Document Content Extraction Using Automatically Discovered Features

Sui-Yu Wang and Henry S. Baird and Chang An

Computer Science & Engineering Dept., Lehigh University
19 Memorial Dr West, Bethlehem, PA 18017 USA

E-mail: syw2@lehigh.edu, baird@cse.lehigh.edu, cha305@lehigh.edu

Abstract

We report an automatic feature discovery method that achieves results comparable to a manually chosen, larger feature set on a document image content extraction problem: the location and segmentation of regions containing handwriting and machine-printed text in documents images. As first detailed in [17], this approach is a greedy forward selection algorithm that iteratively constructs one linear feature at a time. The algorithm finds error clusters in the current feature space, then projects one tight cluster into the null space of the feature mapping, where a new feature that helps to classify these errors can be discovered. We conducted experiments on 87 diverse test images. Four manually chosen linear features with an error rate of 16.2% were given to the algorithm; the algorithm then found an additional ten features; the composite 14 features achieved an error rate of 13.8%. This outperforms a feature set of size 14 chosen by Principal Component Analysis (PCA) with an error rate of 15.4%. It also nearly matches the error rate of 13.6% achieved by twice as many manually chosen features. Thus our algorithm appears to compete with both the widely used PCA method and tedious and expensive trial-and-error manual exploration.

1. Introduction

For many document image classification problems, engineers pick features using knowledge of the specific problem, but occasionally without fully understanding why some work while others don't.

Feature selection algorithms attempt to identify a small set of features that still discriminates well. Such methods can be divided into three categories: filters, wrappers, and embedded methods [10]. Filters rank features according to their individual discriminability, then pick the top performing features; in [14], Nigam et al. used maximum entropy to estimate how important a feature is. However, features that are promising in isolation don't necessarily work well in combination [9, 7]. Wrappers use the classification algorithms to evaluate subset performance; however, wrappers can perform an exhaustive search in the worst case. Embedded methods are similar to wrappers, but use more efficient search methods to avoid an exhaustive search. Adaboost, when used for feature selection, is an embedded algorithm: Adaboost weights features and chooses features with larger weights [16]. Among wrapper methods (e.g., Partial Least Square [18, 19]) and embedded methods (e.g., AdaBoost [16] or our method), greedy methods are most popular. Forward selection usually ranks features and selects a smaller subset than backward elimination. Also, forward selection has more stable performance when the size of the resulting subset is small.

The dimensionality of feature space is sometimes too large for most feature selection algorithms to work efficiently. Researchers often reduce the dimensionality using methods like PCA [15] before applying feature selection algorithms. PCA assumes that features with higher variance are potentially more interesting. However, higher variances do not imply higher discriminability [8]. Applying PCA before applying feature selection risks throwing away potentially discriminating information.

For problems with dimensionality so high that it prohibits direct application of feature selection algorithms, we propose a forward selection method. Our algorithm incrementally adds one new feature at a time, until the desired error rate is reached, without throwing any information away beforehand.

Our feature selection algorithm is detailed in [17], here we briefly review it. We assume that we are working on a two-class problem and that we are given an initial set of features on which an Nearest Neighbor (NN) classifier has been trained. Suppose the performance of this classifier is not satisfactory. Our algorithm uses tight clusters of errors of both types to guide the search for new features. Tight clusters of errors indicate that the current feature set is not discriminating within these regions. In order to re-

solve these errors, we project one cluster into the null space, where we find a decision hyperplane and define the new feature to be a sample's directed distance to the hyperplane. Projecting only samples in the clusters into null space is computationally efficient and allows us to find more precise decision boundaries. Such a process also guarantees that any feature found in the null space is orthogonal to all current features. As we show in Section 2, this method works well if the features are linear.

We have conducted a large-scale experiment that is more than ten times the size of our previously reported experiment [17], and includes more variety of documents. Our experimental framework, document image content extraction, will be described in detail in Section 3. In order to make progress on this difficult problem, we have spent hundreds of hours in a tedious manual search for features such as the average lightness of horizontal lines. We expect many other researchers have had similar experience. Although it is not required that any feature be given to the algorithm initially, we start the experiment with four manually chosen features so that expert knowledge on features can be incorporated. Four manually chosen linear features yield an error rate of 16.2% on 87 diverse test images from various sources like newspaper, hand-written notes, etc. Our algorithm takes these four features and generates a sequence of ten linear features and achieves an error rate of 13.8%. Another set of 28 features, both linear and nonlinear, manually chosen and tested over a period of two years, has an error rate of 13.6% on the same test set. A set of 14 features chosen by PCA has an error rate of 15.4%. The original four manual features with three discovered features, a set of seven features that is half the size of the PCA set, already outperformed PCA features with an error rate of 14.8%. Thus our automatic feature discovery method achieves results comparable to a manually chosen, larger feature set.

2. Formal Definitions

We work on a 2-class problem. We assume that there is a source of labeled training samples $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, ...\}$. Each sample $\mathbf{x}$ is described by a large number D of real-valued features: i.e., $\mathbf{x} \in \mathbf{R^D}$. But D, we assume, is too large for use as a classifier feature space. We also have a much smaller set of d real-valued features $\mathbf{f}^d = \{f_1, f_2, ..., f_d\}$, where $d << D$. Applying $\mathbf{f}^d$ on sample $\mathbf{x}$ we get a d-dimensional vector $\mathbf{f}^d(\mathbf{x}) = (x_1, x_2, ..., x_d)$.

We use the performance of the current classifier to guide our search for new features. We identify clusters of errors in the feature space. Afterwards, we select one cluster that is both "balanced" — the number of errors from both types are similar — and "tight," i.e., with small average pairwise distance. We assume data is drawn randomly such that the data fills the sample space with probability density functions similar to the underlying distribution. Thus a tighter cluster implies that by correctly classifying samples in the region, more errors are likely to be corrected. Different clusters of errors might come from different decision boundary segments. Thus we consider one cluster at a time, hoping to resolve *some* errors.

In the original sample space $\mathbf{R}^D$, there exists some kind of boundary between classes. After projecting the data by $\mathbf{f}^d$ into $\mathbf{R}^d$, the boundary was not preserved well, and misclassification occurs. By projecting the data back to the null space $\mathbf{R}^{D-d}$, we restore information lost by $\mathbf{f}^d$, and can find a new feature independent of $\mathbf{f}^d$ in this space. To do so we restrict the new feature to be a linear combination of the D features.

The null space can be defined as $\mathbf{N}(\mathbf{f}^d) = \{\mathbf{s}|\mathbf{f}^d(\mathbf{s}) = \mathbf{0}\}$. Readers interested in the details of finding $\mathbf{N}(\mathbf{f}^d)$ are referred to [17]. In short, we use a combination of *singular value decomposition* and *orthogonal projection* to project the data to the null space.

To construct the new feature, we first find a separating hyperplane $\vec{\mathbf{w}}$ among the projected samples in the null space. The new feature was constructed by calculating a directed Euclidean distance of given samples to the hyperplane, $\vec{\mathbf{w}} \cdot \mathbf{x}$. We find a separating hyperplane using linear discriminant analysis [8]. The algorithm is as follows.

Algorithm

Repeat

Draw sufficient data from the discovery set $\mathbf{X}$, project them into lower dimensional space by $\mathbf{f}^d$, then train and test NN on the data.

Find clusters of errors.

Repeat

Select a tight cluster containing both types of errors.

Draw more samples from $\mathbf{X}$ into the cluster.

Project samples in the selected cluster back to the null space.

Find a separating hyperplane in the null space.

Construct a new feature and examine its performance.

Until the feature lowers the error rate sufficiently.

Add the feature to the feature set, and set $d = d + 1$

Until the error rate is satisfactory to the user.

The algorithm is illustrated in Fig. 1. We begin with samples in $\mathbf{R}^D$, colored red and blue to indicate the two classes. These are projected, using $\mathbf{f}^d$, into the current feature space $\mathbf{R}^d$, where clusters of errors are found and indicated by green circles. One cluster is chosen and projected into the null space $\mathbf{N}(\mathbf{f}^d)$, as indicated by the green arrow. Finally a separating hyperplane is chosen in $\mathbf{N}(\mathbf{f}^d)$.

Notice that there is no way of telling whether the errors are from the same region in the original space, or are clusters of errors that overlap because of $\mathbf{f}^d$. However, clusters that span a smaller region with higher density are more likely to be from the same decision boundary lost during

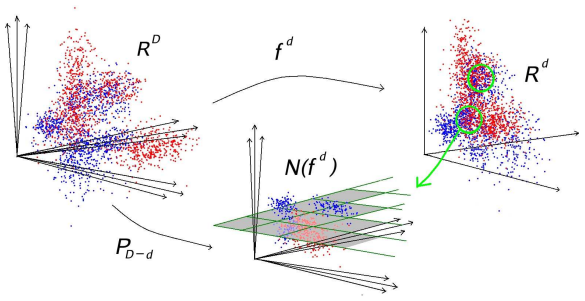


Figure 1. Projections between sample space $\mathbf{R}^D$, feature space $\mathbf{R}^d$, and null space $\mathbf{N}(\mathbf{f}^d)$.

the projection to the current feature space. The clustering step, while allowing the method to adapt to real world data, involves more engineering choices than other parts of the algorithm.

We use NN classifiers because they can work on any distribution without prior knowledge of the parametric models of the distribution. Also, the NN classifier has the helpful property that with an unlimited number of prototypes, the error rate is never worse than twice the Bayes error in the two-category case [6].

3. Document Content Image Extraction (DICE)

We have applied our algorithm to a *document image content extraction* problem: finding regions containing machine-printed text, handwriting, photographs, etc in images of documents [4, 5, 2, 1]. We select a small window around each pixel as our sample space and look for some linear combination of pixel values within this window as our new feature.

Our DICE algorithms cope with a rich diversity of document, image, and content types. Types of document images we process include color, gray-level, and black-and-white; also, any size or resolution (digitizing spatial sampling rate); and in any of a wide range of file formats (TIFF, JPEG, PNG, etc). Our sample page images contain the following types of content: handwriting, machine print, photos, math notation, junk (e.g. margin and gutter noise), and blank. Content types of our samples are across a wide range of languages and image qualities. Fig. 2 shows classification results of two images with the manually chosen feature set. In this example we show that our DICE algorithms are able to extract content in regions of arbitrary shape; however, handwriting poses a particularly hard problem — most of them are classified as machine print.

We convert all image file formats in a three-channel color PNG file in the HSL (Hue, Saturation, and Luminance) color space.

Both training and testing datasets consist of pixels labeled with their ground-truth class: machine print (MP), handwriting (HW), photograph (PH), blank (BL), etc. The task is to classify each pixel into different content class: each image pixel is treated as a sample. Each sample is represented by scalar features. Possible features are extracted from a small region centered on that pixel, with the classification result assigned to the center pixel.

We classify individual *pixels*, not *regions*, in order to avoid arbitrary restrictions on region shapes. Other researchers have attacked this problem of fine-grain classification without restricting region shape. In [13], Nicolas and Dardenne *et al* adapt and apply conditional random fields (CRF) to document image segmentation; they classifies 3x3 regions. Kumar and Gupta *et al* in [11] use globally matched wavelet filters to discriminate text from non-text within color document images; they classify individual pixels.

Ground-truthing was done using on a web based application. The converted document images in PNG format were zoned and labeled using overlapping rectangles. Unzoned pixels are not included in the training set and are ignored when scoring classifier output. From our experience we find that a tight ground-truthing is essential for good classification [12]. By tight ground-truthing we mean the rectangles should be made as small as possible to approximate the shape of corresponding content types. For example, while a row of machine print text can be bound precisely by a single rectangle, one handwriting character may require several to approximate its shape.

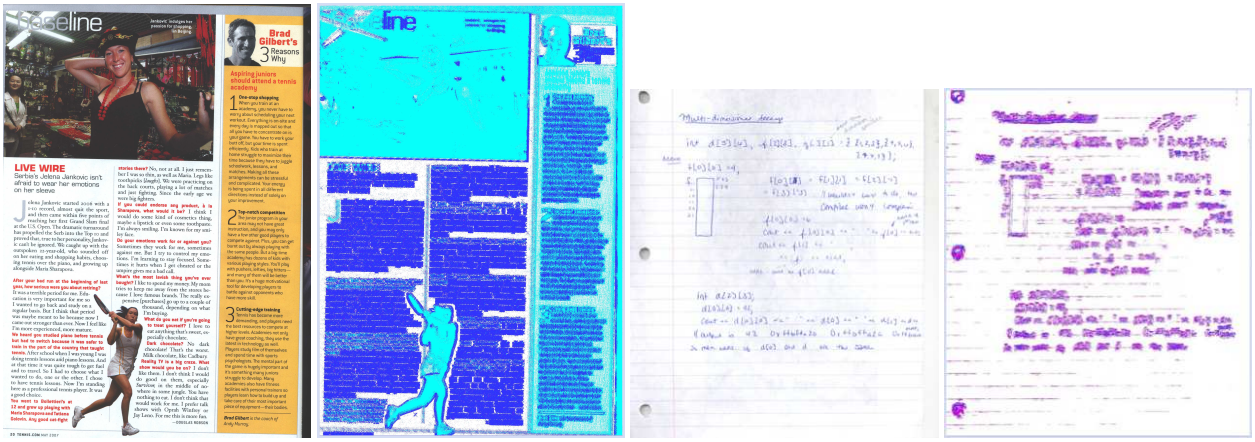
We evaluated performance by *per-pixel accuracy*: the fraction of all pixels in the document image that are correctly classified, that is, those samples whose class label match the class specified by the ground truth labels. Unclassified pixels are counted as incorrect. This is an objective and quantitative measure, but it is somewhat arbitrary due to the variety of ways that content can be zoned. Some content—notably handwriting—often cannot easily be described precisely by overlapping rectangular zones. This problem can be alleviated by more precise ground-truthing.

4. Experiments

We conduct experiments in the DICE framework. Features are extracted from a 25×25 pixels square, so that $D = 625$. We chose MP and HW as our target classes.

The training set consists of 97 images, with a total of 117,028,950 MP samples and 8,946,093 HW samples. The test set consists of 87 images, with a total of 89,846,097 MP samples and 4,354,876 HW samples.

We use the k-means clustering algorithm and assign



(a) machine print and photos (b) classification results (c) handwritten notes (d) classification results

Figure 2. Examples of document images with complex page layouts. The original image (a) is in full color. In the classification results in (b) and (d), machine print (MP) is dark blue, handwriting (HW) red, photographs (PH) light blue-green, and blank (BL) white. This result is obtained with a set of 28 manually chosen features. Note that although photos in the test set are also classified, the result is not taken into consideration in this paper.

eight centers. Note that in our data the number of MP samples is inherently greater than HW data. We pick the error cluster with errors from one type no greater than twenty times of the other. We find that clusters that are too loose or too biased toward one type of error are slightly less likely to yield good features.

We use a simple linear discriminant that assumes Gaussian distribution with a diagonal covariance matrix to find the separating hyperplane. After finding the hyperplane, we normalize the length of vector $\overrightarrow{\mathbf{w}}$ because in the experiments, the length of the vector shrinks rapidly and may cause numerical instability.

We conducted four experiments to compare our algorithm with features chosen by different methods. All experiments were conducted on Lehigh University High Performance Computing Clusters with 8 core Intel Xeon 1.8GHz CPU and 16 GB memory at each node.

Experiment 1 We use a set of 28 features chosen manually and tested over a peroid of two years. Detail description of these features can be found in [3]. The error rate for this set is 13.6% and is shown in Fig. 3 in pink indicated as "28 manual."

Experiment 2 Our algorithm starts with four handcrafted linear features with an error rate of 16.2%. Fig. 3 shows the error rate of composite feature set in deep blue, indicated as "4 manual." The error rate decreases monotonically until the 6th discovered feature, then fluctuates slightly around 14.5% for the next three features, and finally drops to 13.8% at the 10th discovered feature.

Experiment 3 We ran an experiment with a set of 14 features chosen by PCA. The error rate is 15.4% and is shown in Fig. 3 in yellow, indicated as "PCA." The error rate of the PCA features is outperformed at the 3rd discovered feature in Experiment 2.

Experiment 4 We ran an experiment with only one given feature chosen by PCA. The reason to include one PCA feature is because our algorithm needs to be bootstrapped with at least one given feature. The error rate is shown in Fig. 3 in light blue indicated as "1PCA." The result shows that there is no obvious advantage by giving four manually chosen feature.

To extract a new feature from current classification results, we first extract error sample points: the computing time for this step is approximately linear in the number of features, with an average of 15 CPU seconds per feature. The extracted errors are then decimated to approximately 100,000 $\sim$ 150,000 points, which takes an average of 10 CPU seconds. Time for extracting and clustering those points is superlinear in the number of features and ranges from 110 CPU seconds for two features to 47 CPU minutes for ten. Calculating the null space takes an average of 8 CPU seconds and is independent of the number of features; populating the error cluster is sub-linear in the number of features and takes an average of 25 CPU seconds. Finding the separating hyperplane is super-linear and takes an average of 21 CPU seconds for two features, and 2.5 CPU minutes for ten.

Our features usually take more time to extract than man-

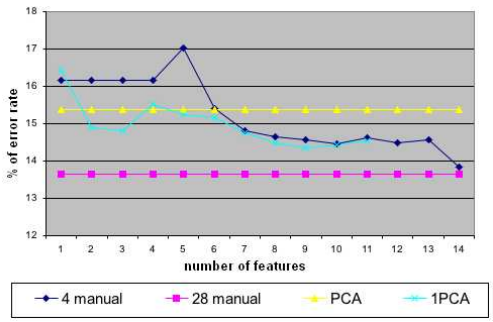


Figure 3. Error rate (%) as a function of number of features. The deep blue line is the error rate of the four manual features plus discovered features. The light blue is the error rate of one feature chosen by PCA plus discovered features. The pink line is the error rate of all 28 manual features. The yellow line is the error rate of 14 features chosen by PCA.

ual features. Training time increases linearly in the number of features with an average of 4 CPU minutes for each feature on each image. The classification phase takes up most time and increases superlinearly in the number of features. The classification time for each image is less than 1.5 CPU hours for 5 features and less than 6.5 hours for 14 features on each image.

5 Discussion and Future Work

In this paper we present a document image content extraction framework using automatically discovered features. Experiments suggest that the performance of a composite set of features containing both manually chosen features and automatically discovered feature is comparable to the performance of a manually chosen feature set. This composite feature set is only half of the size of the manually chosen feature set with the new features discovered fully automatically. Also, a set of three discovered features along with the original four features outperformed a set chosen by PCA with twice as many features.

Our plan for the future includes applying our algorithm on publicly available datasets and comparing our results with other approaches, conducting larger experiments, and establishing general conditions under which our algorithm is guaranteed to improve the error rate.

References

[1] H. Baird and M. Moll. Document content inventory and retrieval. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 93–97, Washington, DC, USA, 2007. IEEE Computer Society.

[2] H. S. Baird and M. R. Casey. Towards versatile document analysis systems. In *Document Analysis Systems*, pages 280–290, 2006.

[3] H. S. Baird, M. A. Moll, C. An, and M. R. Casey. Document image content inventories. volume 6500, pages 65000X+. SPIE, 2007.

[4] H. S. Baird, M. A. Moll, J. Nonnemaker, M. R. Casey, and D. L. Delorenzo. Versatile document image content extraction. In *Proc., SPIE/IS&T Document Recognition & Retrieval XII Conf.*, San Jose, CA, January 2006.

[5] M. R. Casey. Fast approximate nearest neighbors. Master's thesis, Lehigh University, Bethlehem, PA, May 2006. M.S. Thesis.

[6] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. on Information Theory*, 13:21–27, 1967.

[7] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

[8] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, 2nd Edition*. Wiley, New York, 2001.

[9] J. D. Elashoff, R. M. Elashoff, and G. Goldman. On the choice of variables in classification problems with dichotomous variables. *Biometrika*, 54:668–670, 1967.

[10] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[11] S. Kumar, R. Gupta, N. Khanna, S. Chaudhury, and S. Joshi. Text extraction and document image segmentation using matched wavelets and mrf model.

[12] M. A. Moll, H. S. Baird, and C. An. Truthing for pixel-accurate segmentation. *Document Analysis Systems, IAPR International Workshop on*, 0:379–385, 2008.

[13] S. Nicolas, J. Dardenne, T. Paquet, and L. Heutte. Document image segmentation using a 2d conditional random field model. *Document Analysis and Recognition, International Conference on*, 1:407–411, 2007.

[14] K. Nigam, J. Lafferty, and A. Mccallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

[15] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[16] P. Silapachote, D. R. Karuppiah, and A. R. Hanson. Feature selection using adaboost for face expression recognition. 2005.

[17] S.-Y. Wang and H. S. Baird. Feature selection focused within error clusters. In *ICPR*, pages 1–4, Tampa, FL, Dec 8-11 2008.

[18] H. Wold. The fix-point approach to interdependent systems. 1981).

[19] H. Wold. Partial least squares. *Encyclopedia of statistical sciences*, 6:581–591, 1985.