https://doi.org/10.2352/ISSN.2470-1173.2018.2.VIPC-253

Text/Figure Separation in Document Images Using Docstrum Descriptor and Two-Level Clustering

Valery Anisimovskiy**, **Ilya Kurilin**, **Andrey Shcherbinin**, **Petr Pohl
Samsung R&D Institute Russia, Moscow, Russia*

Abstract

We propose a novel algorithm for text/figure separation tailored for binary document images containing line drawings, block diagrams, charts, schemes and other kinds of business graphics. Most of the approaches for this task rely either on clever design of visual descriptor allowing to easily distinguish text and graphics regions or on the supervised learning using dataset of labeled text/figure regions. Such approaches often provide moderate separation accuracy when applied to document images which contain very diverse set of figure classes and lack sufficiently representative labeled training dataset. In contrast, our method is well-suited for vast variety of figure classes and capable of operating either in semi-supervised mode or unsupervised mode. We achieve this by leveraging unsupervised learning algorithms applied to Docstrum descriptors extracted from regions of interest and subsequent semi-supervised label propagation or unsupervised label inference. Another advantage of our method is its suitability for large scale data processing which is achieved through efficient kernel-approximating feature mapping applied to Docstrum descriptors and two-level clustering where fast mini-batch K-means algorithm is first applied to large scale data and only small number of resulting cluster centroids is subsequently processed by one of the more sophisticated clustering algorithms.

Introduction

Text/figure separation problem is relevant for various document image analysis and processing tasks, e.g. document indexing and retrieval, document object detection and recognition, optical character recognition (OCR), document compression and many more. Classification of regions of interest (ROI) into text or figure class is of high importance in such applications since it can significantly reduce the amount of data to be processed by the subsequent stages by removing irrelevant regions (e.g. non-text regions for OCR or text regions for image indexing/retrieval).

Most of the approaches proposed to solve this problem rely on clever design of hand-crafted ROI descriptor producing easily distinguishable feature vectors for text and non-text regions. Such approaches often exhibit high misclassification rate on figure classes which do not completely fit the considerations underlying the design of their features (or heuristics used to classify the features).

Another family of approaches relies on supervised machine learning algorithms trained on manually labeled datasets. Classification performance of such approaches heavily depends on representativeness of the training dataset which is very difficult to achieve if very diverse set of highly variable figure classes is to be handled.

Our method combines the strong points of both families of approaches while trying to mitigate their shortcomings. We achieve that by applying unsupervised machine learning algorithms to shape-based features extracted from document ROIs and subsequent semi-supervised label propagation or unsupervised label inference

Related Works

One of the most popular families of text/non-text separation methods is based on extracting simple features from document ROI and then classifying those features using a few clever handcrafted heuristrics aimed at separating text from non-text regions. Typical features used in such approaches are based on connected components, run-length statistics, cross-correlation between scanlines, projection profiles or black pixel distribution (see review in [1] and more recent methods in [2], [3], [4] and [5]). Such approaches are often fast and efficient for documents where figures are photographs, paintings or other kinds of imagery significantly different from text in structure, but for line drawings, especially text-rich block diagrams, electric schemes and similar material such approaches often exhibit much worse accuracy. One of their major weaknesses is that their heuristics are typically deduced from observations relevant for figures belonging to a particular class (or classes) and may not generalize well for other classes, which is especially problematic for documents containing very diverse set of figure classes (as is the case for figures in patent images) where some of the classes do not completely fit those heuristics. For example, the methods relying on projection or run-length histograms may incorrectly classify block diagrams containing much of text as text regions.

This drawback is tackled by another family of approaches relying on supervised machine learning algorithms applied either to the features of the same kind as above or to the image pixel data. Since the task of text/figure separation may be formulated as a binary classification problem, approaches of this family typically utilize training on manually labeled dataset to learn the distinction between text and non-text regions, the examples including shape-based features classification using k-nearest neighbor (KNN) classifier [6] or multilayer perceptron (MLP) [7], while in [8] support vector machine (SVM) classifier using gradient-based T-HOG descriptor is utilized.

Such approaches might still suffer from insufficient discriminative capability of hand-crafted features which cannot be compensated by the classifier. So, more efficient approaches dealing with image pixel data or low-level features were developed. One prominent approach of such a kind is based on sparse coding techniques. For example, in [9] morphological component analysis (MCA) using two pre-constructed discriminative overcomplete dictionaries (curvelet transform for graphics and undec-

imated wavelet transform for text) is proposed, while in [10] image patches classification based on sparse representations of the patches in two sequences of learned dictionaries (for text and graphical parts) is shown to provide better text/graphics separation and in [11] this method was further improved and elaborated.

However, supervised learning-based algorithms are often prohibitively slow on large scale datasets (as is again the case for patent images containing millions of regions) and they work best when the training dataset has all relevant figure classes sufficiently represented, so that the learning algorithm could learn how to distinguish each class from text. Such training dataset may not be readily available for many kinds of documents, e.g. when the figure classes are so numerous and have very high within-class variation that creation of such dataset by manual labeling would be prohibitively laborious. To remove the need for manually labeled training dataset, the unsupervised learning-based methods were developed: typically, K-means algorithm is used for clustering statistical features computed using, for example, high-frequency wavelet coefficients [12] or edge maps [13]. As we show further in this paper, K-means algorithm, being unable to handle non-convex, nested and elongated clusters, underperforms in the text/figure separation problem, especially when using Euclidean distance.

Finally, one important note regarding previously-developed methods is that most of them are particularly concentrating on the text extraction task, while we focus our work on the extraction of figures. Furthermore, most of the methods outlined above typically work on the natural scenes (photographs or video sequences). This kind of data is vastly different in its structure and properties of contained graphics from patent-like documents, which are the focus of our work. Note also that many previous works use well-known UW-III and ICDAR 2009 datasets for designing their hand-crafted features or training their classifiers, which might negatively affect the generalization capability of both features and classifiers in the context of our task, since those datasets contain only small subset of figure classes present in patent-like documents.

Description of Method

Algorithm Overview and Requirements

Since patent page images exhibit the aforementioned properties (large scale dataset with very diverse set of figure classes having very high within-class variation), we use page images extracted from a randomly chosen subset of patents from USPTO patent database to demonstrate our method. However, the method itself is suitable for other document classes containing business figures similar to the ones used in patents.

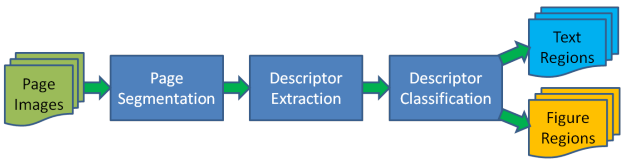

Figure 1. *Flow chart of the proposed method.*

Flow chart of our method is shown in Fig.1. Patent page images are first segmented into well-separated ROIs, then for each ROI a descriptor is extracted and finally the resulting descriptors are classified into one of the two classes: text or figure. We further discuss the basic requirements for each of the stages which motivated our choice of algorithms used in those stages.

Since text/figure separation is typically used as a preprocessing stage preparing data for subsequent much more complex stages, one would ideally expect the separation algorithm to be relatively fast and lightweight (as compared to subsequent stages, which could be, e.g. OCR or image indexing and retrieval engine). This requirement implies usage of a lightweight global descriptor extraction algorithm (rather than more costly aggregation of local descriptors, such as VLAD or Fisher Vector for SIFT descriptors) producing low-dimensional feature vectors. In case of black-and-white document images (such as patents) its also quite logical to use a descriptor extraction algorithm specifically tailored for document images containing text and line drawing figure regions.

Another requirement, stemming from large number (millions) of available patents, is that the classification algorithm should be suitable for processing large scale datasets. Due to the lack of sufficiently representative labeled dataset for patent figures the classification algorithm should provide either semi-supervised (where only small portion of feature vectors are labeled and used to propagate labels to unlabeled data) or unsupervised operation (where dataset labeling is not required at all).

Finally, for the subsequent processing of text or figure regions to be complete and efficient, we require the text/figure separation algorithm to provide high recall and good precision. Since our specific focus in this work is on separation of figures for their subsequent indexing and retrieval, we set the minimum levels for figure ROI recall at 90% and for precision at 75%, which means that we miss at most 10% of all figures and tolerate at most 25% of text contamination in selected set of ROIs. Keeping both recall and precision at least that high is quite crucial for figure retrieval task, since low recall would result in too many figures being not indexed while low precision would significantly increase excess computations done for text ROIs.

In the following subsections we describe each stage in detail.

Page Segmentation

Since patent pages are binary document images which typically use Manhattan layout for text and figures, we use simplified version of very simple and fast Run-Length Smoothing with OR (RLSO) segmentation algorithm [14]. Our version of RLSO algorithm, being a variant of Run-Length Smoothing Algorithm (RLSA), operates by first filling background pixel runs having lengths below pre-defined threshold in both horizontal and vertical directions and then selecting bounding boxes of the resulting connected components as ROIs. RLSO is different from RLSA in that it uses logical OR operation between horizontally and vertically smoothed images instead of logical AND. Our modification of RLSO further simplifies it by replacing sophisticated smoothing threshold estimation with calculation of 90th and 80th percentiles of background pixel run lengths for horizontal and vertical smoothing respectively.

The example of RLSO application to patent figures is illustrated in Fig.2, while Fig.3 shows the ROIs extracted from a mixed content patent page. We discard too small ROIs having area (in pixels) less than 0.1% of the entire image area.

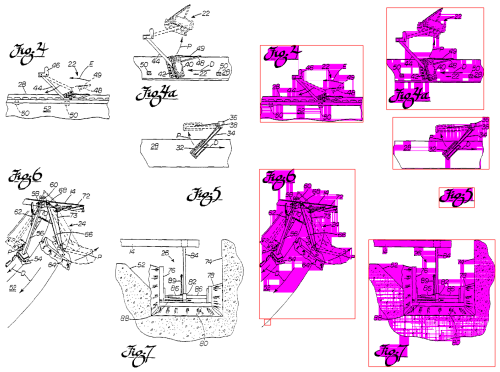

Figure 2. *__Left__: Original patent page with five figures. __Right__: Result of applying RLSO to the page: filled background pixels are magenta-colored, connected components are highlighted by red bounding boxes.*

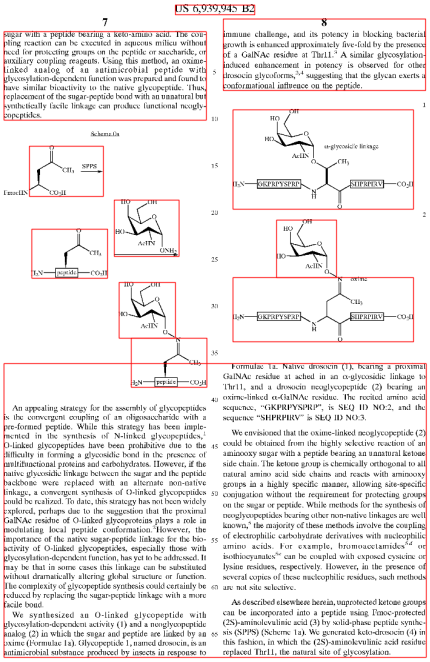

Figure 3. *ROIs extracted by RLSO from patent page image containing mixed text and figure content.*

Descriptor Extraction

For the image ROIs extracted by the simplified RLSO algorithm we tried several global image descriptors tailored for binary document images: Hu moments [15], Haralick features [16], Shape Context Descriptor (SCD) [17], Run-Length Histogram (RLH) [18], Local Binary Pattern (LBP) [19], Adaptive Hierarchical Density Histogram (AHDH) [20], and Docstrum [21]. The most distinct and consistent difference between feature vectors for text and figure regions of our dataset is exhibited using Docstrum descriptor. This result is quite expectable, since Hu moments, Haralick features and LBP are designed primarily for texture classification, SCD is aimed at shape matching, RLH and AHDH are well suited for document retrieval, whereas Docstrum is designed for page layout analysis in text-only documents. This design goal, as we show further, makes this algorithm produce much more "chaotic" feature vectors for non-text regions making them easily distinguishable from more "regular" feature vectors produced for text regions.

Our implementation of Docstrum descriptor operates for each ROI using the following steps:

1. Resize the ROI to the size of 500x500 pixels while preserving aspect ratio, that is, first the ROI is resized so that the longest dimension (width or height) becomes 500 pixels, then along the shortest dimension the ROI is padded to 500 pixels using background pixel value.
2. Connected components are extracted from the resized ROI and their centroids are computed. The components having bounding box width or height below 1% of the resized ROI's corresponding dimension are filtered out.
3. For all the pairs composed of centroid and each of its 5 nearest neighbor centroids the 2D histogram of normalized distances and angles is built. Distances are normalized by dividing by the average distance of all the centroid pairs. If the number of centroids is too small (below 4), uniform histogram is used.
4. The resulting 2D histogram is reshaped to 1D vector and normalized to have L_1-norm of unity.

Note that ROI resizing in step 1 and distance normalization in step 3 differentiates our version of Docstrum from the original algorithm described in [21]. We use ROI resizing to reduce computational complexity of the algorithm (since most of the ROIs have sizes much larger than 500x500 pixels) and the distance normalization to make the descriptor invariant to scaling. Aspect ratio preserving in step 1 is used to prevent distance/angle distribution skewing due to non-isotropic resizing. The destination image size for the resizing step was chosen as a trade-off between descriptor computation complexity and figure details preserving. To build 2D distance/angle histogram we use 64 angle bins and 20 distance bins which results in 1280-dimensional feature vector. These settings are again chosen as a trade-off between the descriptor dimensionality and its discriminative capability for text/figure separation.

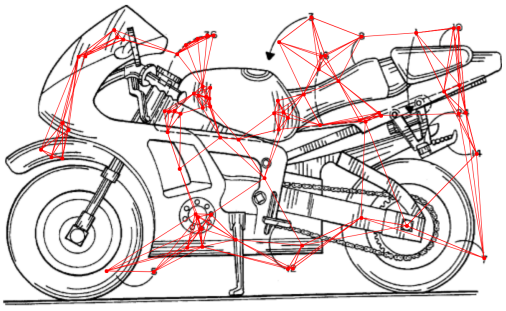

Figure 4. *Nearest neighbor pairings of connected component centroids for example figure ROI.*

The examples of nearest neighbor pairings of connected component centroids for figure and text ROIs are shown in Fig. 4 and Fig. 5, respectively.

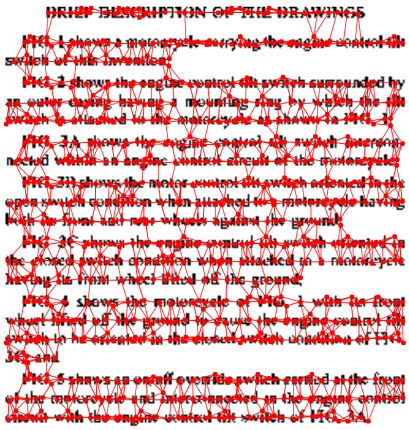

Figure 5. *Nearest neighbor pairings of connected component centroids for example text ROI.*

The Docstrum descriptors (depicted as 1D histograms) computed using the above settings for the same figure and text ROIs are shown in Fig. 6. As one can see, the histogram for text ROI exhibits distinct regularly-spaced peaks, unlike the histogram for figure ROI which looks much more chaotic.

Note that it's possible to reduce descriptor dimensionality (and, thus, computational complexity of descriptor computation and processing) by further decreasing the size of ROI and histogram binning. For example, with 16 angle bins and 20 distance bins one can get 320-dimensional feature vectors. Although the discriminative capability of the descriptor for such settings is worse, it is still suitable enough for the task of text/figure separation.

Descriptor Classification

Flow chart of the descriptor classification is shown in Fig. 7. First, descriptor feature vectors are optionally transformed by kernel-approximating feature mapping, the resulting transformed feature vectors are then clustered using mini-batch K-means algorithm and then centroids of the resulting clusters are in turn clustered by one of the more advanced clustering algorithms. The resulting clusters of centroids are effectively aggregating clusters corresponding to those centroids (output by the first level clustering) into "superclusters". Finally, the resulting superclusters are classified either by unsupervised label inference or by semi-supervised label propagation using the subset of labeled feature vectors.

To evaluate the quality of classification we also included classification performance evaluation path consisting of manual labeling the first level clusters and then computing precision, recall and F_1 scores after supercluster classification.

Further we elaborate on each stage of the outlined flow chart.

Kernel-Approximating Feature Mapping

Since the Docstrum descriptor is a histogram, Euclidean distances (or inner products) used by subsequent algorithms are not much adequate as a measure of feature vectors proximity. The popular kernels for histogram-based descriptors widely used in computer vision and machine learning applications are Hellinger's (Bhattacharyya's), χ^2, intersection and Jensen-Shannon kernels [22]. These kernels, along with their corresponding distances are defined in Table 1, where we also include Jaccard similarity score (adapted for histograms), which, to the best of our knowledge, is not widely used in computer vision applications, but, as we show further, deserves attention in the context of our task as well.

Histogram-oriented kernels and distances

Kernel	$\mathbf{K}(\mathbf{x},\mathbf{y})$	$\mathbf{D^2}(\mathbf{x},\mathbf{y}) = \mathbf{K}(\mathbf{x},\mathbf{x}) + \mathbf{K}(\mathbf{y},\mathbf{y}) - \mathbf{2K}(\mathbf{x},\mathbf{y})$
Hellinger	$\sum_i \sqrt{x_i}\sqrt{y_i}$	$\sum_i (\sqrt{x_i} - \sqrt{y_i})^2$
Intersection	$\sum_i \min(x_i, y_i)$	$\sum_i \lvert x_i - y_i \rvert$
χ^2	$\sum_i \frac{2x_i y_i}{x_i + y_i}$	$\sum_i \frac{(x_i - y_i)^2}{x_i + y_i}$
Jensen-Shannon	$\sum_i \frac{x_i}{2} \log_2 \frac{x_i + y_i}{x_i} + \frac{y_i}{2} \log_2 \frac{x_i + y_i}{y_i}$	$KL\left(x \middle\vert \frac{x+y}{2}\right) + KL\left(y \middle\vert \frac{x+y}{2}\right)$ *(KL is Kullback-Leibler divergence)*
Jaccard	$\frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$	$2 - 2\frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$

However, since the subsequent mini-batch K-means clustering algorithm is only able to use Euclidean distances between the feature vectors, we have to transform Docstrum feature vectors using suitable kernel-approximating feature mapping. Such feature mapping is defined as a mapping $\Psi : \Re^D \to \Re^N$, such that $\forall x, y \in \Re^D : K(x,y) \approx \langle \Psi(x), \Psi(y) \rangle$, that is, linear (Euclidean) inner product in the transformed feature space $\Re^N$ approximates non-linear kernel $K(x,y)$ in the original feature space $\Re^D$. Since a positive-definite kernel $K(x,y)$ corresponds to a distance $D(x,y)$ given by $D^2(x,y) = K(x,x) + K(y,y) - 2K(x,y)$ [23], it's easy to see that $\forall x, y \in \Re^D : D^2(x,y) \approx ||\Psi(x) - \Psi(y)||^2$, where $||\cdot||$ is Euclidean norm in $\Re^N$, so the distance corresponding to the kernel is also approximated by the same feature mapping.

Since Hellinger's, χ^2, intersection and Jensen-Shannon kernels are additive and γ-homogeneous, their approximating feature mappings can be derived in closed form [22]. For Hellinger's kernel we use exact dimensionality-preserving mapping, which takes a square root of each component, while for χ^2, intersection and Jensen-Shannon kernels we use approximating mappings from [22] with 5 components, which results in mappings of a form $\Psi : \Re^{1280} \to \Re^{6400}$ (5 times 1280). For feature vectors in our dataset such mappings provide distance approximation accuracy of 6%.

In contrast to other histogram-oriented kernels, Jaccard kernel is neither additive nor γ-homogeneous (or stationary), so it's much more difficult to derive approximating feature mapping for it in closed form. Therefore, we used siamese neural network to train approximating feature mapping for this kernel. The architecture of the network is shown in Fig. 8.

The input to the network is composed of two randomly-selected Docstrum feature vectors x and y (each being 1280-dimensional). Each of the two vectors is fed into its own fully-connected (FC) layer having Parametric Rectified Linear Unit (PReLU) activation [24] and 5000 outputs. Both FC+PReLU layers share all the weights during training, so each of them actually performs the same feature mapping $\Psi : \Re^{1280} \to \Re^{5000}$ which is being trained by the network. The outputs of the FC+PReLU layers, $\Psi(x)$ and $\Psi(y)$, are fed into L_2 distance computation layer, which computes squared Euclidean distance $||\Psi(x) - \Psi(y)||^2$ and

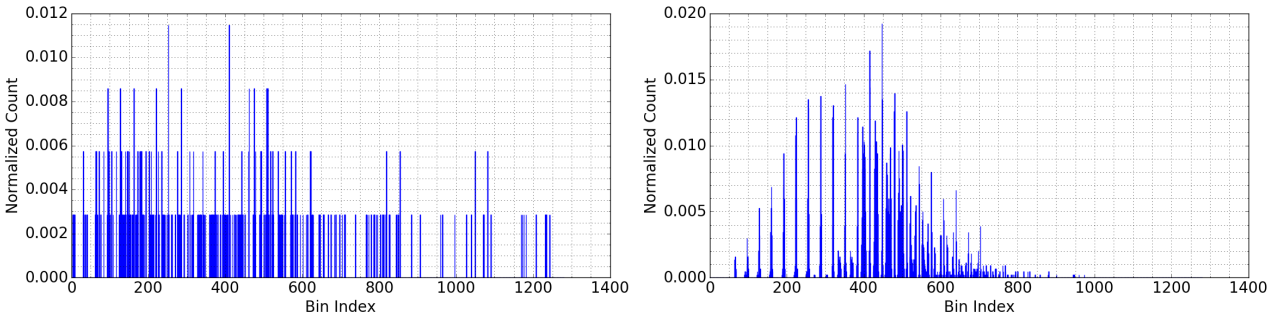

Figure 6. *Docstrum descriptors for figure (left) and text (right) ROIs. Docstrum is computed for ROIs having a size of 500x500 pixels using 64 angle bins and 20 distance bins.*

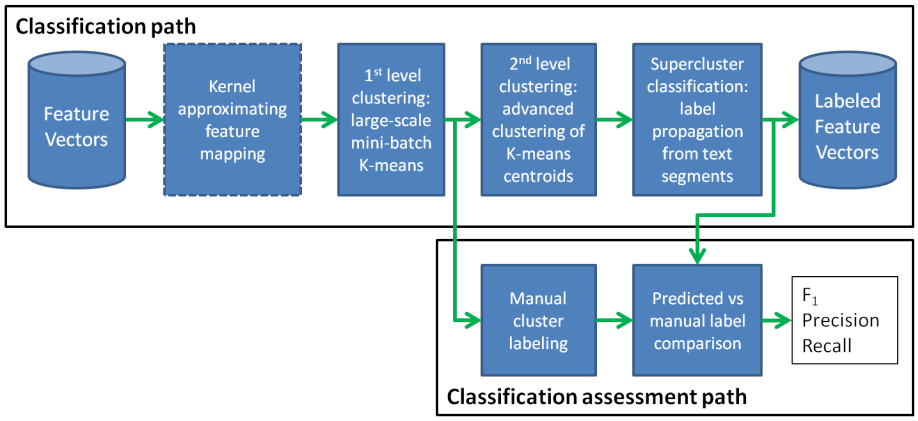

Figure 7. *Flow chart of the descriptor classification.*

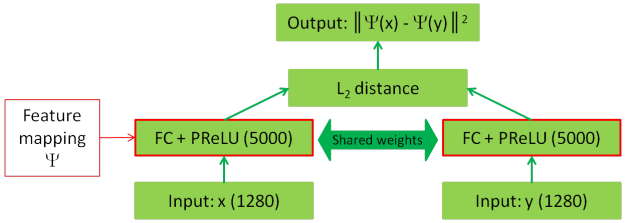

Figure 8. *Siamese neural network used to train feature mapping for approximation of Jaccard kernel. The red-delineated layers have shared weights and constitute the feature mapping Ψ being trained by the network. Where appropriate, layer output dimensionality is given in parentheses.*

does not contain learnable parameters. The resulting squared distance is the output of the network. The squared Jaccard kernel distance $D^2(x,y)$ is used as the target output during training, while Mean Absolute Percentage Error (MAPE) is used for training loss function. The weights of FC layer are initialized using Glorot uniform initialization [25], PReLU parameters are initialized to zero and Nesterov ADAM [26] is used as optimizer algorithm. After training for 300 epochs using batch size of 10000 feature vector pairs, the network achieves test accuracy of 7% (MAPE) in Jaccard distance approximation which is comparable to that for the other kernels.

Note that we still use Euclidean distance (which we refer to as "linear kernel") for the subsequent feature vector processing stages alongside other kernels mentioned above, in which case the kernel approximation stage is not used.

First Level Clustering

We chose mini-batch K-means algorithm [27] for the first level clustering stage due its fast performance, suitability for large scale processing and generally good quality of the resulting clustering.

However, this algorithm, like the standard K-means algorithm, is not capable to correctly handle non-convex or elongated clusters, which may well be the case for our task. We verify this hypothesis by running K-means clustering configured to output small number of clusters (from 2 to 10) on transformed Docstrum descriptors and visually inspecting the ROIs corresponding to the feature vectors of the resulting clusters. This test shows that the resulting clusters contain mixture of both text and figure ROIs without clear domination of one of the classes. However, when we increase a number of clusters to 20 and higher, the output clusters become dominated by either text or figure ROIs, which supports our hypothesis, since higher number of requested clusters allows K-means algorithm to subdivide non-convex or elongated text/figure clusters into smaller convex and isotropic subclusters without much intermixing. By varying requested clusters number from 2 to 1000 and visually inspecting the results, we found 100 clusters to be optimal trade-off between cluster purity and computational complexity of K-means clustering.

We run mini-batch K-means algorithm configured to output 100 clusters with mini-batch size of 1000 feature vectors for 100

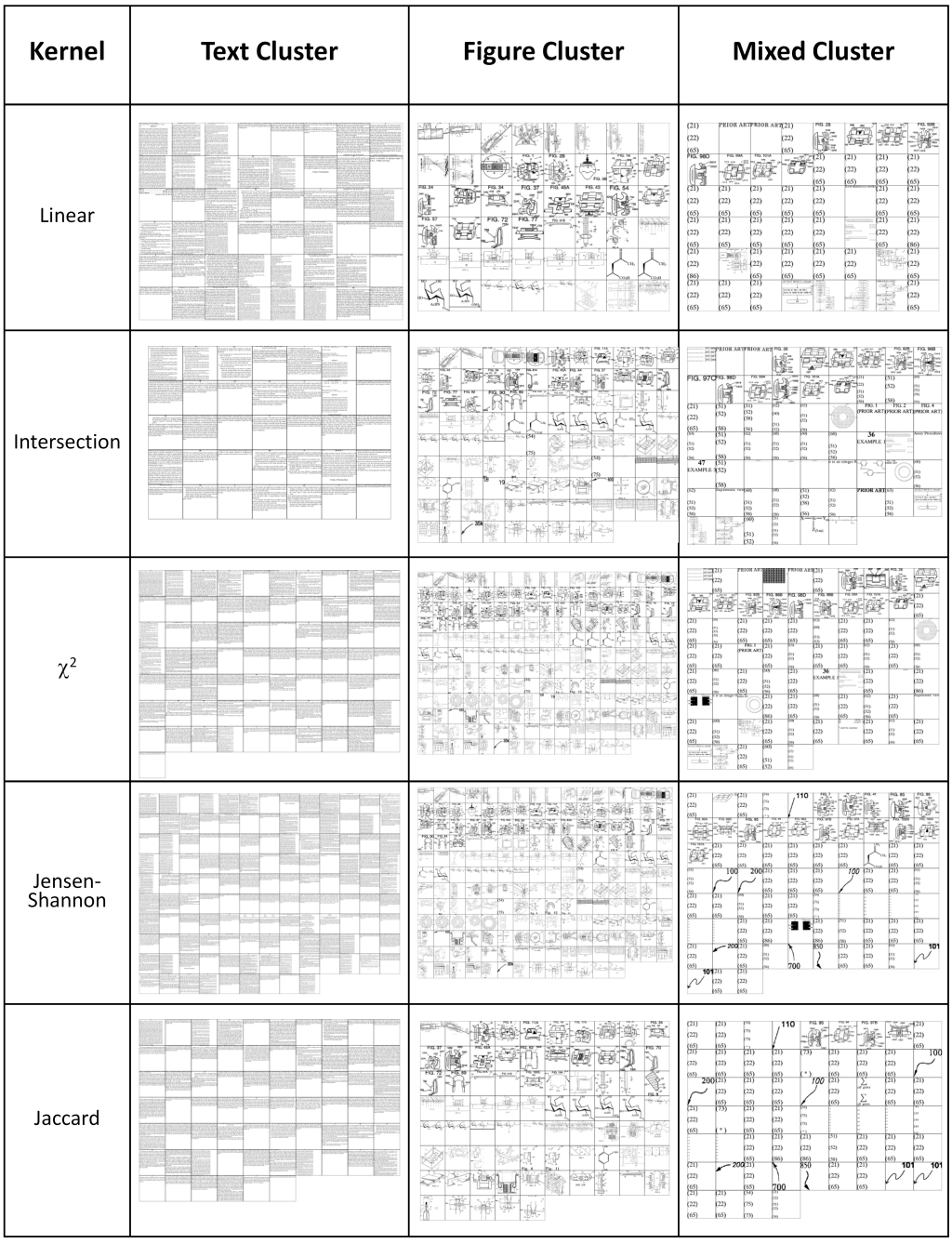

Figure 9. *The examples of clusters output by the first level clustering using different kernels. Columns represent three types of clusters: text, figure and mixed.*

epochs, until mini-batch inertia, averaged over several latest mini-batches, almost stops improving.

The examples of resulting clusters (composed of tiled ROIs corresponding to cluster's feature vectors) are shown in Fig. 9. Three types of typical clusters are represented: text, figure and mixed. One can see that text and figure clusters are almost 100% pure (contain ROIs of only one class text or figure), while typical mixed clusters are dominated by one class and contain at most 30% of impurity from the other class. We performed manual analysis of those mixed clusters which shows that they contain less than 10% of all the feature vectors of our dataset, so the total impurity percentage is less than 3%, which can be safely neglected in the context of our task. So, for further processing and analysis we assume that the first level clustering outputs sufficiently pure text and figure clusters.

However, one can notice that the smallest impurity is achieved when using Jaccard kernel, significantly higher impurity is achieved when using χ^2 and Jensen-Shannon kernels, while

intersection and linear kernels are the worst in terms of cluster impurity. As to the Hellinger's kernel, since it provides clustering results even worse than the linear kernel (while offering no performance benefits over it) we do not use this kernel for further stages (except for the exact kernel variant of our method).

Second Level Clustering

Since K-means clustering outputs convex clusters which tend to be isotropic and such cluster geometry is most likely not the case for Docstrum feature vector clusters corresponding to text and figure ROIs, we have to aggregate clusters resulting from the first level clustering into larger clusters (which we refer to as "superclusters" hereinafter) by a clustering algorithm capable of handling non-convex non-isotropic clusters. Because most of such algorithms scale poorly to large number of samples, we chose to apply them to the centroids of the clusters output by the first level clustering rather than to much more numerous actual feature vectors. Since the number of such centroids is 100 (or less), we are no longer constrained by the requirement of suitability for large scale processing.

We tried several algorithms for the second level clustering: K-means with k-means++ initialization [28], affinity propagation [29], mean shift [30], spectral clustering [31], agglomerative clustering using several linkage methods [32], DBSCAN [33], hierarchical DBSCAN (HDBSCAN) [34], BIRCH [35], and one-class SVM [36] using linear, RBF and sigmoid kernels (which are in fact combined with one of the histogram-oriented kernels given in Table 1 due to kernel-approximating feature mapping stage). Note that the last one is actually a novelty detection algorithm which requires clean dataset for training, but we found that it nevertheless performs quite well for our task. We also tried another algorithm for novelty and outlier detection: fitting an elliptic envelope to the data using minimum covariance determinant estimator [37], but it fails due to the fact the data has neither unimodal nor symmetric distribution.

Note that, since for χ^2, intersection, Jensen-Shannon and Jaccard kernels the dimensionality of the transformed feature space is several times higher than that of the original Docstrum descriptor, the first level clustering for those kernels, being the most time-consuming stage of our method, takes 2-3 times longer than for linear kernel (i.e. for original Docstrum feature vectors). So, one apparent way to improve the computational performance of our method is to omit kernel-approximating feature mapping stage, but use one of the histogram-oriented kernels during the second level clustering. This may be achieved by using clustering algorithm capable of taking as input kernel matrix (or distance matrix for distance-based algorithms) instead of feature vectors, since this matrix for the centroids of the clusters output by the first level clustering is much smaller (only 100x100) than for the large scale dataset input to the first level clustering (which may contain millions of vectors). Such algorithms include affinity propagation, agglomerative clustering with single, complete, average and weighted linkages, DBSCAN, HDBSCAN, one-class SVM and spectral clustering. Since in this case we use exact kernels/distances applied to Docstrum feature vector pairs rather than their approximations, we refer to this modification of our method as "exact kernel variant", in contrast to "approximate kernel variant" described earlier. We use both variants in our experiments and compare their performance.

For each of the second level clustering algorithms, we vary its most important parameters in a wide range to find the combinations of parameter values delivering the best performance in terms of precision, recall and F_1 scores. Note that for agglomerative clustering we use both unstructured and structured variants, the latter using connectivity matrix computed from the k-neighbors graph.

Supercluster Classification

The second level clustering stage outputs typically small number of superclusters. To classify those superclusters into "text" or "figure" class we have to leverage some additional source of information about those classes. Depending on the output of the second level clustering stage we employ one of the two modes of supercluster label inference: 1) unsupervised labeling is used when the second level clustering stage outputs 2 superclusters, 2) label propagation is used otherwise (when there're more than 2 superclusters).

In the unsupervised mode we utilize general statistical information about the classes: in case of patent ROIs we use the observation that text ROIs strongly dominate, so we classify the larger of the two superclusters as a text one, while the smaller one apparently corresponds to figure ROIs.

In the label propagation mode we use small labeled subset of our dataset, obtained from the observation that there's a number of patents without figures, which can be readily identified by the absence of words "fig", "drawing", "##str" in their full text versions. All ROIs (and their corresponding Docstrum feature vectors) from such patents are labeled as text ones. Then for each supercluster we compute its "text contamination", i.e. the fraction of its ROIs which belong to this text-labeled subset. Supercluster classification is then performed as labeling the superclusters having text contamination above a certain threshold as corresponding to text ROIs and the rest of the superclusters as figure ones (thereby propagating text label from the small labeled subset to the entire supercluster having high text contamination). The value of the threshold is computed as a percentile of text contaminations of all the superclusters, while the percentile value itself is varied from 0 to 100% alongside the second level clustering algorithm parameters. Since the text-only patents contain text ROIs which are quite representative for the entire set of patent text ROIs, i.e. contain all classes of text regions specific to patents (heading text, table text, regular text boxes, etc), this text label propagation works quite well for our task.

Note that the unsupervised mode does not use any labeled data at all, only general a-priori information about distribution of text/figure ROIs in the dataset. The label propagation mode only uses small subset (less than 1%) of the dataset labeled as data of one class (text ROIs in our case) using additional side information. Note that in this mode supervised approach is not applicable since we've got labeled data for only one class of the two.

Classification Performance Evaluation

In order to quantitatively evaluate the quality of text/figure separation provided by different variants and parameter combinations of our method, we need ground truth labels for our dataset. The classification of entire patent pages as being text or figure pages available on USPTO or Google Patents does not provide precise labeling of ROIs since a patent page may contain ROIs of

both classes: e.g. figure pages contain textual elements (patent numbers, figure titles, etc), while text pages of patents related to chemistry frequently contain figures of chemical compositions mixed with text regions. Another motivation not to use external ground truth labels is that there may be no external source of such labels for other classes of documents requiring text/figure separation.

So, we use ground truth labels produced manually by visual inspection of random subsets of ROIs for each of the clusters produced by the first level clustering stage (it's quite feasible due to relatively small number of clusters output by this stage). Since, as we mentioned earlier, those clusters are dominated by ROIs of single class (text or figure), we label all ROIs of a given cluster as belonging to the dominating class of this cluster. As we estimated earlier, this approach would suffer from mislabeling to the acceptably small extent (less than a few percent of mislabeled ROIs).

Having manually labeled all of the ROIs in our dataset, we evaluate the quality of text/figure separation using widely used classification performance metrics such as recall, precision and F_1 score computed by comparing labels predicted by our method and the ones provided by the manual labeling (we consider figure ROIs as positive samples and text ROIs as negative ones). It should be noted, however, that our results for those metrics are only approximate due to the mislabeling mentioned above.

Note that the manual labeling is only used for the classification performance evaluation and for the search of optimal parameters of our method, while the method itself is fully automatic and does not rely on any manual work.

Experimental Results

Experimental Conditions

We implemented our method in Python using NumPy/SciPy packages. Scikit-learn [38] and fastcluster [39] were used for kernel approximation and clustering and Keras [40] was used for building and training siamese neural network, image and descriptor data were stored in HDF5 format using PyTables package [41]. Scikit-image [42] was used for image processing operations.

The test system had 8-core AMD FX-8350 CPU and NVIDIA GeForce GTX 780 GPU (used to train the neural network).

The dataset used for our experiments was composed of 12100 randomly selected patents downloaded from USPTO site (without any topic limitations). The patents were taken from the time span of 2005 to 2013. This approach resulted in highly diverse set of patent figures from a vast variety of domains (electronics, construction, machinery, chemistry, etc). The total of 1.1 million ROIs was extracted from the patent pages using the RLSO segmentation procedure.

Out of those 12100 patents only 197 were text-only patents, from which 10458 text ROIs were extracted (less than 1% of all ROIs).

Timing of Processing Stages

The most time-consuming part of our method was training siamese neural network to obtain approximating feature mapping for Jaccard kernel (more than 20 hours). Docstrum descriptor computation and transforming Docstrum feature vectors using kernel-approximating feature mapping took $\sim$ 1.5 hours. Further, the first level clustering stage took $\sim$ 3 hours for linear kernel, $\sim$ 7 hours for each of the χ^2, intersection and Jensen-Shannon kernels and $\sim$ 10.5 hours for Jaccard kernel. All subsequent stages (the second level clustering and supercluster classification) took negligibly small time (a few seconds) as compared to the previous stages. So, most of the time in the classification pipeline is consumed by the first level clustering (especially when non-linear kernel is used).

This distribution of processing times supports our decision to split clustering into two stages performed by different algorithms (fast mini-batch K-means for the first stage and one of the more sophisticated algorithms for the second stage), since otherwise the clustering would be prohibitively slow for any of the well-performing algorithms.

Results for Approximate Kernel Variant

We evaluated classification performance of the approximate kernel variant of our method using the method outlined above for all combinations of variable parameters involved in the second level clustering and supercluster classification stages. The total number of tried parameter combinations for the approximate kernel variant was 406665.

Classification performance results for approximate kernel variant

Clustering algorithm	χ^2	Intersection	Jaccard	Jensen-Shannon	Linear
Affinity propagation	0.68	0.68	**0.70**	0.63	0.53
Agglomerative clustering	**0.86**	0.75	**0.90**	0.76	0.65
BIRCH	0.77	0.81	**0.90**	0.79	0.68
DBSCAN	0.80	**0.94**	0.78	0.86	0.70
HDBSCAN	0.68	**0.89**	0.81	0.76	0.64
K-Means	0.75	0.78	**0.81**	0.68	0.68
Mean shift	**0.70**	0.66	0.64	0.65	0.61
One-class SVM	**0.79**	0.73	0.60	0.69	0.76
Spectral clustering	**0.88**	0.85	0.87	0.76	0.78

Table 2 shows the classification performance results for this variant: each table cell contains the result (F_1 score) for the parameter combination which delivers the highest F_1 score for a corresponding pair of the second level clustering algorithm and approximated kernel (indicated by row and column labels respectively). For each second level clustering algorithm the best result (in terms of F_1 score) is indicated in bold.

From this table one can see that among the kernels the best results are provided by the Jaccard, intersection and χ^2 kernels, while among the second level clustering algorithms the best results are delivered by agglomerative clustering, BIRCH, DBSCAN, HDBSCAN and spectral clustering. The best result overall is achieved using DBSCAN with intersection kernel.

Note that linear kernel results are substantially worse than those of the non-linear kernels for all used second level cluster-

ing algorithms, which supports our choice of using transform approximating one of the non-linear kernels as a first stage of our classification method.

To visualize precision-recall results we used the following approach: for each second level clustering algorithm (or each approximated kernel) we compute convex hull of all the precision-recall points associated with this algorithm (or kernel) and plot upper right segments of this convex hull (which correspond to the best results, i.e. both high precision and high recall). Fig. 10 shows precision-recall (PR) curves plotted using this approach for the approximate kernel variant of our method. Those curves support the conclusions of the above analysis of Table 2.

We selected three best-performing results for further analysis (underlined in Table 2): DBSCAN with intersection kernel, spectral clustering with χ^2 kernel and agglomerative clustering using complete linkage with χ^2 kernel. Although some other settings provide better F_1 score than the latter two of the selected results, they offer no improvement over DBSCAN with intersection kernel in terms of both precision and recall, while agglomerative clustering with χ^2 kernel offers better recall (at the cost of worse precision) and spectral clustering with χ^2 kernel provides a good trade-off between the two.

Results for Exact Kernel Variant

For the exact kernel variant of our method the total number of tried parameter combinations was 438170. Table 3 shows the classification performance results for this variant.

Classification performance results for exact kernel variant

Clustering algorithm	χ^2	**Hellinger**	**Intersection**	**Jaccard**	**Jensen-Shannon**	**Linear**
Affinity propagation	0.67	0.70	**0.74**	**0.74**	0.67	0.53
Agglomerative clustering	0.94	**0.94**	0.94	0.94	0.94	0.65
DBSCAN	**0.93**	**0.93**	0.89	0.89	**0.93**	0.70
HDBSCAN	0.76	**0.78**	0.68	0.68	**0.78**	0.64
One-class SVM	0.34	0.29	0.41	0.42	0.32	**0.66**
Spectral clustering	0.87	0.84	0.90	**0.93**	0.87	0.78

From this table one can easily see that among the kernels the best results are provided by the Jaccard and χ^2 kernels (similarly to the approximate kernel variant), as well as Hellingers and Jensen-Shannon kernels. Among the second level clustering algorithms the best results are provided by agglomerative clustering, DBSCAN and spectral clustering algorithms (again similarly to the approximate kernel variant).

From these results one can see that this variant offers a viable alternative to the much more computationally-demanding approximate kernel variant, since the most time-consuming stage (first level clustering) for linear kernel is more than twice faster than for any of the non-linear kernels.

Discussion of Best-Performing Configurations

Since the first level clustering stage using linear kernel (utilized in the exact kernel variant) outputs larger number of mixed clusters (thereby providing less pure separation) than when using non-linear kernels, we selected for further analysis three best-performing configurations out of all the configurations tried for the approximate kernel variant.

Best-performing configurations summary

2^{nd} **level clustering algorithm, approximated kernel**	2^{nd} **level clustering parameters**	**Selected figure ROIs, %**	**Percentile value, %**	F_1, **precision, recall**
Agglomerative clustering, χ^2	$N_{clusters} = 2$, complete linkage	28	0	0.86 0.75 1.00
Spectral clustering, χ^2	$N_{clusters} = 10$, K-means label assignment	23	12.5	0.88 0.82 0.95
DBSCAN, intersection	$\varepsilon = 0.43$ ($N_{clusters} = 5$), $N_{samples} = 5$	18	0	0.94 1.00 0.89

The summary of the second level clustering algorithm parameters as well as supercluster classification parameters (percentile value) delivering the best results in the approximate kernel variant is given in Table 4.

From this table one can see that in terms of precision/recall performance these three variants provide two extreme cases and a trade-off in-between: on one extreme is unstructured agglomerative clustering with χ^2 kernel providing 100% recall (and 75% precision), on the other extreme is DBSCAN with intersection kernel providing 100% precision (and 89% recall), while spectral clustering with χ^2 kernel achieves intermediate 82% precision and 95% recall.

For agglomerative clustering with χ^2 kernel the optimal number of (super)clusters is two, which makes it possible to use this configuration for unsupervised mode. This result is quite natural for agglomerative clustering algorithm with complete linkage which is known for its "rich get richer" behavior resulting in small number of uneven-sized clusters. For the two remaining configurations the number of output superclusters is 10 (spectral clustering) and 5 (DBSCAN) which does not allow usage of unsupervised mode.

One important concern is a possible dependence of the optimal parameters given in Table 4 on the properties of a particular dataset for which those parameters were obtained, and, thus, usability of those optimal parameters in a general case for other datasets. For the first configuration (using agglomerative clustering) the optimal parameters include setting number of clusters to two and usage of complete linkage. Both settings are perfectly generic for the task in hand, since the aim of our method is to separate two distinct clusters, one of which being strongly dominating, so complete linkage promoting the "rich get richer" behavior is a natural choice here. For the third configuration (using DBSCAN clustering) the optimal settings include setting the radius of sample neighborhood (which is determined by the properties of

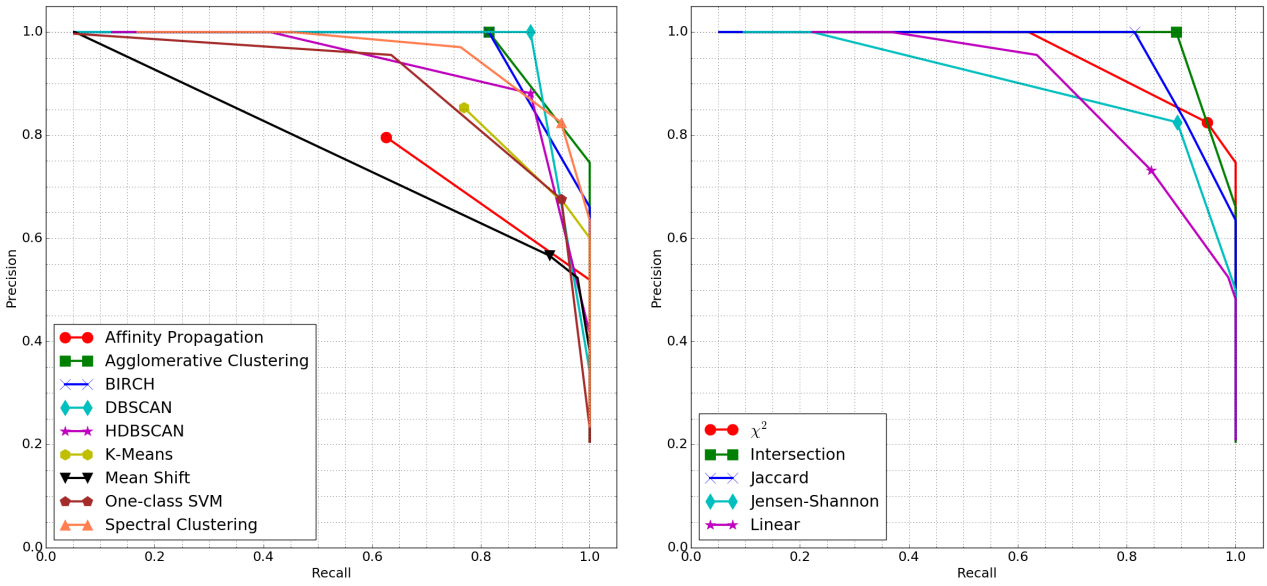

Figure 10. *PR curves for the approximate kernel variant of our method.* ***Left****: PR curves for different second level clustering algorithms,* ***right****: PR curves for different approximated kernels. Markers indicate best-*F_1 *points.*

the Docstrum descriptor, the used kernel and the classes to be separated rather than by the properties of the dataset) and the number of samples in a neighborhood for a point to be considered as a core point (the default generic value of 5 appears to be optimal). The common parameter for all configurations, percentile value, for both the first and the third configurations is zero which means that a single supercluster having the minimum text contamination is labeled as a figure supercluster, which also seems to be quite generic setting. To confirm the theoretical considerations outlined above, we applied our method using the optimal parameters from Table 4 (for the first and the third configurations) to reasonably-sized random subsets of our dataset and visually inspected the output. This experiment showed good text/figure separation for different kinds of patent figure and text regions, thus supporting the idea of usability of those optimal settings for a general case.

However, for the second configuration (using spectral clustering) the optimal parameter values ($N_{clusters} = 10$ and percentile value of 12.5%) do not exhibit the same stability for different subsets of our dataset, neither do they have a strong theoretical ground for their generality, since $N_{clusters}$ which is above two may indeed indicate the number of subclasses of main classes present in the particular dataset (e.g. different kinds of text or figure ROIs). Thus this configuration should be used with caution on datasets which are significantly different from the patent ROI dataset we used to derive it.

Finally, it is noteworthy to mention that our results for classification performance comparison for the second level clustering algorithms are in a good consistency with the behavior of different clustering algorithms illustrated in [43]: only three clustering algorithms were able to correctly handle both non-convex cluster toy datasets (one of which also contains nested clusters): agglomerative clustering, spectral clustering and DBSCAN and these are the three best-performing second level clustering algorithms in the approximate kernel variant of our method. This observation suggests us an intuition that for our dataset we may have similar supercluster geometry (non-convex and possibly even nested). To get an insight into the geometry of the text/figure clusters, we performed non-linear dimensionality reduction on Docstrum feature vectors transformed by the mapping approximating intersection kernel using t-SNE method and used the resulting 2D embedded space to plot the point clouds corresponding to feature vectors labeled as "text" and "figure" by our method using the third configuration of Table 4 (the best one in terms of F_1 score). The scatter plot of those point clouds in 2D embedded space is shown in Fig. 11

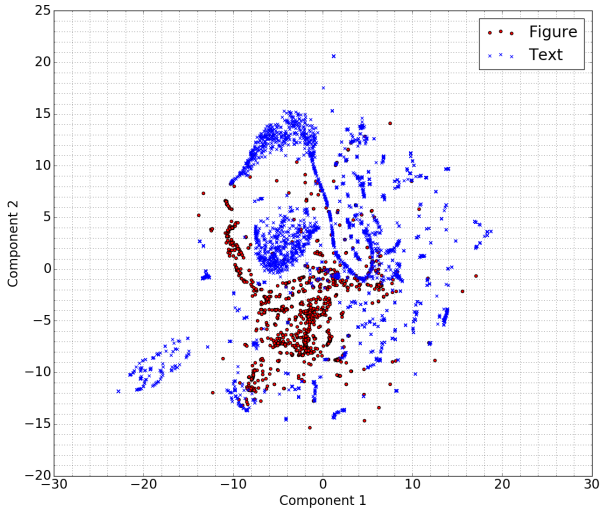

Figure 11. *Scatter plot for t-SNE components computed for Docstrum feature vectors of ROIs classified as being text and figure ROIs by our method.*

One can see that "text" points form several clusters, some of

them having sophisticated non-convex geometry and being quite distant from one another, while a majority of "figure" points form a dense well-localized cluster surrounded by the text clusters. Although this embedded space may lack dimensions to fully represent the original cluster geometry, the scatter plot in this space still serves to support our intuition and, thus, the choice of the best performing second level clustering algorithms. This sophisticated cluster geometry may also be responsible for the inferior classification performance of configurations using one-class SVM. Since one-class SVM is a novelty detection method, it is a natural choice for our task, where we have training subset for the text class and aim at identifying figure class as outliers for the text class. However, even when using generalized RBF and sigmoid kernels (through the combination of histogram-oriented kernel in kernel-approximating feature mapping stage and RBF or sigmoid kernel in SVM itself), one-class SVM fails to achieve classification performance of the algorithms which are especially efficient in handling complex cluster geometry (such as DBSCAN and agglomerative clustering).

Conclusion

We proposed a new method of text/figure separation for binary document images containing highly diverse set of figure classes (e.g. patents). Our method offers significant advantages over the previously-known approaches: it can separate many figure classes misclassified as text by more simple methods, is suitable for processing large scale datasets and is capable of working in either unsupervised mode (using only general a-priori information about the text/figure distribution in the dataset) or in semi-supervised mode (using small subset of the data labeled as text). These advantages come from the usage of the Docstrum descriptor, kernel-approximating feature mappings for various histogram-oriented kernels (including χ^2, intersection and Jaccard) and efficient two-level clustering (using mini-batch K-means, agglomerative clustering and DBSCAN algorithms).

We showed the efficiency of our method on very large and diverse patent image dataset, where it achieves F_1 scores of 0.86 for unsupervised mode and 0.94 for semi-supervised mode. The optimal parameters for the various stages of our method are obtained using extensive parameter optimization performed on this dataset. We analyzed the experimental results of our method and motivated our design choices by both theoretical considerations and experimental evidences.

Our method may be used as a preprocessing stage for various document image processing tasks: document indexing and retrieval, document object detection and recognition, OCR, document compression and many more.

Acknowledgments

The authors wish to thank Prof. Mikhail Rychagov for continued support during the course of this work.

References

[1] O. Okun, D. Dœrmann, and M. Pietikainen, Page Segmentation and Zone Classification: the State of the Art, no. LAMP-TR-036. OULU UNIV (FINLAND) DEPT OF ELECTRICAL ENGINEERING, 1999.

[2] J. Duong, M.Côté, H. Emptoz, and C.Y. Suen, Extraction of Text Areas in Printed Document Images, Proceedings of the 2001 ACM Symposium on Document Engineering, pp. 157-165, 2001.

[3] F. Zirari, A. Ennaji, S. Nicolas, and D. Mammass, A Simple Text/Graphic Separation Method for Document Image Segmentation, 2013 ACS International Conference on Computer Systems and Applications (AICCSA), IEEE, pp. 1-4, 2013.

[4] K. Tombre, S. Tabbone, L. Pélissier, B. Lamiroy, and P. Dosch, Text/Graphics Separation Revisited, International Workshop on Document Analysis Systems, Springer Berlin Heidelberg, pp. 200-211, 2002.

[5] P.P. Rege, and C.A. Chandrakar, Text-image Separation in Document Images Using Boundary/Perimeter Detection, International Journal on Signal and Image Processing, vol. 4, no. 1, pp. 29-35, 2013.

[6] D. Keysers, F. Shafait, and T.M. Breuel, Document Image Zone Classification - A Simple High-performance Approach, 2nd Int. Conf. on Computer Vision Theory and Applications, 2007.

[7] N. Priyadharshini, and M.S. Vijaya, Document Segmentation and Region Classification Using Multilayer Perceptron, IJCSI International Journal of Computer Science Issues, vol. 10, no. 2, 2013.

[8] R. Minetto, N. Thome, M. Cord, N.J. Leite, and J. Stolfi, T-HOG: An Effective Gradient-based Descriptor for Single Line Text Regions, Pattern Recognition, vol. 46, no. 3, pp. 1078-1090, 2013.

[9] T.V. Hoang, and S. Tabbone, Text Extraction from Graphical Document Images Using Sparse Representation, Proceedings of the 9th IAPR International Workshop on Document Analysis Systems, ACM, pp. 143-150, 2010.

[10] T.H. Do, S. Tabbone, and O. Ramos-Terrades, Text/Graphic Separation Using A Sparse Representation with Multi-learned Dictionaries, 21st International Conference on Pattern Recognition (ICPR 2012), pp. 689-692, 2012.

[11] M. Zhao, S. Li, and J. Kwok, Text Detection in Images Using Sparse Representation with Discriminative Dictionaries, Image and Vision Computing, vol. 28, no. 12, pp. 1590-1599, 2010.

[12] J. Gllavata, R. Ewerth, and B. Freisleben, Text Detection in Images Based on Unsupervised Classification of High-Frequency Wavelet Coefficients, Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004), vol. 1, pp. 425-428, 2004.

[13] C. Liu, C. Wang, and R. Dai, Text Detection in Images Based on Unsupervised Classification of Edge-based Features, Eighth International Conference on Document Analysis and Recognition (ICDAR'05), IEEE, pp. 610-614, 2005.

[14] S. Ferilli, F. Leuzzi, F. Rotella, and F. Esposito, A Run Length Smoothing-Based Algorithm For Non-Manhattan Document Segmentation, Convegno del Gruppo Italiano Ricercatori in Pattern Recognition 2012, 2012.

[15] M.K. Hu, Visual Pattern Recognition by Moment Invariants, IRE Trans. Info. Theory, vol. 8, no. 2, pp. 179-187, 1962.

[16] R.M. Haralick, and K. Shanmugam, Textural Features for Image Classification, IEEE Transactions on Systems, Man, and Cybernetics, vol. 3, no. 6, pp. 610-621, 1973.

[17] S. Belongie, J. Malik, and J. Puzicha, Shape Matching and Object Recognition Using Shape Contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pp. 509-522, 2002.

[18] A. Gordo, F. Perronnin, and E. Valveny, Large-scale Document Image Retrieval and Classification with Runlength Histograms and Binary Embeddings, Pattern Recognition, vol. 46, no. 7, pp. 1898-1905, 2013.

[19] T. Ojala, M. Pietikainen, and T. Maenpaa, Multiresolution Gray Scale and Rotation Invariant Texture Classification with Local Bi-

nary Patterns, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971-987, 2002.

[20] P. Sidiropoulos, S. Vrochidis, and I. Kompatsiaris, Adaptive Hierarchical Density Histogram for Complex Binary Image Retrieval, 2010 International Workshop on Content-Based Multimedia Indexing (CBMI), pp. 1-6, 2010.

[21] L. O'Gorman, The Document Spectrum for Page Layout Analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 11, pp. 1162-1173, 1993.

[22] A. Vedaldi, and A. Zisserman, Efficient Additive Kernels via Explicit Feature Maps, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 3, pp. 480-492, 2012.

[23] B. Schölkopf, The Kernel Trick for Distances, Advances in Neural Information Processing Systems, vol. 13, pp. 301-307, 2001.

[24] K. He, X. Zhang, S. Ren, and J. Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, Proceedings of the IEEE International Conference on Computer Vision, pp. 1026-1034, 2015.

[25] X. Glorot, and Y. Bengio, Understanding the Difficulty of Training Deep Feedforward Neural Networks, Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, vol. 9, pp. 249-256, 2010.

[26] T. Dozat, Incorporating Nesterov Momentum into Adam, Stanford University, Tech. Rep., `http://cs229.stanford.edu/proj2015/054_report.pdf`. 2015.

[27] D. Sculley, Web-Scale K-Means Clustering, Proceedings of the 19th International Conference on World Wide Web, ACM, pp. 1177-1178. 2010.

[28] D. Arthur, and S. Vassilvitskii, k-means++: The Advantages of Careful Seeding, Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, pp. 1027-1035, 2007.

[29] B.J. Frey, and D. Dueck, Clustering by Passing Messages Between Data Points, Science, vol. 315, no. 5814, pp. 972-976, 2007.

[30] D. Comaniciu, and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 603-619, 2002.

[31] A.Y. Ng, M.I. Jordan, and Y. Weiss, On Spectral Clustering: Analysis and an Algorithm, Advances in Neural Information Processing Systems, vol. 2, pp. 849-856, 2002.

[32] D. Müllner, Modern Hierarchical, Agglomerative Clustering Algorithms, arXiv preprint arXiv:1109.2378, 2011.

[33] M. Ester, H.P. Kriegel, J. Sander, and X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, vol. 96, no. 34, pp. 226-231, 1996.

[34] R. J. Campello, D. Moulavi, and J. Sander, Density-Based Clustering Based on Hierarchical Density Estimates, Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, pp. 160-172, 2013.

[35] T. Zhang, R. Ramakrishnan, and M. Livny, BIRCH: An Efficient Data Clustering Method for Very Large Databases, ACM Sigmod Record, vol. 25, no. 2, pp. 103-114, ACM, 1996.

[36] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson, Estimating the Support of a High-Dimensional Distribution, Neural Computation, vol. 13, no. 7, pp. 1443-1471, 2001.

[37] P.J. Rousseeuw, and K.V. Driessen, A Fast Algorithm for the Minimum Covariance Determinant Estimator, Technometrics, vol. 41, no. 3, pp. 212-223, 1999.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, vol. 12, no. Oct, pp. 2825-2830, 2011.

[39] D. Müllner, Fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python, Journal of Statistical Software, vol. 53, no. 9, pp. 1-18, 2013.

[40] F. Chollet, Keras: Deep Learning Library for Theano and TensorFlow, GitHub repository, `https://github.com/fchollet/keras`. 2015.

[41] F. Alted, I. Vilata, S. Prater, V. Mas, T. Hedley, and A. Valentino, PyTables: Hierarchical Datasets in Python, Available from World Wide Web: http://www.pytables.org. 2002.

[42] S. Van der Walt, J.L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J.D. Warner, N. Yager, E. Gouillart, and T. Yu, Scikit-image: Image Processing in Python, PeerJ, vol. 2, p. e453, 2014.

[43] Scikit-learn developers, Comparing Different Clustering Algorithms on Toy Datasets, Scikit-learn User Guide, `http://scikit-learn.org/0.18/auto_examples/cluster/plot_cluster_comparison.html`. 2016.

Author Biography

Valery Anisimovskiy received his BS (1998) and MS (2000) in applied mathematics and physics from the Moscow Institute of Physics and Technology (MIPT). Since then he worked at the Institute for Nuclear Research of RAS (2000-2004), SPIRIT Corp (2004-2009), Luxoft (2009-2011), Huawei Russian Research Center (2011-2014). Since 2014 he is working at Samsung R&D Institute Russia, Moscow. His research interests include machine learning, pattern recognition, computer vision, video coding and audio coding.

Ilya Kurilin received his MS degree in radio engineering at Novosibirsk State Technical University (NSTU), Russia in 1999 and his PhD degree in theoretical bases of informatics at NSTU in 2006. In 2007 Dr. I. Kurilin joined Image Processing Group, Algorithm Lab at Samsung R&D Institute Russia, where he is engaged in photo/document image enhancement and image understanding projects.

Andrey Shcherbinin received his MS (1997) in applied mathematics from the Moscow Institute of Steal and Alloys (MISiS). Since then he worked in Orgtekhdiagnostics Ltd (1998-2004), Vildis Ltd (2004-2011), 1C Ltd (2011-2013). Since 2013 he is working at Samsung R&D Institute Russia, Moscow. His research interests include machine learning, pattern recognition, computer vision, video enhancement and content restoration/enrichment.

Petr Pohl received MSc (2002) in Technical Cybernetics from Faculty of Electrical Engineering (FEE) of Czech Technical University (CTU). Since then he worked in Neovision Ltd. (2002-2011) and Samsung R&D Institute Russia (2011-today). Main interests include computer vision and image and video processing. Latest works are related with motion estimation, temporal interpolation and dense tracking.