also solves label bias problem in most other models such as hidden Markov models and maximum entropy Markov models (MEMMs). While this model has more expressive power than HMMs, it has a more complicated parameter estimation process. We plan to combine CRFs with static classifiers for classification tasks that require more powerful language models than finite state automaton.

In this paper, we describe a method for supervised feature subset selection and classification using both static classifiers and class syntax models, and apply it for logical entity recognition in scanned historical documents. The novelty of our method is that 1) discriminative power of static classifiers and class syntax models are integrated in a unified algorithm for optimal classification, 2) both static classifiers and class syntax models are also used for optimal feature subset selection in a unified algorithm, and 3) an optimal feature subset, associated static classifiers, and class syntax models are all learned from training data. Incorporating class syntax models into the classification process makes some features redundant since their discriminative information is already represented in the class syntax models.

The remainder of this paper is organized as follows. In Section 2, we introduce static feature classifiers and class syntax models. In Section 3, we present our classification and feature selection algorithms. We report experimental results and provide detailed discussion in Section 4. Finally, we summarize our paper in Section 5.

2. Static Classifiers and Class Syntax Models

While many static classifiers could be used, we use Support Vector Machines (SVMs) [5]. SVMs achieve better classification performance by producing nonlinear class boundaries in the original feature space by constructing linear space in a larger and transformed version of the original feature space.

Formally, given a set of N pairs of training data $\{(\mathbf{x}_i, y_i), i = 1, \ldots, N\}$ where $\mathbf{x}_i \in \mathbb{R}^v$ is a v-dimensional feature vector in $\mathbb{R}^v$, a v-dimensional Euclidean space, and $y_i \in \{-1, 1\}$ is the class label of $\mathbf{x}_i$. Define a hyperplane in the transformed feature space as $f(\mathbf{x}) = h(\mathbf{x})^T\beta + \beta_0 = 0$ where $h(\mathbf{x}_i) = \{h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \ldots, h_M(\mathbf{x}_i)\}$, $i = 1, \ldots N$ and $h_m(\mathbf{x}), m = 1, \ldots M$ is a set of M basis functions. The goal is to find the hyperplane that produces the largest margin between training features of class 1 and -1. If class overlap is allowed in feature space, define a set of slack variables $\xi_1, \ldots, \xi_N$, the optimization problem is

$$\begin{aligned} &\min_{\beta\beta_0} \tfrac{1}{2}||\beta||^2 + \gamma \textstyle\sum_{i=1}^{N} \xi_i \\ \text{subject to } \; &y_i(h(\mathbf{x}_i)^T\beta + \beta_0) \geq 1 - \xi_i, \xi_i \geq 0, \forall i. \end{aligned}$$

In the transformed feature space, if feature vector $h(\mathbf{x}_i)$ is on the wrong side of the margin, it has positive ξ_i, if it is on the correct side, $\xi_i = 0$. It can be shown that the solution to the above optimization problem only depends on the inner product of basis function $h(\mathbf{x})$, i.e., its kernel function $K(\mathbf{x}, \mathbf{x}^{'}) = \langle h(\mathbf{x}), h(\mathbf{x}^{'})\rangle$. We selected a kernel function based on the radial basis functions $K(\mathbf{x}, \mathbf{x}^{'}) = \exp(-||\mathbf{x} - \mathbf{x}^{'}||^2/c)$, where $c > 0$. Note that γ and c are kernel parameters and need to be determined in the training step. The classification rule is $G(\mathbf{x}) = \text{sign}[f(\mathbf{x})]$. To extend two-class classification to multi-class classification, we use the "one-against-one" approach in which $K(K-1)/2$ two-class classifiers are constructed, where K denotes the total number of classes. A voting strategy used to select the class label of each feature vector is as follows: each two-class classification result is a vote on each feature vector point, and the one that has the most votes is the class label of the feature vector.

While SVMs are static classifiers that have strong discriminative power, they cannot model temporal or locational evolvement of class labels of feature observations. In hidden Markov models (HMMs), a feature observation sequence is characterized as a parametric random process. The class label $y_t \in \{1, \ldots, K\}$ of a feature observation $\mathbf{x}_t \in \mathbb{R}^v$ of a feature observation sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is defined as the state of a system at discrete time $t, 1 \leq t \leq T$. A discrete time, first order, Markov chain is used to specify the probabilistic dependency of current state at time t and its preceding state at $t-1$. In other words, it is assumed that the class label of the current feature observation depends only on that of the preceding feature observation. This has the effect of characterizing class label sequences as the sentences generated by a stochastic finite state automaton, or equivalently, stochastic regular grammar. Finally, states in an HMM are hidden and only feature observations are observable, and they are probabilistic functions of the states (or class label). The Viterbi algorithm [12] can be used to search for the optimal state (class label) sequence for a given feature observation sequence.

While HMMs place constraints on allowable class label sequences for a given feature observation sequence, they have weak discriminative power. In order to combine the power of the two machines, we first compute posterior class probabilities given feature observations from outputs of SVMs, which are then integrated in the Viterbi algorithm. We will also show that class syntax models can be used to reduce the dimensionality of feature observations.

3. The Algorithms

In this section, we first describe an algorithm that combines static classifiers with class syntax models for optimal classification. We then describe an algorithm for selecting an optimal subset of features.

3.1. Classification Algorithm

Let $\lambda = (A, B, \pi)$ be a hidden Markov model (HMM) where $A = \{a_{ij} = P[y_{t+1} = j | y_t = i], 1 \leq i, j \leq K\}$ is the state transition probability matrix that defines a discrete time, first order Markov chain, and $P[y_{t+1} = j | y_t = i]$ is assumed to be independent of time t, $B = \{b_i(\mathbf{x}_t) = f(\mathbf{x}_t | y_t = i), 1 \leq i \leq K\}$ is a vector of conditional feature observation probability density functions, and $\pi = \{\pi_i = P[y_1 = i], 1 \leq i \leq K\}$ is initial state distribution. For a given sequence of *continuous* feature observations $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ of length T, we want to predict their *discrete* class label sequence $\mathbf{y} = (y_1, \ldots y_T)$, where $\mathbf{x}_t \in \mathbb{R}^v$, $y_t \in \{1, \ldots, K\}$, $t \in \{1, \ldots, T\}$, K is the total number of classes, and v is the dimensionality of feature observation. The optimal class label sequence is found by the Viterbi algorithm as follows.

Define $\delta_t(j) = \max_{y_1, y_2, \ldots, y_{t-1}} f(y_1, y_2, \ldots, y_{t-1}, y_t = j, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t | \lambda)$ be the highest joint probability density path ending in state j at time t. Let $\omega_t(j) = \delta_t(j)/(f(\mathbf{x}_t) f(\mathbf{x}_{t-1}) \ldots f(\mathbf{x}_1))$. The steps of the algorithm is as follows:

• Initialization:

$$\begin{aligned}
\delta_1(i) &= f(y_1 = i, \mathbf{x}_1 | \lambda) \\
&= P[y_1 = i] f(\mathbf{x}_1 | y_1 = i) = \pi_i b_i(\mathbf{x}_1) \\
&= \pi_i \frac{P[y_1 = i | \mathbf{x}_1] f(\mathbf{x}_1)}{P[y_1 = i]} \\
\Rightarrow \omega_1(i) &= \frac{\delta_1(i)}{f(\mathbf{x}_1)} = \pi_i \frac{P[y_1 = i | \mathbf{x}_1]}{P[y_1 = i]}, \\
\psi_1(i) &= 0, 1 \leq i \leq K.
\end{aligned}$$

Note that π_i is not canceled out with $P[y_1 = i]$ here since π_i and the quantity

$$\frac{P[y_t = i | \mathbf{x}_t]}{P[y_t = i]}$$

are estimated and computed separately as will be shown in Equation 9 and Equations 5- 8, respectively.

• Recursion:

$$\begin{aligned}
\delta_t(j) &= \max_{1 \leq i \leq K} [\delta_{t-1}(i) a_{ij}] b_j(\mathbf{x}_t) \\
&= \max_{1 \leq i \leq K} [\delta_{t-1}(i) a_{ij}] f(\mathbf{x}_t | y_t = j) \\
&= \max_{1 \leq i \leq K} [\delta_{t-1}(i) a_{ij}] \frac{P[y_t = j | \mathbf{x}_t] f(\mathbf{x}_t)}{P[y_t = j]} \\
\Rightarrow \frac{\delta_t(j)}{f(\mathbf{x}_t)} &= \max_{1 \leq i \leq K} [\delta_{t-1}(i) a_{ij}] \frac{P[y_t = j | \mathbf{x}_t]}{P[y_t = j]} \\
\Rightarrow \omega_t(j) &= \frac{\delta_t(j)}{f(\mathbf{x}_t) f(\mathbf{x}_{t-1}) \ldots f(\mathbf{x}_1)} \\
&= \max_{1 \leq i \leq K} [\frac{\delta_{t-1}(i)}{f(\mathbf{x}_{t-1}) \ldots f(\mathbf{x}_1)} a_{ij}] \frac{P[y_t = j | \mathbf{x}_t]}{P[y_t = j]} \\
&= \max_{1 \leq i \leq K} [\omega_{t-1}(i) a_{ij}] \frac{P[y_t = j | \mathbf{x}_t]}{P[y_t = j]} \qquad (1) \\
\psi_t(j) &= \arg \max_{1 \leq i \leq K} [\delta_{t-1}(i) a_{ij}] \\
&= \arg \max_{1 \leq i \leq K} [\omega_{t-1}(i) a_{ij}], \\
&\quad 2 \leq t \leq T, 1 \leq j \leq K. \qquad (2)
\end{aligned}$$

• Termination:

$$P^* = \max_{1 \leq i \leq K} [\delta_T(i)] = \max_{1 \leq i \leq K} [\omega_T(i)], \qquad (3)$$

$$y_T^* = \arg \max_{1 \leq i \leq K} [\delta_T(i)] = \arg \max_{1 \leq i \leq K} [\omega_T(i)]. \qquad (4)$$

• Path (state sequence) backtracking:

$$y_t^* = \psi_{t+1}(y_{t+1}^*), t = T-1, T-2, \ldots, 1.$$

Now the recursive relationship in terms of δ's is rewritten as that of ω's in Equation (1) so that neither $f(\mathbf{x}_t)$'s nor $b_j(\mathbf{x}_t)$'s are involved in the computation, therefore we do not need to specify their explicit probability density functions. The derivations in (1), (2), (3), and (4) rely on the fact that $f(\mathbf{x}_t)$'s are constant with respect to variable i.

We now approximate the stochastic quantity in Equation (1) by a static one as

$$\frac{P[y_t = j | \mathbf{x}_t]}{P[y_t = j]} \approx \frac{P[y = j | \mathbf{x}_t]}{P[y = j]}, \qquad (5)$$

and the recursion relationship in Equation 1 becomes

$$\omega_t(j) = \max_{1 \leq i \leq K} [\omega_{t-1}(i) a_{ij}] \frac{P[y = j | \mathbf{x}_t]}{P[y = j]}.$$

While this quantity loses stochastic property through the approximation in Equation 5, the parameters π_i's and a_{ij}'s in our algorithm are still used control allowable class label sequences for a given input feature observation sequence.

To compute $P[y = j | \mathbf{x}_t]$, the probability of assigning the observation $\mathbf{x}_t$ the class j where $j \in \{1, \ldots, K\}$, we use

the method described in [14] that employs support vector machines (SVMs) as the binary classifier. In SVMs binary classification, the one-against-one approach is used to compute pairwise class probabilities $r_{ij} \approx P[y = i|y = i \text{ or } j, \mathbf{x}]$ where

$$r_{ij} \approx \frac{1}{1 + e^{\alpha \hat{f} + \beta}} \tag{6}$$

is an improved implementation of [10]. Parameters α and β are estimated using training data and their decision values $\hat{f}$ obtained in cross validation. The Algorithm 2 described in [14] is used to compute $P_j = P[y = j|\mathbf{x}]$ from r_{ij}'s by solving the following optimization problem

$$\min_{P_1,\ldots,P_K} \frac{1}{2} \sum_{i=1}^{K} \sum_{j:j \neq i} (r_{ji} P_i - r_{ij} P_j)^2, \tag{7}$$

subject to $\sum_{i=1}^{K} P_i = 1, P_j \geq 0, 1 \leq j \leq K$. We see that by computing $P[y = j|\mathbf{x}]$ using the outputs of SVMs, the discriminative classification power of SVMs is integrated into our algorithm through approximation in Equation 5.

3.2. Parameter Estimation

Given a training dataset of N sequences of feature observations $\{(\mathbf{x}_1^n, \ldots, \mathbf{x}_{T(n)}^n), 1 \leq n \leq N\}$ and their true class label sequences $\{(y_1^n, \ldots, y_{T(n)}^n), 1 \leq n \leq N\}$, we first estimate α, β and learn decision rule $\hat{f}$ for computing r_{ij} in Equation 6 using the method in [10]. We then compute maximum likelihood estimate of $P[y = j]$ using simple counts as

$$\hat{P}[y = j] = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T(n)} (y_t^n = j)}{\sum_{n=1}^{N} \sum_{t=1}^{T(n)} \sum_{i=1}^{K} (y_t^n = i)}, 1 \leq j \leq K. \tag{8}$$

Finally, we compute maximum likelihood estimates of initial probabilities π_i's and state transition probabilities a_{ij}'s using simple counts as

$$\hat{\pi}_i = \frac{\sum_{n=1}^{N} (y_1^n = i)}{\sum_{j=1}^{K} \sum_{n=1}^{N} (y_1^n = j)}, \tag{9}$$

$$\hat{a}_{ij} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T(n)} (y_t^n = j, y_{t-1}^n = i)}{\sum_{n=1}^{N} \sum_{s=1}^{T(n)} (y_s^n = i)}, 1 \leq i, j \leq K. \tag{10}$$

3.3. Feature Subset Selection Algorithm

Each feature observation is a v-dimensional feature vector. In the training phase, we search for an optimal subset of features S^* of size $d < v$ such that the following error function is minimized:

$$S^* = \arg \min_{|S|=d, S \subseteq F, d<v} \sum_{n=1}^{N} \sum_{t=1}^{T(n)} (y_t^{*n}(S) \neq \bar{y}_t^n(S)).$$

where F is the original set of features, and $y_t^{n*}(S)$'s and $\bar{y}_t^n(S)$'s are optimal class labels found using the classification algorithm in Section 3.1 and true class labels from the training dataset, respectively, for the given feature observations $\{\mathbf{x}_1^n(S), \ldots, \mathbf{x}_{T(n)}^n(S), 1 \leq n \leq N\}$ in the training dataset. Let

$$ALG(\{\mathbf{x}_1^n(S), \ldots, \mathbf{x}_{T(n)}^n(S)\}) = \{y_1^{*n}(S), \ldots, y_{T(n)}^{*n}(S)\},$$

where $ALG(\cdot)$ denotes the classification algorithm in Section 3.1 and $\mathbf{x}_m^n(S), 1 \leq n \leq N, 1 \leq m \leq T(n)$ is a d dimensional feature observation whose features are defined by S. Note that we now explicitly express feature observations and their class labels as functions of a feature set. Many researchers have studied feature subset selection. Jain and Zongker [7] conducted a survey and comparative evaluation of feature subset selection methods and identified the sequential forward floating selection (SFFS) [11] algorithm as the most effective one. We use this algorithm to search for an optimal subset of features of size d, where $\sum_{n=1}^{N} \sum_{t=1}^{T(n)} (y_t^{*n}(S) \neq \bar{y}_t^n(S))$ is the criterion to be minimized. Note that this criterion value combines the power of both static classifiers (SVMs) and class syntax models (HMMs).

3.4. Complexity Analysis

Since we are concerned about performance we have investigated the complexity of our method. The classification algorithm described in Section 3.1 requires an order of K^2T computations without considering the computation complexity for computing $P[y = j|\mathbf{x}_t]$. The complexity for computing $P[y = j|\mathbf{x}_t], j = 1, \ldots, K$ is $O(KI)$ where I is the total number of iterations need to reach the solution to Equation 7 in Algorithm 2 described in [14]. I is controlled by a stop criterion in the algorithm. Therefore, the computational complexity for computing $P[y = j|\mathbf{x}_t], y = 1, \ldots K, t = 1, \ldots, T$ is $O(KIT)$. Since SVM computations are only used to compute r_{ij} and are one-time computations, the total computations are $K(K-1)$. Therefore, the algorithm complexity is $O((K^2 + KI) \cdot T + K^2 - K) = O((K^2 + KI) \cdot T)$.

4. Experiments

4.1. Dataset

We apply our algorithm to logical entity recognition in scanned historical U.S. Food and Drug Administration (FDA) documents called *Notices of Judgment* (NJs) which detail court cases concerning violations related to food, drug, and cosmetics by companies in the early twentieth century. Each NJ 1) is printed in small and old-styled fonts, and have different layout styles, 2) could start or end anywhere in a document page, and 3) could occupy a partial

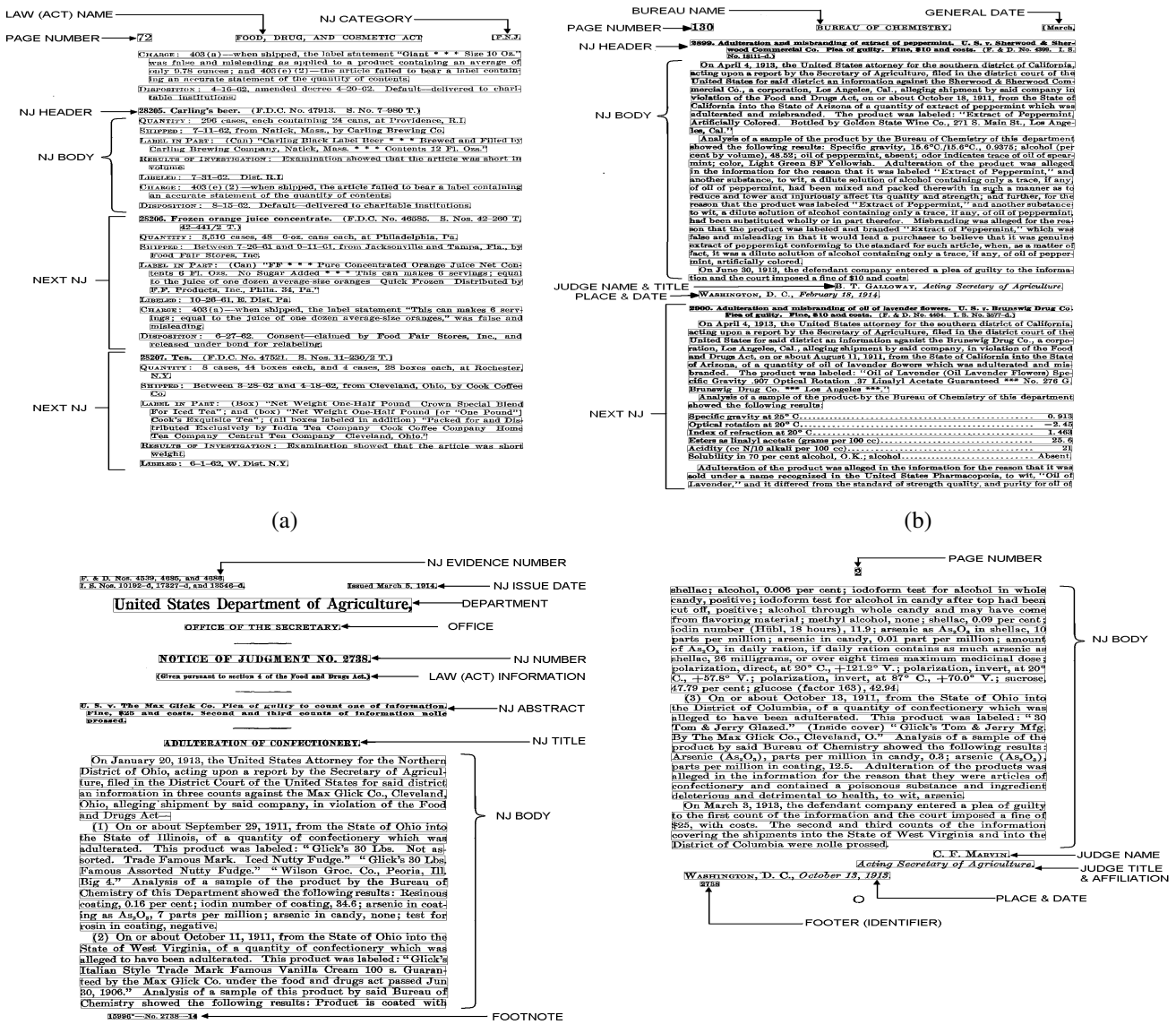

Figure 1. Notices of Judgment in three different styles. Note that an NJ occupies two pages in style (c). Some logical entities are associated with each NJ. Others with a set of NJs such as law (act) name, and NJ category in style (a), bureau name, and general date in style (b). The logical entity that is associated with a document page is the page number. Logical entities associated with each NJ include NJ header and NJ body in style (a), NJ header, NJ body, judge name and title, place and date in style (b), and NJ evidence number, NJ issue date, department, office, NJ number, law (act) information, NJ abstract, NJ title, NJ body, footnote, judge name, judge title and affiliation, place and date in style (c).

page or multiple pages. NJs showing three layout styles and logical entities of interest appear in Figure 1. The number of logical entities in each NJ increases from style (a) to (c).

For each layout style, we randomly select a set of document pages as the training dataset. In our experiments, we choose the first 30 pages as the training dataset for layout style (b), and (c). Since there are two variations in layout style (a), we choose the first 30 pages from the first variation and the first 15 pages from the second variation, and consider these 45 pages as the training dataset. The remaining pages in each layout style are considered as the test dataset for that style. Textlines are basic classification units and there are a total of 26,282 textlines in 955 NJs from 518 pages in our dataset as summarized in Table 1.

4.2. Textline Segmentation and Feature Extraction

While numerous algorithms can be used to perform page segmentation, the zoning results and character bounding

Table 1. Training and test datasets in our experiments.

Layout style	Dataset Pages;NJs;Textlines	Training set Pages;NJs;Textlines	Test set Pages;NJs;Textlines
(a)	234;648;10,947	45;94;1,423	189;554;9,524
(b)	225;269;13,495	30;28;1,849	195;241;11,646
(c)	59;38;1,840	30;15;929	29;23;911

box information generated by the FineReader OCR engine 8.0 [1] are used in our experiment to obtain textline segmentation from scanned document pages. Figure 1 shows examples of such textline segmentation results in the various layout styles. They are ordered from left to right and top to bottom to form a one-dimensional string of textlines. The fourteen features extracted from each textline are as follows: 1: average ratio of black pixels in character bounding boxes in the textline; 2-5: mean of character width, height, aspect ratio, and area; 6-9: variance of character width, height, aspect ratio, and area; 10: total number of letters and digits/total number of characters; 11: total number of letters/total number of letters and digits; 12: total number of capital letters/total number of letters; 13-14: indentation (where 00 denotes center line, 10 denotes left indented line, 11 denotes full line (i.e. those textlines occupying the full range of a text column), and 01 denotes right indented line). Thus the 13th feature value could indicate if the line touches the left margin of a text column, and the 14th feature value could indicate if the line touches the right margin of a text column.

4.3. Training Results

Textline classification accuracy is the performance metric used in both the training and test phases of our experiments. In the training phase, we learn a class syntax model (HMMs), an optimal feature subset, and associated static classifiers (SVMs) for each layout style, all from the training dataset as described in Sections 3.2 and 3.3. To search for an optimal feature subset of size $1 \leq d \leq v = 14$, we first train an SVM as follows: do a grid search on its tunable parameters γ and c and find the one set that gives the highest five-fold cross validation accuracy, which is then used to train the SVM on the whole training set to get the final SVM classifier. The LIBSVM library [3] is used to train SVM classifiers and compute class posterior probabilities. The posterior probabilities are used in our classification algorithm to obtain final classification accuracy, which serves as the criterion value for the SFFS [11] algorithm. Finally, the SFFS algorithm is used to find an optimal feature subset of size d that maximizes this criterion value. SVM training and feature subset selection using SFFS are relatively slow, but they are done only once and results are saved. Table 2 to 4 show the training results and Figure 2 shows class syntax models learned for the three layout styles.

Table 2. Training results for NJs of layout (a).

Optimal feature subset	SVM accuracy	Algorithm accuracy	SVM errors	Corrected by HMM	Algorithm errors	% Error reduction
11	88.83%	94.38%	159	79	80	49.68%
3,7	96.77%	98.95%	46	31	15	67.39%
3, 7, 11	98.24%	99.16%	25	13	12	52.00%
3, 7, 11, 13	98.52%	98.95%	21	6	15	28.57%
3, 4, 7, 9, 11	99.16%	99.65%	12	7	5	58.33%
3,7,8,11,12,13	99.16%	99.51%	12	5	7	41.67%
2,3,6,7,8,11,13	99.16%	99.58%	12	6	6	50.00%
1,2,3,7,8,11,12,13	99.09%	99.37%	13	4	9	30.77%
1,3,5,6,7,8,11,12,13	99.23%	99.58%	11	5	6	45.45%
1,3,5,6,7,8,9,11,12,13	99.23%	99.58%	11	5	6	45.45%
1,2,3,4,6,7,8,9,11,12,13	99.23%	99.58%	11	5	6	45.45%
1,2,3,5,6,7,8,9,10,11,12,13	99.30%	99.44%	10	2	8	20.00%
1,2,3,4,5,6,7,8,9,10,11,12,13	99.51%	99.65%	7	2	5	28.57%
all	99.65%	99.86%	5	3	2	60.00%

Table 3. Training results for NJs of layout (b).

Optimal feature subset	SVM accuracy	Algorithm accuracy	SVM errors	Corrected by HMM	Algorithm errors	% Error reduction
3	93.35%	98.92%	123	103	20	83.74%
3,12	97.13%	99.51%	53	44	9	83.02%
3, 11, 12	99.13%	99.89%	16	14	2	87.50%
3, 7, 11, 12	99.41%	99.78%	11	7	4	63.64%
3, 7, 11, 12, 14	99.78%	100%	4	4	0	100.00%
3,7,10,11,12,14	99.73%	99.89%	5	3	2	60.00%
3,5,7,10,11,12,14	99.84%	100%	3	3	0	100.00%
3,5,6,7,10,11,12,14	99.68%	100%	6	6	0	100.00%
1,2,3,6,7,10,11,12,14	99.89%	100%	2	2	0	100.00%
1,2,3,5,6,7,10,11,12,14	99.78%	100%	4	4	0	100.00%
1,2,3,4,5,6,9,10,11,12,14	99.73%	100%	5	5	0	100.00%
1,2,4,5,6,7,9,10,11,12,13,14	99.89%	100%	2	2	0	100.00%
1,2,3,4,5,6,7,8,9,10,11,12,13	99.84%	100%	3	3	0	100.00%
all	99.84%	99.89%	3	1	2	33.33%

Table 4. Training results for NJs of layout (c).

Optimal feature subset	SVM accuracy	Algorithm accuracy	SVM errors	Corrected by HMM	Algorithm errors	% Error reduction
3	89.67%	99.57%	96	92	4	95.83%
3,12	97.74%	100%	21	21	0	100.00%
5, 7, 11	99.14%	99.78%	8	6	2	75.00%
3, 10, 12, 13	99.78%	100%	2	2	0	100.00%
2, 3, 7, 11, 12	99.78%	100%	2	2	0	100.00%
3,5,7, 9,11,12	99.57%	100%	4	4	0	100.00%
2,3,7,8,10,12,13	99.89%	100%	1	1	0	100.00%
1,3,4,7,10,11,12,13	100%	100%	0	0	0	0.00%
1,4,5,7,8,9,11,12,14	100%	100%	0	0	0	0.00%
1,3,4,5,6,7,9,11,12,13	100%	100%	0	0	0	0.00%
1,3,4,5,6,7,9,10,11,12,13	100%	100%	0	0	0	0.00%
1,3,4,5,6,7,8,9,10,11,12,13	100%	100%	0	0	0	0.00%
1,2,3,4,5,6,7,8,9,10,11,12,13	100%	100%	0	0	0	0.00%
all	100%	100%	0	0	0	0.00%

4.4. Test Results

Our algorithm is tested on the test dataset using optimal feature subsets found in the training phase. Each textline is classified into one of logical entities shown in Figure 1. In order to compare the effectiveness of class syntax models to correct classification errors made by static classifiers on NJ body and NJ non-body textlines, we sum up the numbers for textlines classified as one of NJ non-body types such as NJ issue date, NJ header, page number, etc. We summarize the test results in Table 5. We see that our algorithm achieves a 98.06% classification accuracy using only five features (3,4,7,9,11) for layout style (a), achieved a 99.67% classification accuracy using only five features (3,7,11,12,14) for layout style (b), and achieved a 99.34% classification accuracy using only two features (3, 12) for layout style (c). Most textlines in our dataset are NJ body textlines, and we see most of them are correctly classified by SVMs alone simply because the data used for training the SVMs is extensive. We also see class syntax models (HMMs) are

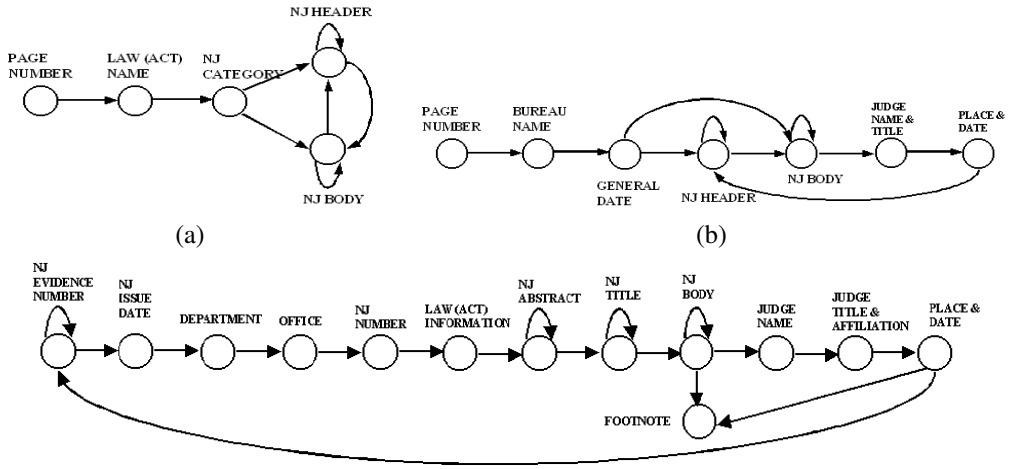

Figure 2. A class syntax model is learned for each layout style.

Table 5. Test results.

Layout style	Optimal feature subset	Textlines	SVM accuracy	Algorithm accuracy	SVM errors	Corrected by HMM	Algorithm errors	% Error reduction
(a)	3, 4, 7, 9, 11	Total: 9,524	96.26%	98.06%	356	171	185	48.03%
		NJ body: 7,798	99.36%	99.58%	50	17	33	34.00%
		NJ non-body: 1,726	82.27%	91.19%	306	154	152	50.33%
(b)	3, 7, 11, 12, 14	Total: 11,646	98.05%	99.67%	227	189	38	83.26%
		NJ body: 10,179	99.47%	99.65%	54	18	36	33.33%
		NJ non-body:1,467	88.21%	99.86%	173	171	2	98.84%
(c)	3, 12	Total: 911	93.85%	99.34%	56	50	6	89.29%
		NJ body:693	98.85%	100%	8	8	0	100.00%
		NJ non-body:218	77.98%	97.25%	48	42	6	87.5%

most effective to correct classification errors on NJ non-body textlines in layout style (a) and (b). SVMs alone show low classification performance on NJ non-body textlines since the size of the training data is relatively small and feature observations from different logical categories overlaps in feature space. Class syntax models place constraints on allowable class sequence for NJ non-body textlines. The strength of such constraints seems to be stronger for style (c) and only two features are needed to achieve an optimal training classification accuracy of 99.34%.

In Figure 3, we compare classification performance of SVMs and our algorithm on NJ body and NJ non-body textlines using optimal feature subsets of all sizes learned in the training phase. We see that 1) SVMs require many more features to achieve similar performance to that of our algorithm, 2) for all layout styles, the classification performance of our algorithm quickly converges with an optimal feature subset of only a few features, and 3) the shapes of the overall curves depend on those for NJ non-body textlines, for which class syntax models (HMMs) are most effective to correct misclassification errors.

5. Summary and Future Work

We have described a method that combines static classifiers (SVMs) and class syntax models (HMMs) for optimal classification and feature subset selection. This technique was applied to logical entity recognition in scanned historical documents. We showed in our experiments that the use of class syntax models not only corrects most classification errors created by static classifiers, but also significantly reduces the dimensionality of feature observations with negligible impact on classification performance. Future work includes studying the relationship between the complexity of class syntax models and its effect on dimensionality reduction of feature observations, and use of more powerful class syntax models such as conditional random fields.

References

[1] ABBYY FineReader OCR 8.0, 2006.

[2] D. M. Bikel, R. Schwartz, and R. M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34:211–231, 1999.

[3] C. C. Chang and C. J. Lin. LIBSVM: a library for support vector machines, 2001.

[4] B. Chidlovskii and J. Fuselier. A probabilistic learning method for XML annotation of documents. In *International Joint Conferences on Artificial Intelligence*, pages 1016–1021, Edinburgh, Scotland, July 2005.

[5] C. Cortes and V. Vapnik. Support-vecor network. *Machine Learning*, 20:273–297, 1995.

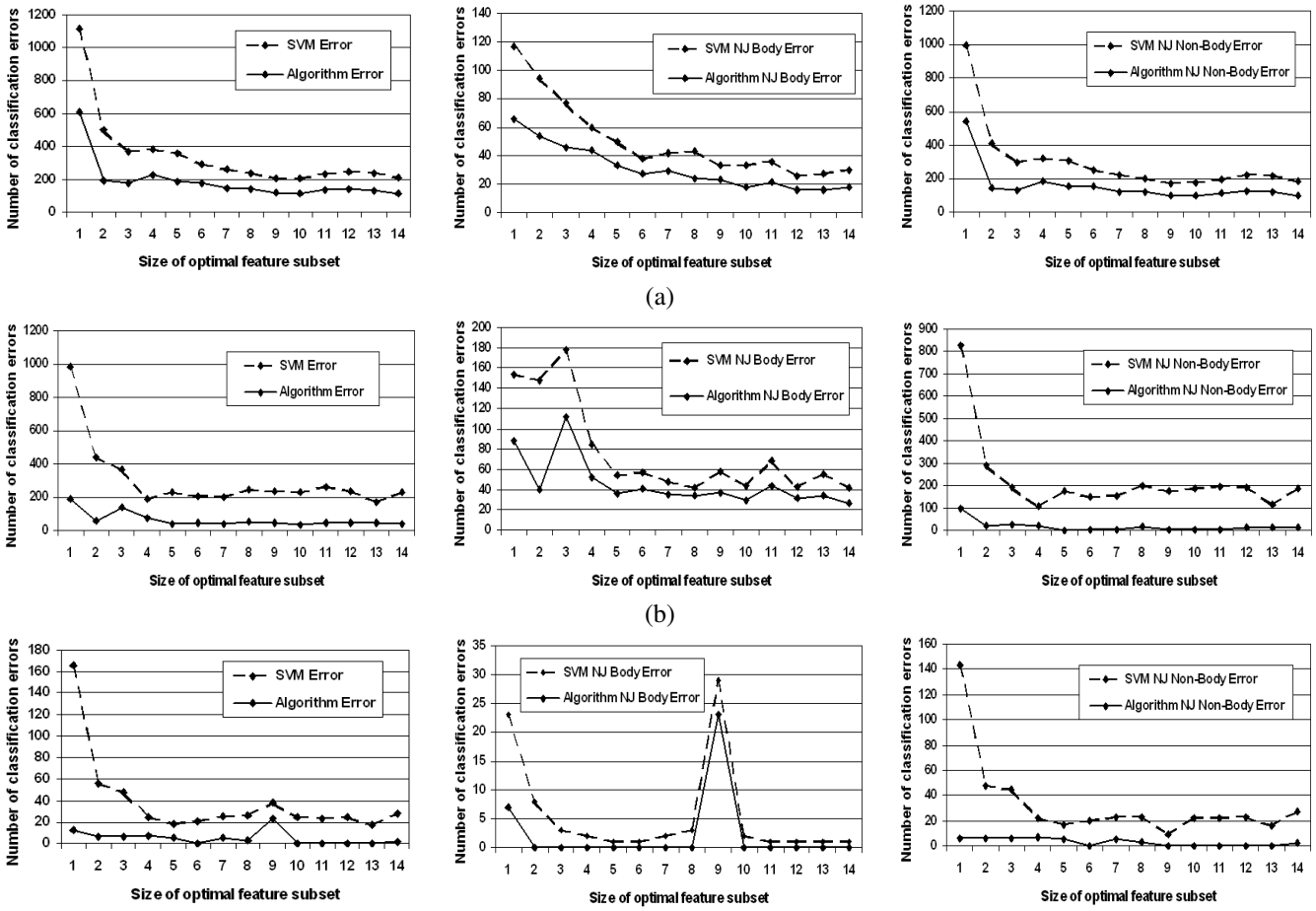

Figure 3. Classification performance of static classifiers (SVMs) and our algorithm on three layout styles. The vertical distance between the two curves represents the number of errors corrected by our algorithm using class syntax model HMMs. Note that the scales of plots are different and the number of classification errors on NJ body textlines are much less than that on NJ non-body textlines. Our algorithm achieves much greater reduction of classification errors on NJ non-body textlines than on NJ body textlines.

[6] A. Ganapathiraju, J. Hamaker, and J. Picone. Hybrid SVM/HMM architectures for speech recognition. In *2000 Speech Transcription Workshop*, College Park, Maryland, May 2000.

[7] A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:153–158, 1997.

[8] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of International Conference on Machine Learning*, pages 282–289, Williams College, MA, June 2001.

[9] S. Mao, A. Rosenfeld, and T. Kanungo. Document structure analysis: A survey. In *Proceedings of SPIE Conference on Document Recognition*, pages 197–207, San Jose, CA, January 2003.

[10] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, Cambridge, MA, 2000.

[11] P. Pudil, J. Novovicova, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125, 1994.

[12] L. Rabiner and B. H. Juang. *Fundamentals of Speech recognition*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[13] M. Shilman, P. Liang, and P. Viola. Learning non-generative grammatical models for document analysis. In *Proceedings of IEEE International Conference on Computer Vision*, pages 962–969, Beijing, China, October 2005.

[14] T. F. Wu, C. J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.

[15] Y. Zheng, H. Li, and D. Doermann. Machine printed text and handwriting identification in noisy document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:337–353, 2004.