MI3: Machine-Initiated Intelligent Interaction for Interactive Classification and Data Reconstruction

In many applications, while machine learning (ML) can be used to derive algorithmic models to aid decision processes, it is often difficult to learn a precise model when the number of similar data points is limited. One example of such applications is data reconstruction from historical visualizations, many of which encode precious data, but their numerical records are lost. On the one hand, there is not enough similar data for training an ML model. On the other hand, manual reconstruction of the data is both tedious and arduous. Hence, a desirable approach is to train an ML model dynamically using interactive classification, and hopefully, after some training, the model can complete the data reconstruction tasks with less human interference. In order for this approach to be effective, the number of annotated data objects used for training the ML model should be as small as possible, while the number of data objects to be reconstructed automatically should be as large as possible. In this paper, we present a novel technique for the machine to initiate intelligent interactions to reduce the user's interaction cost in interactive classification tasks. The technique of machine-initiated intelligent interaction (MI3) builds on a generic framework featuring active sampling and default labelling. To demonstrate the MI3 technique, we use the well-known Cholera Map visualization as an example as it features three instances of MI3 pipelines. The experiment has confirmed the merits of the MI3 technique.

CCS Concepts: • **Theory of computation** → **Active learning**; • **Human-centered computing** → **Interactive systems and tools**; *Visualization*; • **Applied computing** → *Data recovery*; • **Computing methodologies** → Object identification.

Additional Key Words and Phrases: Data reconstruction, interactive classification, data annotation, active learning, interaction reduction, historical visualization

ACM Reference Format:
. 2019. MI3: Machine-Initiated Intelligent Interaction for Interactive Classification and Data Reconstruction. *ACM Trans. Intell. Syst. Technol.* 1, 1, Article 1 (January 2019), 27 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

History has left us many wonderful visualization images, such as William Playfair's time series chart of trade-balance (1786) [32], bar chart of Scotland's imports/exports (1786) [32], and pie chart of Turkish Empire's land holdings (1801) [33]; Charles Joseph Minard's flow map of road traffic (1845) [27], and flow map of Napoleon's Russian campaign (1869) [28]; John Snow's cholera map (1855) [42]; Florence Nightingale's coxcomb chart (1858) [31][30]; and so on. These visualization images capture important statistics of historical events, and therefore are of great interest to scholars in humanities and social sciences. In numerous cases, the original datasets are lost. It is desirable to reconstruct the datasets from the visualization images.

Reconstruct a dataset manually with pen-and-ruler is laborious. Many attempts have been made to extract data from computer-generated visualization images. Some [10, 39, 51] used hand-crafted image processing algorithms, while others [1, 15, 19, 24, 34, 35, 40] used machine learning to derive models for recognizing visual objects that encode data.

Author's address:

2157-6904/2019/1-ART1 $15.00
https://doi.org/0000001.0000001

However, unlike computer-generated imagery, a historical visualization image typically features a unique visual design. In addition, the hand-drawn nature of the visualization and the deterioration of and the damages to the papers pose further challenges to the process of recovering the data from historical visualization images. If there were a fully-automated solution, one would have to develop an algorithm for each image individually by programming or using machine learning. Likely the algorithm would be unsuitable for other historical visualization images. Therefore, the cost of manually recovering data would likely be lower than programming, and ironically would be almost the same as the cost of preparing a training dataset before the actual machine learning process.

A practically more effective solution would be an interactive intelligent system that is equipped with a collection of elementary algorithms, can ask users questions intelligently, and is able to adapt its algorithms automatically to work with a given historical visualization image. It is not an idealized system that could recover data automatically from many different historical visualization images, but a computerized assistant what can initiate intelligent interactions with human users in order to adapt itself for each specific task. The design goal of "intelligent interactions" is to minimize the number of interactions.

In this paper, we present a generic approach for machine-initiated intelligent interactions (MI3). The approach is governed by an iterative machine learning framework that features *algorithmic sampling (active sampling)* for dynamically acquiring labels from the user and *algorithmic default labelling (label propagation)* for maximizing the informational value of the user's inputs. It consists of a collection of image processing algorithms to accommodate the needs for detecting different types of visual objects. The goal of the MI3 approach is to perform the data reconstruction task with as few interactions as possible. Using machine learning to train a model is primarily for supporting the task on hand at the moment, rather than for deriving a model that can be reused for many other visualization images (since there are seldom any similar images). The main contributions of this work include:

- a generic approach for machine-initiated intelligent interactions (MI3) in the context for recovering data from historical visualization images;
- a demonstration of actualizing the MI3 approach in three functional pipelines;
- a quantitative evaluation of the effectiveness of the MI3 approach in the implementation of these three pipelines;
- a prototype system that supports data reconstruction from spatial data visualization.

2 RELATED WORK

2.1 Chart Data Reconstruction

Reconstruction of data from visualization images has attracted research interest in the HCI and image processing communities for the last two decades. Many valuable datasets are only available through their visualization imagery in printed or scanned media. Manually reconstructing a dataset from a visualization image with pen-and-ruler is accurate but laborious, and is often not scalable for charts with many data objects, such as John Snow's cholera map [42]. Many techniques have been proposed to reduce human effort in data reconstruction.

Several tools [4, 11, 46] provide a digital extension of the pen-and-ruler approach by allowing users to interactively inform the computer about where is the data to be recovered. For example, Ycasd [11] is a tool for digitizing line charts. It requires the user to indicate the location of axes, specify the scale of the axes, and point out individual data points. The total number of interactions required is thus at the same scale as the number of data points to be digitized. We will compare our MI3 approach with this computerized pen-and-ruler approach in Section 6.

Many techniques have been proposed to process visualization images automatically, and reconstruct data records from recognized data objects (e.g., [10, 37, 39, 51]). Each technique is constructed based on the known visual specification of a particular type of charts. There are often human-controllable parameters for ensuring a good match between the techniques and minor variations of the charts within the same type group. Such techniques are suitable for those commonly-used and computer-generated statistical charts. However, they rarely work well with historical visualization images, typically hindered by the non-standard visual design, the distortion of the hand-drawn shapes, and the noise due to deterioration of the paper media.

In order to ensure high-quality data reconstruction, semi-automatic image processing techniques have been deployed in some systems [18, 26]. iVoLVER [26] is such an interactive system, with which the user specifies how visual objects map to data. ChartSense [18] is a mixed-initiative system, which enables users to determine image processing parameters. As our application concerns the recovering of valuable data from historical visualization, the quality of data reconstruction is a crucial requirement. Hence involving human users in the reconstruction process is unavoidable. Nevertheless, we also recognize that a user's specification of the mapping from visual objects to data and various image processing parameters can guarantee the accuracy of data reconstruction from historical visualization, and any trial-and-error interactions could easily incur undesirable effects (e.g., frustration, cognitive load, and time cost). We have thus designed our user interface to focus on the questions that users can answer easily without any explicit knowledge of the underlying algorithms for image processing or mapping specifications. We have introduced machine learning and machine-initiated intelligent interactions (MI3) to reduce the burden of the user.

To avoid the necessity for a precise specification of a type of charts, machine learning (ML) has been used to construct algorithmic models and define their parameters by using human-annotated datasets as the training data (e.g., [1, 15, 19, 24, 34, 35, 40]). Al-Zaidy and Gile applied this approach to data reconstruction from bar charts by using decision-tree-based classifiers [1]. Poco and Heer used SVM to classify textual elements in visualization images [34], and Poco et al. further developed a technique for extracting the colour encoding of visualization images by recognizing the legend [35]. The combination of machine learning and image processing presents an attractive advantage when there are many annotated visualization images available for training a reasonably-accurate model, and there are many more not-yet-annotated visualization images for which the trained model can be used to recover the unknown data. Such an advantage is absent with historical visualization images since the number of similar datasets available usually is insufficient as the training data. By the time when all similar visualization images are annotated for training a model, there is no need for the model anymore.

This naturally leads to the idea that one may select parts of an image to train a model with possibly mediocre accuracy, and applied the trained model for the rest part of the image, and if any, some other similar images. This idea is the basis of this work. We further enhance this idea by introducing an algorithmic selection of "parts of an image" and algorithmic provision of "tentative labels" in order to reduce the number of interactions required for interactive classification. We will evaluate the merits of these two additions in Section 6.

2.2 Interactive Classification with Intelligent User Interfaces

Data annotation is an essential, and often costly, step for any supervised learning techniques. In several applications, intelligent user interfaces for interactive classification have been used to reduce the cost of data annotation. Fails and Olsen proposed Crayons [8], an interactive classification technique for image pixel classification, that learns the labels generated from user's painting interaction. In the area of image searching, Fogarty et al. developed a search system, CueFlik [9], with which the user can define search criteria for a concept by using positive and negative examples

and rank search results according to the similarity to the concept. Active learning is used to inform the user about the images that confuse the system the most, guiding the user to provide examples that would benefit the concept classifier the most. To support efficient iterative image searching, Luo et al. proposed multi-class query ranking [25], a semi-supervised learning algorithm based on manifold ranking.

In the area of computer vision, Russell et al. developed LabelMe [38], a system for annotating objects in images. The user can specify an object contour by freehand drawing. Andriluka et al. developed an interactive system, Fluid Annotation [2], for annotating ground-truth results for image segmentation. It uses a neural network model trained in advance to compute an initial set of segments in an image and ask the user to edit the geometry of the segments to correct any errors. In our MI3 approach, the ML framework does not assume the availability of any pre-labelled training data, and instead, it provides users with default labels to aid their annotation tasks dynamically.

Techniques for accelerating the interactive classification process in the literature are typically categorized into *active learning based methods* and *clustering based methods* [43]. A method in the former category attempts to select and label more informative data points and use them to train an interim model that is used to label other data points. With an iterative process, the interim model becomes better and better, hence reducing the number of data points needed to be manually annotated. The latter attempt to make use of each human annotation to label more data points. We give below several examples of clustering-based methods, and we will discuss the active learning-based methods in the next subsection.

Cui et al. described an interactive photo annotation system, EasyAlbum [7], allowing the user to annotate data points in a cluster-by-cluster manner. Liu et al. described a method that first divides unlabelled data points into clusters, selects exemplars from each cluster for the user to label, and propagates the labels to other data points in the cluster [23]. Rafailidis et al. described the content-based tag propagation technique that propagates user-provided tags to similar items to address the "cold start" problem and boost the accuracy of tag-based search engine [36]. Kucher described ALVA, an interactive classification technique for text data annotation and visualization of the annotation. ALVA exploits active learning in the process of annotating text dataset with multiple non-exclusive labels. To visualize the label of data points, each of which is a vector of binary values, they propose a visual representation called CatCombos that groups data points with the same label vector [20]. Tian et al. described a hybrid method that first groups unlabelled data points into evident clusters and a background cluster. It then allows the user to annotate each evident cluster as a whole, and guides the user to annotate the background cluster using an active learning-based method [44]. Tang et al. described a multi-scale method that allows the user to label data points in a cluster-by-cluster manner, and to refine the labels in each cluster iteratively using the same clustering-based mechanism [43]. The generic MI3 approach includes an algorithmic default labelling component, which in principle can be an algorithmic clustering-based method, an interactive clustering method, or any other future method for label propagation.

2.3 Active Learning and Semi-Supervised Learning

Active learning is a family of ML methods that interact with the user during a learning process, typically (in a narrow definition) for seeking labels for unlabelled training data points, and in some cases (in a broad definition), for seeking important decisions that can improve the quality and performance of the ML process.

A critical feature in many active learning methods is to select data points "intelligently" to seek labels from the user. Lewis and Catlett proposed such a method, which estimates the uncertainty of each unlabelled data point, and selects the data point with the highest uncertainty for the user to label [22]. Brinker introduced a batch active learning method that selects a batch of unlabelled

data points for the user to label [5]. Xu et al. described a batch method that selects data points for labelling based on their relevance, density, and diversity measures [47]. Nguyen and Smeulders described a method that makes use of clustering property of the data and selects data points in favour of dense clusters [29]. Guo and Schuurmans described a batch method that selects the data points in a manner that maximizes the discriminative performance of the target classifier while minimizing the entropy of the missing labels [12]. In addition, active learning has been used for accelerating interactive classification [14, 17].

The MI3 approach presented in this paper contains an active learning component. In particular, MI3 actively initiate intelligent interactions by selecting a batch of data points for the user to label. In the implementation of MI3 pipelines reported in this paper, we used the methods by Lewis and Catlett [22] and Xu et al. [47].

Semi-supervised learning is a family of ML methods that make use of both labelled and unlabelled data points for training a model. For example, Chapelle et al. proposed a framework for incorporating unlabelled data in kernel classifiers [6]. Leistner et al. extended the random forests method with a semi-supervised mechanism [21]. They reformulated the optimization goal of maximizing multi-class margin by making use of unlabelled data in addition to labelled data. Yarowsky proposed self-training [48], a boosting technique, that uses pseudo-labels of unlabelled data during training. Using a classifier learned on labelled data as an interim ML model, the technique applies the interim model to unlabelled data, and extends the training dataset that is used to retrain the model. Blum and Mitchell proposed co-training [3], a similar technique to self-training, that uses two interim classifiers trained using two disjoint subsets of the data features. The predicted tentative labels on unlabelled dataset from one of the classifiers is feed to the other to enlarge the training set. Joachims developed the TSVM (transductive support vector machine) method as an extension of the traditional SVM by involving unlabelled data in addition to labelled data for margin maximization in the model construction process [16]. Zhu and Ghahramani described a graph-based algorithm for transferring known labels to unlabelled data points iteratively [52]. Zhou et al. described another graph-based algorithm with a different label propagation strategy [50]. Tong and Jin described a mixed label propagation algorithm [45] that exploits both the similarity and dissimilarity information.

Typical unsupervised methods, such as clustering, can be used for assisting in label propagation. Therefore, an implementation of the MI3 approach may also integrate semi-supervised learning algorithms. In particular, we designed a graph-based label propagation process based on Zhou et al.'s method [50] as one of the optional algorithmic default labelling components in our implementation of MI3 pipelines.

3 OVERVIEW OF THE MI3 APPROACH

In this section, we first describe an example case that illustrates the challenges in a class of data reconstruction applications. We then outline a generic approach for supporting such applications.

3.1 Examples of Technical Challenges: John Snow's Cholera Map

The 19th-century witnessed the boom of visualizations techniques. Among many well-known historical visualizations created at that time, the Cholera Map, which was created by John Snow in 1854 [41], provides a telling example for illustrating the challenges in data reconstruction from historical visualization images. As shown in Fig. 1, the map depicts the spatial distribution of fatalities in an area during the 1854 London cholera outbreak.

The Cholera Map encodes a precious historical dataset, which can be stored as an array of records, each of which is a tuple $(x, y, \#victims)$. The goal of data reconstruction is thus to scan the map, identify the location (x, y) of each bar with at least one block, and count the number of blocks as the value of $\#victims$. As illustrated on the right of Fig. 1, the identification of each

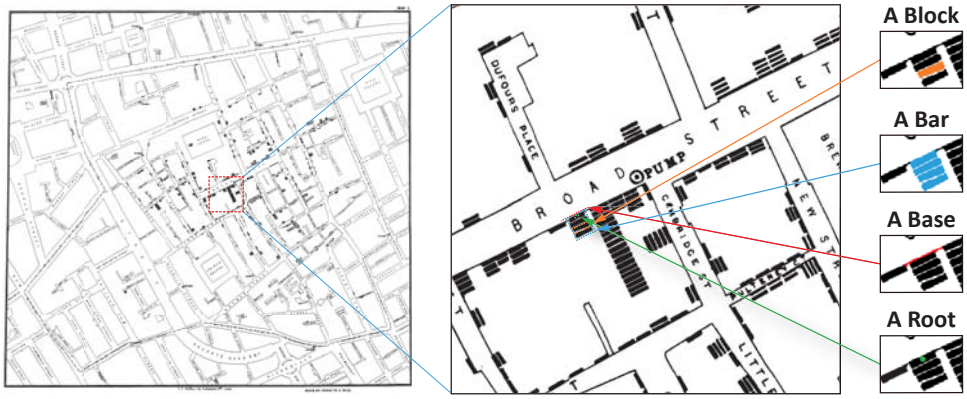


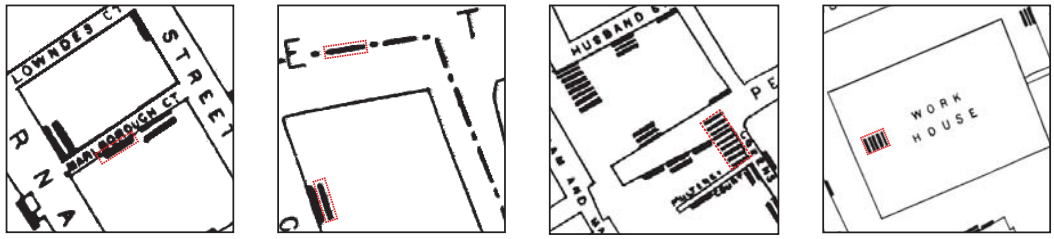
Fig. 1. Cholera Map [41] shows the number of fatalities in the 1854 London Cholera Outbreak with a discrete bar chart. The fatalities are distributed at more than 300 locations on the map and visualized as stacked blocks. The right figure illustrates the visual encoding of Cholera Map. A bar aggregates the fatalities at a location, where the number of blocks denotes the number of fatalities. A bar typically sits on a base and has a root point that denotes its location information.

location x, y requires several object recognition processes for identifying the corresponding bar, the blocks that form the bar, the base that determines the first block, and the root position (i.e., a consistently-defined position in the first block).

The Cholera map consists of 579 blocks that form 321 bars. Manually acquiring the values for the 321 records will be tedious and time-consuming. Even with an optimistic estimation, one were to take 10 seconds to navigate to a bar, 10 seconds to count the number of blocks in the bar, 20 seconds to measure the location, and 10 seconds to type in the values into the computer, it would take some 50 seconds per bar and more than 4 hours to reconstruct the dataset.

Naturally, one would like to automate this process as much as possible. However, data reconstruction from historical visualizations poses several challenges. Many of such visualizations were hand-drawn, often featuring unique visual designs and suffering from poor image quality. Any general-purpose algorithms for detecting blocks, bars, bases or roots may deliver erroneous results when encountering difficult patterns such as those shown in Fig. 2. In (a), a block conglutinates with the baseline and the letters on the other side. In (b), a segment in a dashed line can easily be confused as a block. In (c), the grouping of blocks into bars is ambiguous, even for humans. At a glance, the highlighted pattern may be interpreted either as a bar with nine blocks or as two bars with four and five blocks respectively. It requires careful observation of other bars in the neighbourhood, and some logical reasoning in order for one to conclude that this is a bar of nine blocks. In (d), a bar does not have a baseline because it is not associated with a street location. Hence the identity of its first block is ambiguous, and so is its root position.

While one may anticipate the potential of using machine learning to train a sequence of models for detecting blocks, grouping blocks into bars, and finding the root of each bar respectively, the unavailability of a sufficient number of training data points poses another critical challenge. Many historical visualizations feature unconventional visual designs. For instance, the Cholera map is rather unique among the visualizations produced before the digital age. Given such a visualization image, the data objects (e.g., blocks, bars, or roots) that may be pre-labelled to aid a supervised learning process would constitute a very sparsely sampled and possibly biased training dataset, if there were other visualizations drawn with the same visual design. Therefore, pre-labelling



(a) Segmentation difficulty (b) Similar non-block object (c) Ambiguous grouping (d) No baseline

Fig. 2. Some examples of challenging cases in processing the Cholera Map image [41]. (a) A block conglutinates with background. (b) A segment of dashed line may be confused for a block. (c) The grouping of blocks is ambiguous. (d) Blocks are not distributed close to a baseline.

the 579 blocks, 321 bars, and 321 roots would unlikely cost less than measuring the 321 tuples $(x, y, \#victims)$ manually and typing them into the computer.

Hence, it is highly desirable to find an approach that is neither totally automated nor totally manual. Ideally, a software system enabling such an approach can encode some basic knowledge about a family of plots (e.g., bar charts, discrete bar charts, pictogram bar charts). It can ask a user questions about the difficulties and variations specific to a given visualization (e.g., noise and distortion, and root definition), and can dynamically learn from the user's answers and improve its ability to handle similar difficulties and variations. Just like an intelligent assistant, it can initiate interactions intelligently, and can learn dynamically. It is this desire that motivates the development of the MI3 approach, where MI3 stands for "Machine-Initiated Intelligent Interaction".

3.2 The Generic MI3 Pipeline for Interactive Classification

The goal of an MI3 Pipeline is to perform an interactive classification task. Many detection problems in data reconstruction are inherently classification problems. For example, given a collection of pixel-based objects in the Cholera map, to differentiate those blocks that represent fatalities from other shapes is a typical classification problem. Even when a detection problem may not immediately be seen as a classification problem, it can normally be decomposed into a classification problem plus some pre-processing and/or post-processing. For example, the problem of grouping blocks into a bar can be transformed to a classification problem for determining whether two blocks are related (i.e., belong to the same bar), together with a pre-processing step for compiling candidates of block pairs and a post-processing step for grouping those related blocks. Both the pre- and post-processing steps can be reliably computed using predefined algorithms without any user intervention. By focusing on classification problems, the MI3 approach can be applied to different pipelines and sub-pipelines in data reconstruction workflows.

Fig. 3 depicts the MI3 approach as a generic pipeline. Let $\mathbf{D}_0$ be a set of n unlabelled data objectives at step 0 (i.e., before the iteration starts). The goal of the pipeline is to deliver a labelled dataset $\binom{D_0}{L_0}$ after k iteration steps such that the amount of human effort in these k steps is less than that for labelling all n data objects manually. In this work, we use the number of human-computer interactions to approximate the amount of human effort.

In the pipeline, *data pre-processing* is an optional process, and it is used when the raw data has not yet provided a set of data objects to be labelled. For example, given the Cholera Map image, one may use a pre-processing step to extract all color connected pixel components (i.e., all connected black pixel groups and all connected white pixel groups) as candidates of blocks. Normally, this pre-processing step is totally automated. The subsequent processes will classify these candidates

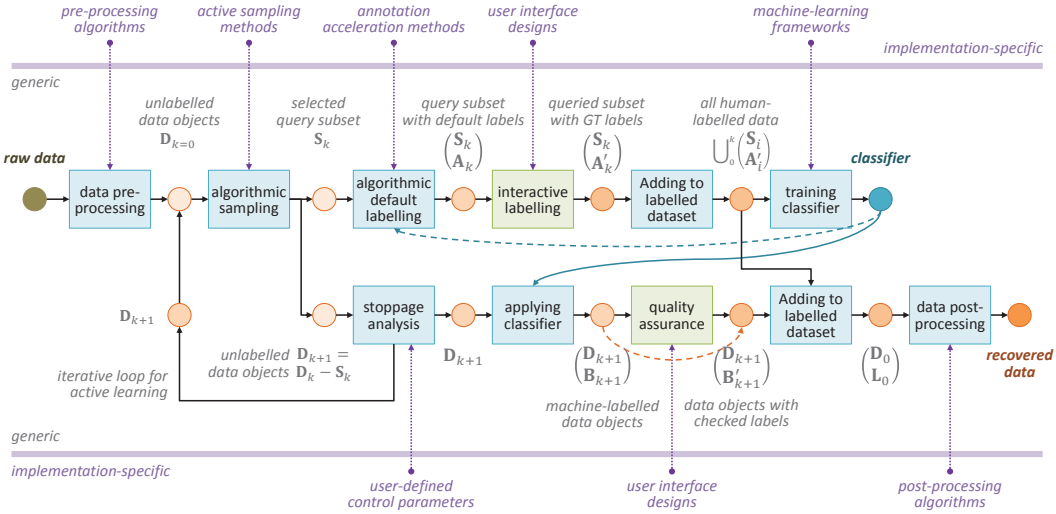


Fig. 3. The generic MI3 pipeline. The optional data preprocessing step transforms the raw data from the original problem into a classification task for unlabelled data objects D_0. We consider the k-th iteration of the active learning as an instance. Here, by "active learning", we refer to its general definition that the system actively seeks decisions from the user to aid the ML process. The algorithmic sampling step selects the data objects S_k to be labelled. The algorithmic default labelling step assigns default label annotation A_k for data objects S_k. In the interactive labelling step, the user corrects the mislabelled data objects in the interface and produces ground truth (GT) labels A'_k. The confirmed label annotations A'_k are then added to the labelled dataset. An interim classifier is trained with the partially labelled dataset, which can be used to compute default labels in the next iteration. The stoppage analysis step checks whether the interactive classification stage should stop. If not, the next session of interactive classification starts. Else, the interim classifier can be applied to label the remaining unlabelled data objects D_{k+1} with labels B_{k+1}. The user goes through a quality assurance step to verify the labels for these data objects. The data objects D_{k+1} and verified labels B'_{k+1} are then added to the labelled dataset and forms the labels L_0 for the whole data object set D_0. An optional postprocessing step transforms the labelled data objects into the desired final output data structure. The MI3 pipeline is generic and its components can be implemented with application specific algorithms.

into correct categories, such as "true blocks" and "false blocks". It is thus important in practice for this pre-processing step to minimize false negatives.

As all data objects in $\mathbf{D}_0$ are unlabelled, the core of the pipeline is an iterative loop based on active learning (according to its general definition). Considering the k-th iteration as an instance, the data flows through various component processes as follows:

- The process *algorithmic sampling* selects a subset of data objects $\mathbf{S}_k \subset \mathbf{D}_k$. While a simple solution can be random sampling, a more sophisticated solution can select data objects that can inform the classifier learning more effectively.
- The process *algorithmic default labelling* assigns a tentative label to each object in $\mathbf{S}_k$. We denote this interim labelled subset as $\binom{S_k}{A_k}$. It is helpful for those tentative labels to be as correct as possible. All incorrect labels will have to be corrected by the succeeding process manually, hence incurring more interactions.
- The process *interactive labelling* then initiates an interaction session, asking the user to check the interim labelled subset and make a correction if necessary. This results in a correctly labelled subset $\binom{S_k}{A'_k}$. The main criterion for designing an effective user interface is to enable

the checking of the tentative labels and correcting any mistakes with the minimal amount of human effort. In this work, we focus on the number of interactions as an approximation of human effort.

- In the process *adding to labelled dataset*, the labelled subset $\binom{S_k}{A'_k}$ is then combined with all labelled subsets in the previous $k-1$ iterations, resulting in a larger set of labelled data objects $\bigcup_0^k \binom{S_i}{A'_i}$, which can be used to train a classifier while forming part of the classification results that the pipeline is designed to deliver. Here we define the union of a series of set-pairs as the pair of two sets, each resulting from the union of the corresponding series of sets.
- The process *training classifier* is an automated ML process that uses the labelled data objects $\bigcup_0^k \binom{S_i}{A'_i}$ to train a classification model. Because this training process is iteratively invoked with bigger and bigger training datasets, the ML models are expected to become better and better. The interim model can be used to classify unlabelled data objects in the processes of *algorithmic default labelling* and *applying classifier*.
- The process *stoppage analysis* makes an algorithmic decision as to the trained model obtained after the k iteration steps is good enough for classifying unlabelled dataset $\mathbf{D}_{k+1}$. This decision can be made using the information collected in the previous k iterations, e.g., the testing measures in the process *training classifier*, the percentage of the labels that need to be corrected in the process *interactive labelling*, and some statistical measures about the labelled data and unlabelled data. If the trained classifier is equipped with uncertainty analysis, there can be an additional *stoppage analysis* process after the process of *applying classifier*.
- The process *applying classifier* is invoked once the *stoppage analysis* gives the green-light. As $\mathbf{S}_k$ is selected from $\mathbf{D}_k$, the set of remaining data objects yet to be labelled is denoted as $\mathbf{D}_{k+1}$. Using the trained model, the process labels all the data objects in $\mathbf{D}_{k+1}$, resulting in a labelled dataset $\binom{D_{k+1}}{B_{k+1}}$.
- The process of *quality assurance* is necessary for any application that demands a very high quality of data reconstruction. It is an interactive process for a human user to inspect the machine-labelled data objects in $\binom{D_{k+1}}{B_{k+1}}$ and correct all errors found. This results in a checked and corrected dataset $\binom{D_{k+1}}{B'_{k+1}}$.
- The second process of *adding to labelled dataset* combines $\binom{D_{k+1}}{B'_{k+1}}$ and $\bigcup_0^k \binom{S_i}{A'_i}$ to form the final set of labelled data objects, which is denoted as $\binom{D_0}{L_0}$.

Similar to *data pre-processing*, *data post-processing* is an optional process for transforming the labelled data objects $\binom{D_0}{L_0}$ to those in an intended data structure and format. In some applications, an MI3 pipeline may be followed by another MI3 pipeline. We will see examples of *data pre-processing*, *data post-processing*, and multiple pipelines in the following sections.

4 REALIZING MI3 PIPELINES IN PRACTICE

In this section, we use the process of designing a software system for reconstructing data from discrete bar charts to showcase how to design an interactive classification process following the generic MI3 pipeline in practice. The Cholera Map shown in Fig. 1 typifies such a visual representation. The data reconstruction system consists of three pipelines, all of which were designed and implemented following the generic MI3 pipeline. They are pipelines for (i) detecting objects (e.g., detecting blocks in the Cholera Map), (ii) grouping components (e.g., detecting bars in the Cholera Map), and (iii) determining key positions (e.g., detecting roots in the Cholera Map).

4.1 Transforming a Decision Problem to a Classification Problem

In order to benefit from the generic MI3 approach, one needs to transform various decision problems such as detecting objects, grouping components, and determining key positions to classification problems. As shown in Fig. 3, we can use a pre-processing step to generate a list of candidates representing potentially correct decisions. These candidates then become the inputs of a classification pipeline, which categorizes these candidates into different label classes.

For example, when the first MI3 pipeline is activated for detecting blocks in the Cholera Map, the pre-processing step may construct a list of candidates, each of which is a group of connected pixels. The role of the interactive classification pipeline is then to label each candidate as a meaningful "block" representing a victim, or a "non-block". All candidates labelled as "block" are then passed to the next pipeline for grouping these blocks into bars.

When the second MI3 pipeline is activated for grouping the detected blocks into bars, the pre-processing step may generate a list of pairwise relations, each indicating that a pair of blocks may potentially belong to the same bar. These candidate relations are then fed into the interactive classification pipeline for labelling each candidate as "yes" for belonging to the same bar, and "no" otherwise. In this case, a post-processing step is necessary for transforming the results of the classification problem to the results of the original grouping problem. This can be achieved by using a simple algorithm for converting a list of detected blocks and a list of confirmed relations to a list of bars, each of which consists of a group of blocks linked with the confirmed relations. This post-processing step can also count the blocks in each bar, resulting in the data value $\#victims$ for each bar.

When the third MI3 pipeline is activated for determining the root positions of the bars, the pre-processing step may generate a set of potential positions for each bar, such as the center of the bar and that of each block in the bar, the middle point of each edge of the bar and its blocks, the corners of the bar and its blocks, and so on. The classification pipeline then labels the candidate positions to be "root" and "non-root", while ensuring exactly one root per bar. A post-processing step then converts the local root position relative to the bar to a global position (x, y) relative to the original visualization image. In combination with the data value $\#victims$ for each bar, the system generates a list of tuples in the form of $(x, y, \#victims)$.

If the user requires the actual geographical location (e.g., longitude and latitude, or other geodetic systems) for each tuple, this can be done trivially using an image-specific coordinate transformation as a post-processing step.

4.2 Variations of MI3 Pipelines

The design and implementation an MI3 pipeline in practice depends on many factors, such as the technical availability of various algorithmic components, the knowledge and skills of the developers, the cost and time constraint of the software life-cycle, and many other application-specific requirements. In general, one can consider the development as an agile software engineering approach, with a gradual introduction of better algorithms, techniques, or user interfaces for different processes in Fig. 3. One may consider some typical variations of the processes in the interactive classification part of MI3 pipelines:

- Variations of *algorithmic sampling* — Perhaps the most naive sampling method, which is denoted as **NS**, is to fetch data objects to be labelled simply according to their order in the input dataset $\mathbf{D}_k$. More commonly used in ML, *random sampling* (**RS**) is considered as the baseline sampling method, which selects data objects for labelling in a stochastic order. The approach of *algorithmic sampling* (**AS**) selects data objects according to a predefined metric that predicts the potential benefit of labelling a data object to the learning process. Such

a metric may assess the potential benefit according to the uncertainty of the label [5], the diversity of the samples [47], clustering information [29, 47], and the expected performance of the classifier [12]. In this work, we used two methods including Lewis and Catlett's uncertainty-based sampling [22] and Xu et al.'s sampling method according to weighted uncertainty, cluster density, and sample diversity score [47].

- Variations of *algorithmic default labelling* — When data objects are presented in a user interface for dynamic labelling, the system may assign a default label to each data object. If the system can predict some labels correctly, this can reduce the number of interactions that a human user has to perform in labelling the data objects. A most naive approach is *no default* labelling (**ND**). An alternative approach is *random default* labelling (**RD**), which assigns a data object to one of the label classes in a stochastic manner. The approach of *algorithmic default* labelling (**AD**) uses a model to predict the label of a data object. For example, one may use the interim model during an ML process to label a data object. Alternatively, one may use a label propagation algorithm (e.g., self-training [48], co-training [3], graph-based label propagation [50, 52]) to predict the label of a data object according to those data objects that have already been labelled. In this work, we have provided our MI3 pipelines with the mechanisms for using the interim model (**AD (Interim)**) as well as a graph-based label propagation algorithm (**AD (Graph)**) [50].
- Variations of *interactive labelling* — The approach of active learning represents a small portion among all research papers published on ML. One can naturally consider that the baseline approach in ML is to pre-label all data objects in the training and testing datasets, without any interactive labelling during an ML process. This baseline approach is denoted as (**PL**). Since the MI3 approach is designed for active learning, the process of interactive labelling (**IL**) is expected to be present in almost all MI3 pipelines. Different designs of the user interface (UI) for interactive labelling will impact on an MI3 pipeline differently. It is desirable to explore many design options. In this work, we focused on the use of the simplest interaction modality of button clicking to minimize the time and cognitive load per interaction.
- Variations of *training classifier* — We consider that the baseline is without any automatic and semi-automatic ML, and all data objects will be labelled manually. We denote this *brute force* approach as **BF**, while denoting the full automatic and semi-automatic ML approaches as **ML**. In practice, there are many design options for an ML pipeline. The MI3 approach does not impose any restriction. For example, MI3 pipelines can be instantiated with different ML frameworks for classification, e.g., variations of neural networks, decision tree and random forest, support vector machines, Bayesian networks. In this work, we have used decision tree, and the transductive model learned in label propagation [50] as two options for the process *training classifier*.

In Section 5, we will provide further technical details of the algorithms implemented in a software prototype built based on the MI3 approach, and in Section 6, we will compare how different combinations of the above variations may impact on the performance of the implemented MI3 pipelines.

5 DATA RECONSTRUCTION FROM CHART IMAGES

In order to validate the feasibility of the MI3 approach, and to study and evaluate different design options in realizing MI3 pipelines, we have developed a prototype software system for reconstructing data from a family of visual representations, which are composed of a collection of spatially distributed similar data objects. Each data object depicts a data tuple $(x, y, \#value)$. The spatial location (x, y) may be geographically meaningful and related to a background map or image. The

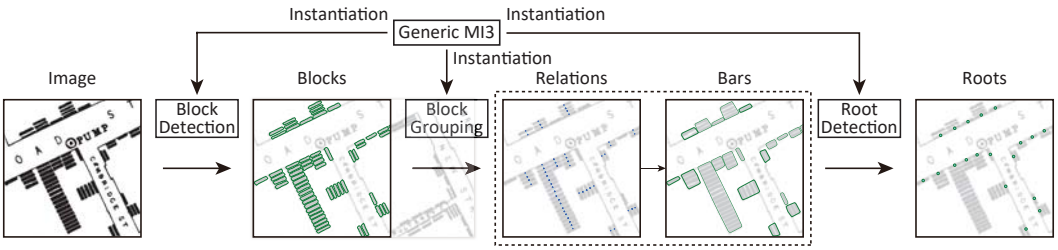


Fig. 4. A data reconstruction procedure for spatial data visualization with three subroutines: block detection, block grouping, and root detection. Each of the three subtasks is an interactive classification task, and is solved with an instantiation of the generic MI3 pipeline. **(1) Block Detection:** Given an input image, we use color connected components detection as a preprocessing to convert block detection into a classification task for connected components. **(2) Block Grouping:** With the detected blocks, we build a graph of block relations and classify the existence of relations to determine which pairs of blocks should be grouped in the same bar. The confirmed relations are used to compile the detection of bars. **(3) Root Detection:** For each bar, we generate candidate root points. Therefore, the task of determining the key point is transformed to candidate root classification.

variable *#value* may be encoded as the size of the data object, the number of components inside the data object, or other attributes. The Cholera Map shown in Fig. 1 exemplifies such a visual representation. Fig. 4 shows the procedure of data reconstruction from the discrete bar chart. We decompose the data reconstruction task into three subtasks: block detection, block grouping, and root detection, and discuss the details in the following subsections.

5.1 Block Detection Preprocessing

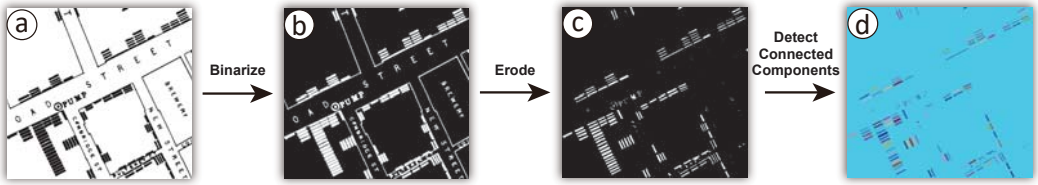


Fig. 5. Candidate block detection process on part of the Cholera Map [42]. Each connected component (denoted with different colors in (d)) is a candidate block. (a) Input part of the Cholera Map. (b) Binarize the image to reduce color variation. (c) Erode the image to reduce conglutination. (d) Color connected component detection.

Following the generic MI3 pipeline, we need to transform the block detection problem into a classification problem. For this transformation, we need to first generate candidate data objects, in this case colour connected components, to be classified, and then provide feature representation for each data object to be able to learn the interim classifier.

Candidate Data Object Detection: We design a preprocessing algorithm, as illustrated in Fig. 5. With the observation that blocks differ from the background in colour, we let colour connected components serve as candidate blocks. Colour connected component detection generates a large set of components to be classified, most of which are not blocks. It calls for MI3 strategies to reduce the interaction cost in this classification task.

More specifically, the candidate block detection procedure is as follows:

(1) Binarize the image with Otsu thresholding to eliminate minor color differences (Fig. 5(b)).
(2) Erode the binary image with 3×3 cross kernel for two iterations to reduce the conglutination of elements (Fig. 5(c)).
(3) Detect color connected components (Fig. 5(d)).
(4) Dilate the each connected component separately using the cross kernel for two iterations to compensate the size shrink during erosion.

Feature Computation: For each candidate block, we compute two sets of features: 13 appearance features and 14 neighbourhood features, which in total sums to 27 features. Appearance features capture the visual appearance of the candidate block. Neighbourhood features capture the image features of the candidate block's neighbourhood.

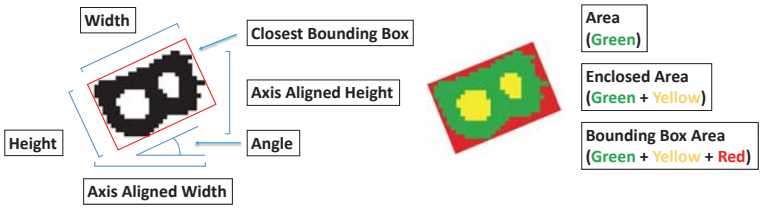


Fig. 6. Measurements related to appearance features of candidate blocks. The left shows the size features and the angle. The right shows area definitions involved in the computation of shape features.

We compute size features, shape features, and colour features as appearance features to capture the appearance of a candidate block. Size features include *width*, *height*, *axis aligned width*, *axis aligned height*, and *area*, which are all measured in pixels. Shape features include $solidity = \frac{area}{enclosed\ area}$, $convexity = \frac{area}{convex\ hull\ area}$, $extent = \frac{area}{bounding\ box\ area}$, $aspect\ ratio = \frac{width}{height}$, and *angle*. Fig. 6 shows visual attributes used as size features and attributes used for computing shape features. Color features include r, g, and b computed by averaging the color of all the pixels. In total, 13 appearance features are computed.

The 14 neighbourhood features are mainly based on the distribution of pixel colours in the neighbourhood of the candidate block. Multiple neighbourhood definitions are considered in the computation. We describe the details of neighbourhood features in Appendix A.

5.2 Block Grouping Preprocessing

To aggregate the number of victims according to location, we need to group neighbouring blocks into bars. We reduce the grouping problem to determining which pairs of blocks are neighbours. If we regard each block as a node and each possible neighbouring relationship as an edge, the grouping problem is transformed to graph edge classification, which fits the MI3 pipeline.

Candidate Data Object Detection: Each pair of blocks is potentially a pair of neighbouring blocks. Given n blocks, there are in total $n^2 - n$ candidate relations. In practice, the quadratically exploding number of data objects pose a computation challenge for algorithms. Therefore, we design two heuristic filtering rules as follows:

(1) There is no relation between a pair of blocks when they are not mutually the top four closest to the other block among all the blocks.
(2) There is no relation between a pair of blocks when the distance between two blocks is larger than the length of the longest axis of all blocks.

Feature Computation: For each candidate neighbouring relation, we compute *position distance*, *color distance*, and *overall distance* as the features. For a pair of blocks, the *position distance* is defined

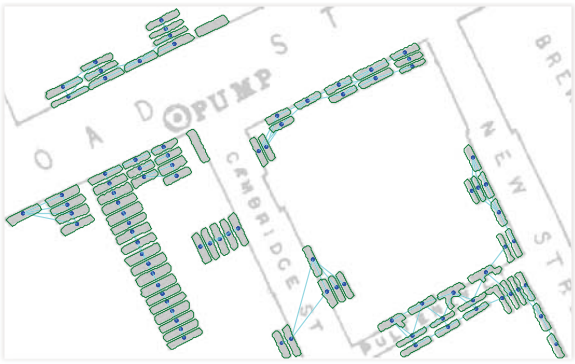


Fig. 7. Candidate neighbouring relation detection result on part of the Cholera Map [42]. The blocks are denoted with green contours. The blue dots show the center of each block. The detected candidate neighbouring relations are shown as cyan lines linking the center of the two blocks involved.

as the Euclidean distance between the position of the two blocks (x_1, y_1), (x_2, y_2) in pixels. For a pair of blocks, the *color distance* is defined as the Euclidean distance between the two blocks in the (r, g, b) space. The *overall distance* is defined as the root mean square of the *position distance* and *color distance*.

5.3 Root Detection Preprocessing

Fig. 8. Candidate root detection result on part of the Cholera Map [42]. The bars are denoted with green contours. Each candidate root is denoted with a cyan dot.

To reconstruct the position information of a data object in the spatial visualization, we need to detect the representative point of the data object. To align this problem to the MI3 pipeline, we need to generate a set of candidate roots to be classified, where a "root" refers to the key point in the data object that corresponds to this position.

Candidate Data Object Detection: We simplify the root detection problem as finding the centre of a block in a bar that represents the position. The root of a bar is an extreme point of the bar that is typically closest to boundary lines in the image. Therefore, we take the centre of the topmost and the bottommost blocks of the bar to be candidate roots. Fig. 8 shows the detected candidate roots on part of the Cholera Map.

Feature Computation: For each candidate root, we compute in total 9 neighbourhood features. We observe that roots are typically close to boundary lines in the image, and that roots of multiple bars may be aligned. Therefore, we reuse some of the neighbourhood features of block as root features, including *horizontal neighbour distance*, *surrounding neighbour number*, *surrounding foreground rate*, *horizontal foreground rate*, *host area* and *host solidity*. Besides, we compute a boolean feature *single* to denote whether the block containing the root is the only block in a bar. When a root is *single*, it is always a true detection. We also compute two additional neighbour distribution features: *distance to alignment line* and *distance to skeleton*. The definition of the two features are described in Appendix B.

5.4 Interactive Classification in Action

In the previous subsections, we illustrate how to transform subproblems of data reconstruction into interactive classification problems that MI3 concerns. In the following, we introduce how the two major algorithmic components of MI3, algorithmic sampling and algorithmic default labelling, are instantiated in our implementation. We also introduce our prototype implementation of an MI3-based data reconstruction system, which shows how the major interactive component, interactive labelling, is instantiated. The implementations of algorithmic sampling, algorithmic default labelling, and interactive labelling are reused for all the three subproblems.

5.4.1 Algorithmic Sampling. Through the *algorithmic sampling* process, we determines the data object samples $S_k = (x_{k_1}, ..., x_{k_l})$, where l is the number of data objects shown in the interface at a time. The samples will then be default labelled and presented to the user for labelling. The input to *algorithmic sampling* is unlabelled data objects $D_k = (x_1, x_2, ..., x_n)$ as show in Fig. 3. We instantiate *algorithmic sampling* (AS) with two optional sampling methods: *entropy-based sampling* and *density, diversity, and entropy-based sampling* (abbreviated as **AS (ES)** and **AS (DDES)** respectively) based on two sampling methods in the literature [22][47].

With *entropy-based sampling*, the metric of sample priority is defined as the uncertainty of the label [22]. The higher the uncertainty of the label, the more the label information can inform the classifier learning process, and thus resulting in less interaction cost. The uncertainty of an unlabelled data object x_i's label y_i is quantified by entropy as $H(y_i|x_i) = -\sum_j p(y_i = j|x_i)log(p(y_i = j|x_i))$. When l instances are to be presented to the user, the top l unlabelled instances that maximize the entropy are sampled. In reality, we do not have access to the true posteriori $p(y|x)$ used in entropy calculation, and therefore it has to be approximated. When the interim model trained in the *training classifier* stage is a probability-based classifier, we directly use the posteriori estimated by the classifier as the true posteriori. When the interim model is not a probability-based classifier, e.g., decision tree, we use Zhou et al.'s algorithm [50] to train an accompanying probability-based classifier that serves as a posteriori estimator.

Entropy-based sampling only aims to maximize the uncertainty of the samples, while *density, diversity, and entropy-based sampling* sets two additional goals for optimization [47]. It requires that in each iteration, the sampled instances should be distant from each other (diverse), and the sampled instances should be representative in the feature space (dense). To this aim, the scoring function of each data object's priority is defined as $score(x_i) = (1 - \alpha - \beta)H(y_i|x_i) + \alpha \sum_{j=1}^{n} \frac{1}{1+d(x_i,x_j)} + \beta \sum_{j \in S_k}(1 - \frac{1}{1+d(x_i,x_j)})$. It maximizes a weighted sum of label entropy, average inverse distance to other unlabelled data objects, and distance to the data objects already sampled in this batch. The data object with the highest score is added to the sampled set S_k and sampling is conducted iteratively until $|S_k| = l$. In our implementation, we set $\alpha = \beta = \frac{1}{3}$, and the distance function d to measure Euclidean distance.

5.4.2 Algorithmic Default Labelling. Through the *algorithmic default labelling* process, default labels are assigned to sampled data objects, which saves the user's effort for labelling. We instantiate *algorithmic default labelling* with two implementations, including *interim model-based default* (abbreviated as **AD (Interim)**) and *graph-based default* (abbreviated as **AD (Graph)**).

With *interim model-based default*, an interim classification model is trained and updated each time a batch of newly labelled data points are added to the labelled set. The interim model predicts the default labels for the instances to be presented to the user in the current iteration.

Algorithm 1 Transition Matrix Computation

Require: instances $X = \{x_1, ..., x_n\}$
Ensure: transition matrix $K_{n\times n}$
1: $W \leftarrow [I(i \neq j)exp(-\frac{||x_i - x_j||^2}{2\sigma^2})]_{n\times n}$
2: $D \leftarrow [I(i = j)\sum_{j=1}^{n} W_{ij}]_{n\times n}$
3: $S \leftarrow D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$
4: $K \leftarrow (I - \alpha S)^{-1}$
5: **return** K

The idea of *graph-based default* is that data points close to each other in the feature space tend to have the same label, and therefore labelled data objects can propagate its label to unlabelled data objects. The closer a pair of instances are, the more likely the label can propagate between them.

Quantitatively, matrix S in the pseudo code 1 denotes the probability of propagation. $S_{i,j}$ is the probability that the label of data object i can propagate to data object j. This probability is based on the normalized link weight distance between different data objects i and j defined as $W_{ij} = \frac{1}{Z_i} exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$, where $Z_i = \sum_{j=1}^{n} W_{ij}$ is the sum of unnormalized link weight between data object i and all the other data objects. This propagation process is run for multiple iterations until convergence. Transition matrix K denotes the transformation that transforms the initial label distribution to the convergent state.

Algorithm 2 Label Propagation

Require: transition matrix $K_{n\times n}$, labels of newly labelled instances $y_{q_1}, ..., y_{q_l}$, the range of class labels $\{1, 2, ..., c\}$, label distribution in the last turn $F^t_{n\times c}$ (initialized as $F^0 = 0_{n\times c}$ in the first turn)
Ensure: updated label distribution $F^{t+1}_{n\times c}$
1: **for** $q \in \{q_1, ..., q_l\}$ **do**
2: $\Delta F \leftarrow [I(j = y_q)K_{i,q}]_{n\times 2}$
3: $F^{t+1} \leftarrow F^t + \Delta F$
4: **end for**
5: **return** F^{t+1}

Each time the user annotates a batch of l data objects, we propagate the labels using the transition matrix K as shown in pseudocode 2 to incrementally update the default labels. Using the propagated label distribution F, we compute the default labels by maximal likelihood.

Graph-based default can be computed very efficiently in the iterative labelling sessions, because the incremental update algorithm 2 runs in $O(n)$ time. The *graph-based default* is based on Zhou et al.'s label propagation algorithm [50]. The original algorithm reported in the literature takes $O(n^3)$ time to run, and does not consider how to conduct incremental update, which is not efficient enough to support real-time interaction.

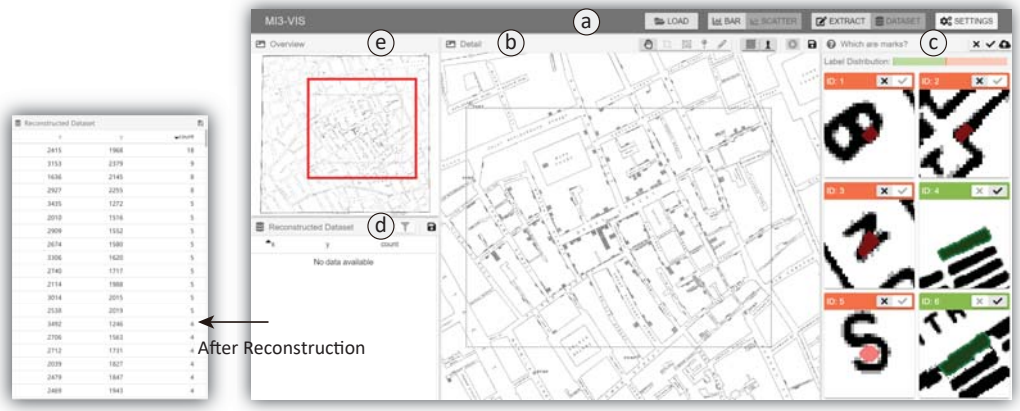


Fig. 9. The prototype user interface for data reconstruction. The interface consists of five parts: (a) Control Toolbar: the user can upload the image, select the chart type, switch interface layout, and change the algorithm parameters. (b) Image View: the user can navigate by pan and zoom, clip the image, set the scale of the image, and manually create visual objects if needed. The reconstructed dataset is re-visualized in the Image View and can by interacted by spatial cross-filtering with lasso selection. (c) Annotation Panel: the user can annotate the label for data objects. (f) Dataset View: the Dataset View shows the reconstructed dataset after the reconstruction process is finished. (e) Image Overview: the overview highlights the currently zoomed region of the Image View.

5.4.3 The Interactive Labelling Interface. In the interface, the user first uploads a visualization image, specify the type of chart to change the preprocessing method in the Control Toolbar (a). Then, the user clicks the start button on the header of the Image View (b) to start the reconstruction. If needed, the user can switch between different implementations of the MI3 algorithmic components with the setting popup menu.

After the image is uploaded, the user can click the start button to start the data reconstruction. If needed, the user may clip the image to focus the algorithm on a subpart. Once the reconstruction starts, the system goes through the preprocessing, algorithmic sampling, and default labelling. The sampled objects in each iteration are displayed in the Annotation Panel (c). After the sampling, the Image View (b) automatically zooms to the area that the sampled objects lie in. The currently zoomed area is highlighted in the Image Overview (e). If needed, the user can pan and zoom the image in the Image View.

In the Annotation Panel, the user can click on the thumbnail image to flip the label of mislabelled instances or press the corresponding number key to label. The user can also click the object in the Image View to change the label. When the label of the object is hard to determine from the thumbnail image, and the user's mouse has been hovering on the thumbnail for more than 500ms, the system automatically zooms the Image View to the corresponding object to help the user make the judgement.

Once the sampled data objects are labelled, the user can click the confirm button in the Annotation Panel or press the "enter" key to add the data points to the labelled set. Then, the labelling process goes iteratively until the termination criterion is met.

After all the interactive classification pipelines for data reconstruction are gone through, the Dataset View (d) shows the reconstructed dataset. The user can export the reconstructed dataset in

JSON format. The reconstructed dataset will also be superimposed in the input visualization image. This reconstructed visualization can be exported in SVG format.

6 EVALUATION

In the previous section, we introduce multiple optional implementations of MI3 components which generate multiple instantiations of the MI3 pipeline. In this section, we evaluate the performance of different instantiations. We consider six major variations of solutions to the classification task:

- **BF** — The **brute force** approach where the user needs to use pen and ruler to measure and record all the data points without the system's intelligent support.
- **ML + x%-PL** — A conventional **ML-based** system where the user **pre-labels x%** of the training set. Then, a classification model is trained with the training set, and assigns default labels for all the remaining data points.
- **RS + RD + k-DL** — An instantiation of the MI3 pipeline that exploits **random sampling**, and **random default labelling**. In each interactive labelling session, **k** data points are random sampled, assigned a default label randomly, and presented to the user. This provides a reference benchmark for evaluating the algorithmic sampling and algorithmic default labelling technique.
- **AS + RD + k-DL** — Similar to the above, except that **algorithmic sampling** is used in place of random sampling.
- **RS + AD + k-DL** — An instantiation of the MI3 pipeline that exploits **random sampling**, and an **algorithmic default labelling** strategy. In each interactive labelling session, **k** data objects are randomly sampled and then default labelled according to the strategy. So the number of interactions depends on the error rate of the AD technique as well as the number k of data objects presented in each iteration.
- **AS + AD + k-DL** — Similar to the above, except that **algorithmic sampling** is used in place of random sampling.

Although it is useful to run user studies, the numerous system design options would require a large number of subjects and trials in the study, which makes it hard to carry out in practice. To evaluate different options of instantiating the MI3 pipeline, we adopt Zhang et al.'s simulation-based evaluation method [49]. Precisely, the number of interactions needed to finish the data reconstruction task serves as the evaluation metric, and is estimated by the simulation. The simulation-based evaluation method makes it possible to gather a large number of repeated trials, and the result does not suffer from the variance of human subjects.

6.1 Dataset

We use John Snow's Cholera Map [42] as the dataset for evaluation. In the Cholera Map, there are 579 rectangular blocks that form 321 bars. As illustrated in the previous sections, to reconstruct data from this image, the user needs to carry out three interactive classification tasks, i.e., classify candidate blocks, relations, and roots as true or false detections.

For the block detection problem, there are 4416 candidate marks to be classified detected by the preprocessing algorithm based on colour connected component detection. Among the 4416 candidates, 533 are positive (true detections), and 3883 are negative (false detections). Note that there are 46 marks missed by the preprocessing algorithm. Therefore, for all the interactive classification pipelines, the user needs to spare additional efforts to annotate these 46 marks. For the relation classification problem, there are 868 candidates relations to be labelled where 258 are positive, and 610 are negative. For the root classification problem, there are 450 candidate roots to be labelled where 321 are positive, and 129 are negative.

6.2 Experiment Design

Compared Methods In the simulation, we compare the aforementioned six solutions to interactive classification tasks: (i) **BF**, (ii) **ML + 20%-PL + NeD + 6-DL**, (iii) **RS + RD + 6-DL**, (iv) **AS + RD + 6-DL**, (v) **RS + AD + 6-DL**, and (vi) **AS + AD + 6-DL**. Note that in the block detection dataset, most of the data objects are false detections. For the **ML + x%-PL** solution using conventional ML-based system with $x\%$ data points pre-labelled, we can enhance it by always pre-labelling data points to have negative default (NeD). Therefore, in the experiment, we substitute the conventional ML solution with **ML + x%-PL + NeD + k-DL**, which is a more competitive benchmark. Specifically, we set the pre-label rate $x\%$ to be 20%. For all the solutions, we fix the number of data objects k presented in each interactive labelling session to be 6. For conciseness of the result, we fix entropy-based sampling to be the instantiation of the active sampling (AS (ES)), and interim decision tree model to be the model for algorithmic default labelling (AD (Interim)).

Evaluation Metric In the experiment, for all the three subtasks of data reconstruction, we measure the interaction cost for each method to achieve 100% accuracy for the interactive classification task. For example, in the block detection problem, there are 579 true blocks. The preprocessing pipeline detects 4416 data points to be classified where 533 are true detections. For the **BF** solution using pen-and-ruler or its digital equivalent, it would take a minimal 579 interactions to measure and record all the data objects. For the conventional ML solution with **ML + 20%-PL + NeD + 6-DL**, it would require $883 \approx 4416 \times 20\%$ data points to be labelled in the pre-labelling session. Approximately 12% of the data points are true detections. Therefore, the negative default strategy (NeD) would produce incorrect default labels for 12% of the 883 data points. It needs $108 \approx 883 \times 12\%$ interactions for correction of incorrect default labels, and $148 \approx 883/6$ interactions for clicking the confirm button to register the corrections. An additional 46 interactions is needed to correct missing blocks in the preprocessing. Our initial testing result shows that with 883 data points labelled, the model achieved 98.837% accuracy for the remaining $3533 = 4416 - 883$ data points. Therefore, $41 \approx (1 - 98.837\%) \times 3533$ interactions is needed to correct the misclassification. In total, the number of interactions to achieve 100% accuracy for **ML + 20%-PL + NeD + 6-DL** is $343 = 108 + 148 + 46 + 41$.

We can estimate the interaction cost for other solutions to interactive classification similarly. Note that for the four instantiations of MI3, the interaction cost is dependent on the number of interactive labelling sessions conducted. The more sessions, the more data points labelled, and therefore the more interactions needed for default label correction. Meanwhile, the more data points labelled, the higher the accuracy of the model, which reduce the interaction cost to quality assure and correct the labels for the remaining data points. In the experiment, we consider these dynamics and represent the overall interaction cost as a function of the number of interactions in interactive labelling sessions. For the block detection problem, we run 75 repeated measures and average the results. For the relation and root detection problem, we run 90 repeated measures and average the results.

6.3 Results and Analysis

Fig. 10 shows the experiment results in block detection with the six methods. In the following, we discuss patterns observed in the experiment result. For conciseness, we refer to the total interaction cost to finish the interactive classification task (i.e., achieve 100% accuracy) to be the "overall cost", the interaction cost in the iterative labelling sessions to be the "active learning cost", and the interaction cost in the quality assurance session to correct mislabelled data objects to be "quality assurance cost".

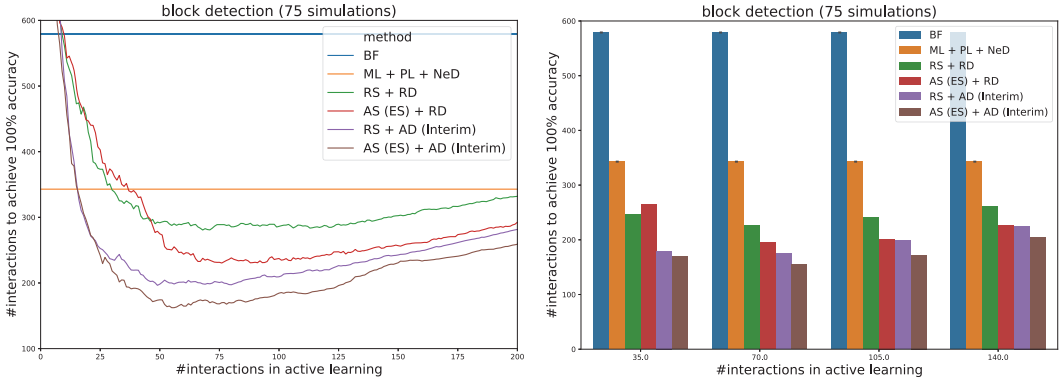


Fig. 10. Comparison of the 6 interactive classification methods on the block detection problem. The curves on the left show the change of interaction cost to achieve 100% accuracy with regard to the interaction cost on active learning. The bar chart compares the performance of the methods at 35, 70, 105, and 140 active learning interactions. The correspondence between and curve and bar with the methods is: blue for the brute force approach (**BF**); orange for the conventional ML-based approach with 20% data objects pre-labelled with negative default labelling (**ML** + **20%-PL** + **NeD** + **6-DL**); green for random sampling with random default labelling (**RS** + **RD** + **6-DL**); red for algorithmic sampling with random default labelling (**AS** + **RD** + **6-DL**); purple for random sampling with algorithmic default labelling (**RS** + **AD** + **6-DL**); brown for algorithmic sampling with algorithmic default labelling (**AS** + **AD** + **6-DL**). All the methods except **BF** shows 6 data objects in each labelling session, and therefore, we drop the **6-DL** in the method names in the legend.

1. Performance of MI3 pipeline instances follows a U-shaped curve. The overall cost of **RS** + **RD**, **AS** + **RD**, **RS** + **AD**, and **AS** + **AD** follows the same trend of first decrease and then increase with the active learning cost. Such a pattern emerges because the overall cost is comprised of the active learning cost and quality assurance cost. The quality assurance cost typically experiences a rapid decrease during the first few interactive labelling sessions, because the accuracy of classifier typically grows fast at the beginning with the training set size. With more interactions spent on active learning, the rate that quality assurance cost decreases gets slower as the interim classification model's performance reaches a plateau. At some point, the increased contribution of active learning cost to the overall cost would overcome the decrease of the quality assurance cost, making the overall cost increase again.

Because of this pattern of the overall cost, for each labelling method, there is an optimal point of the labelling interaction that minimizes the overall cost. For example, for **RS** + **RD** (green), the optimal point is when 68 interactions are spent on active learning. The lower the overall cost at the optimal point, the better. The overall cost at the optimal point depicts the performance of the method when the iterative labelling sessions terminates at a suitable time point as determined by the termination criterion. Therefore, we compare different methods mainly by the lowest point of the curve in the following.

2. MI3 pipeline instances outperform the brute force approach and the conventional machine learning pipeline. The optimal overall cost of **RS** + **RD**, **AS** + **RD**, **RS** + **AD**, and **AS** + **AD** are smaller than the overall cost of **BF** and **ML** + **PL** + **NeD**. For the block detection task, it takes **BF** 579 interactions to record all the blocks. It takes **ML** + **PL** + **NeD** 343 interactions to detect all the blocks with 100% accuracy. By comparison, the other four MI3 instances have optimal overall cost between [150, 300] interactions.

3. Algorithmic sampling (AS) reduces interaction cost. For MI3 instances with the same default labelling strategy, the instance with an algorithmic sampling strategy outperforms the instance with random sampling. Specifically, the overall cost of **RS + RD** (green) reaches its minimal, 280 interactions, with 68 active learning interactions and then gradually increases. When the sampling method is fixed as **RD**, after incorporating algorithmic sampling, **RS + RD** transforms to **AS + RD** (red). The optimal overall cost reduces to 230 interactions, which is achieved at 76 active learning interactions. Similar differences are observed between **RS + AD** (purple) and **AS + AD** (brown). We interpret the smaller overall interaction cost as algorithmic sampling manages to continuously explore informative samples for the algorithmic to learn, which sustainably decrease the number of errors and reduce the interaction cost to correct.

4. Algorithmic default labelling (AD) reduces interaction cost. For MI3 instances with the same sampling strategy, the instance with an algorithmic default labelling strategy outperforms the instance with random default labelling. With algorithmic default labelling, a lower overall cost can be achieved with the same number of interactions compared with random default labelling. For example, the overall cost of **RS + RD** (green) at the optimal point is 280 interactions, and is larger than that of **RS + AD** (purple), which is 196 interactions. The reason is that algorithmic default labelling is typically more accurate than random default. Therefore, algorithmic default labelling enables more data points to be labelled with the same active learning cost. Thus, the quality assurance cost of algorithmic default labelling is less than that of random default labelling when the active learning cost is the same.

In summary, instantiations of the MI3 pipeline outperforms the brute force approach (**BF**) and the conventional machine learning pipeline (**ML + 20%-PL + NeD + 6-DL**). Moreover, among different instantiations of the MI3 pipeline, we find that for the sampling component, pipelines using algorithmic sampling (**AS**) performs better than those using random sampling (**RS**). For the default labelling component, pipelines using algorithmic default labelling (**AD**) is better than those using random default labelling (**RD**). Among the implemented MI3 instances, the one with the best performance, **AS + AD**, requires a minimal 162 interactions to accomplish the classification task with 100% accuracy, which is less than one-third the cost of the brute force approach, and less than half the cost of the conventional machine learning pipeline.

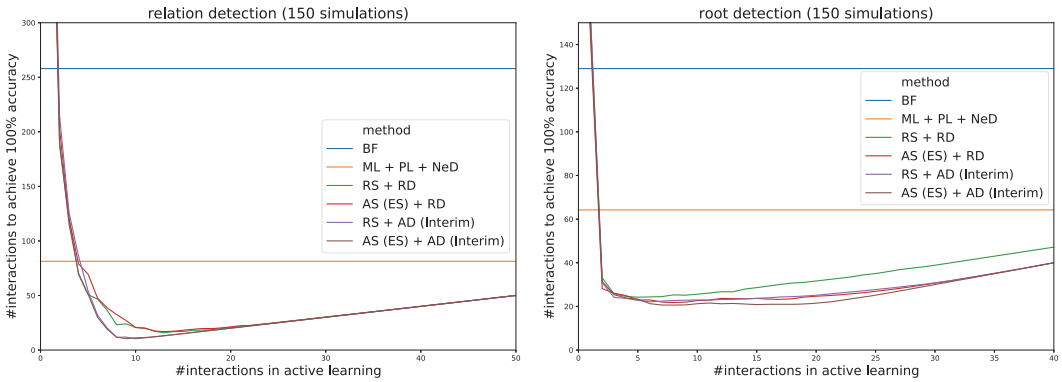


Fig. 11. Comparison of the 6 methods on the relation detection and root detection. The curve shows the change of overall interaction cost with regard to the labelling interaction cost.

Aside from the block detection problem, we also evaluate the six pipelines on relation detection (block grouping) and root detection. The experiment results are shown in Fig. 11. We observe that

similar to the result of block detection, MI3 pipeline instances outperform the brute force approach and the conventional machine learning pipeline. However, for these two subtasks, algorithmic sampling (**AS**) and algorithmic default labelling (**AD**) bring insignificant benefits. We interpret it as the result that the two subproblems are much simpler than the block detection problem, as both block detection and position decoding can be accomplished with less than 30 interactions.

7 DISCUSSION AND CONCLUSION

We propose MI3, a generic pipeline for interactive classification tasks that reduces user's interaction effort. MI3 is especially useful for interactive classification applications where there is not sufficient data to learn a precise classification model. In this work, we introduce two components of MI3 for interaction saving: algorithmic sampling and algorithmic default labelling. We demonstrate how to instantiate the generic MI3 pipeline in practice. Specifically, we demonstrate that the scenario of data reconstruction from historical visualization, which seemingly is not a classification problem, can be decomposed and transformed into three interactive classification subtasks. All the three subtasks fit into the MI3 pipeline and can make use of MI3 components for interaction saving.

Based on the decomposition, we develop a prototype software system for data reconstruction from spatial data visualizations which can be regarded as an instantiation of MI3's interactive labelling component. To evaluate the usefulness of our instantiation of MI3 in the data reconstruction application, we use the simulation-based approach and compare the number of interactions needed for different solutions to accomplish the interactive classification tasks in data reconstruction. The simulation result shows that MI3's algorithmic sampling and algorithmic default labelling components manage to reduce the required number of interactions. For specific subtasks of data reconstruction, an instantiation of the MI3 pipeline can save up to half of the interactions compared with a conventional ML pipeline.

There are a few aspects and limitation of this work that we aim to address in the future. In this work, we only inspect one specific application scenario, namely data reconstruction, to examine the usefulness of the generic MI3 pipeline, and we are working on exploring other application scenarios for MI3. For the algorithmic components of MI3, we aim to investigate more optional algorithm designs. For the interactive component of MI3, we only develop one interface as an instantiation, and we are working on exploring the design options of the interface. In the evaluation, we use a simplistic assumption that the interaction cost is proportional to the number of button clicks. The validity of this assumption can be investigated in future user studies. We also assume that with the termination criterion, the interactive labelling session can determinate at a point that approximately minimizes the overall interaction cost. In practice, a naive termination criterion, such as setting a fixed sample rate, may not be able to achieve this goal. It calls for better termination strategies.

REFERENCES

[1] Rabah A. Al-Zaidy and C. Lee Giles. 2017. A Machine Learning Approach for Semantic Structuring of Scientific Charts in Scholarly Documents. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. 4644–4649. http://aaai.org/ocs/index.php/IAAI/IAAI17/paper/view/14275

[2] Mykhaylo Andriluka, Jasper R. R. Uijlings, and Vittorio Ferrari. 2018. Fluid Annotation: A Human-Machine Collaboration Interface for Full Image Annotation. In *Proceedings of the 26th ACM international conference on Multimedia*. ACM, 1957–1966. https://doi.org/10.1145/3240508.3241916

[3] Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*. ACM Press, New York, New York, USA, 92–100. https://doi.org/10.1145/279943.279962

[4] Simon Bovet and Jean Bovet. 2006. GraphClick. http://www.arizona-software.ch/graphclick/

[5] Klaus Brinker. 2003. Incorporating Diversity in Active Learning with Support Vector Machines. In *Proceedings of the 20th International Conference on Machine Learning*. AAAI Press, Washington, DC, USA, 59–66. https://dl.acm.org/citation.cfm?id=3041846

[6] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. 2002. Cluster Kernels for Semi-supervised Learning. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, 601–608. http://dl.acm.org/citation.cfm?id=2968618.2968693

[7] Jingyu Cui, Fang Wen, Rong Xiao, Yuandong Tian, and Xiaoou Tang. 2007. EasyAlbum: An Interactive Photo Annotation System Based on Face Clustering and Re-ranking. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press, New York, New York, USA, 367–376. https://doi.org/10.1145/1240624.1240684

[8] Jerry Alan Fails and Dan R. Olsen. 2003. Interactive Machine Learning. In *Proceedings of the International Conference on Intelligent User Interfaces*. 39–45. https://doi.org/10.1145/604045.604056

[9] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. 2008. CueFlik: Interactive Concept Learning in Image Search. In *Proceeding of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, New York, New York, USA, 29–38. https://doi.org/10.1145/1357054.1357061

[10] Jinglun Gao, Yin Zhou, and Kenneth E Barner. 2012. View: Visual Information Extraction Widget for Improving Chart Images Accessibility. In *Proceedings of the IEEE International Conference on Image Processing*. IEEE, 2865–2868. https://doi.org/10.1109/ICIP.2012.6467497

[11] Arnd Gross, Sibylle Schirm, and Markus Scholz. 2014. Ycasd - a tool for capturing and scaling data from graphical representations. *BMC Bioinformatics* 15, 1 (2014), 219. https://doi.org/10.1186/1471-2105-15-219

[12] Yuhong Guo and Dale Schuurmans. 2007. Discriminative Batch Mode Active Learning. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Vancouver, British Columbia, Canada, 593–600. http://dl.acm.org/citation.cfm?id=2981562.2981637

[13] Zicheng Guo and Richard W Hall. 1989. Parallel Thinning with Two-subiteration Algorithms. *Commun. ACM* 32, 3 (1989), 359–373. https://doi.org/10.1145/62065.62074

[14] Thomas S. Huang, Charlie K. Dagli, Shyamsundar Rajaram, Edward Y. Chang, Michael I. Mandel, Graham E. Poliner, and Daniel P.W. Ellis. 2008. Active Learning for Interactive Multimedia Retrieval. *Proc. IEEE* 96, 4 (apr 2008), 648–667. https://doi.org/10.1109/JPROC.2008.916364

[15] Weihua Huang and Chew Lim Tan. 2007. A System for Understanding Imaged Infographics and Its Applications. In *Proceedings of the ACM Symposium on Document Engineering*. ACM, New York, NY, USA, 9–18. https://doi.org/10.1145/1284420.1284427

[16] Thorsten Joachims. 1999. Transductive Inference for Text Classification Using Support Vector Machines. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 200–209. http://dl.acm.org/citation.cfm?id=645528.657646

[17] Ajay J. Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. 2009. Multi-Class Active Learning for Image Classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2372–2379. https://doi.org/10.1109/CVPR.2009.5206627

[18] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. 2017. ChartSense: Interactive Data Extraction from Chart Images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 6706–6717. https://doi.org/10.1145/3025453.3025957

[19] Saurabh Kataria, William Browuer, Prasenjit Mitra, and C Lee Giles. 2008. Automatic Extraction of Data Points and Text Blocks from 2-dimensional Plots in Digital Documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 1169–1174. http://dl.acm.org/citation.cfm?id=1620163.1620254

[20] Kostiantyn Kucher, Carita Paradis, Magnus Sahlgren, and Andreas Kerren. 2017. Active Learning and Visual Analytics for Stance Classification with ALVA. *ACM Transactions on Interactive Intelligent Systems* 7, 3 (oct 2017), 1–31. https://doi.org/10.1145/3132169

[21] Christian Leistner, Amir Saffari, Jakob Santner, and Horst Bischof. 2009. Semi-Supervised Random Forests. In *Proceedings of the IEEE 12th International Conference on Computer Vision*. IEEE, 506–513. https://doi.org/10.1109/ICCV.2009.5459198

[22] David D Lewis and Jason Catlett. 1994. Heterogeneous Uncertainty Sampling for Supervised Learning. In *Proceedings of the International Conference on Machine Learning*, William W Cohen and Haym Hirsh (Eds.). Morgan Kaufmann, San Francisco (CA), 148–156. https://doi.org/10.1016/B978-1-55860-335-6.50026-X

[23] Dong Liu, Meng Wang, Xian-Sheng Hua, and Hong-Jiang Zhang. 2009. Smart Batch Tagging of Photo Albums. In *Proceedings of the 17th ACM International Conference on Multimedia*. ACM Press, New York, New York, USA, 809–812. https://doi.org/10.1145/1631272.1631420

[24] Ruizhe Liu, Weihua Huang, and Chew Lim Tan. 2007. Extraction of Vectorized Graphical Information from Scientific Chart Images. In *Proceedings of International Conference on Document Analysis and Recognition*, Vol. 1. IEEE, 521–525. https://doi.org/10.1109/ICDAR.2007.4378764

[25] Yiwen Luo, Wei Liu, Jianzhuang Liu, and Xiaoou Tang. 2008. MQSearch: Image Search by Multi-class Query. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 49–52. https://doi.org/10.1145/1357054.1357063

[26] Gonzalo Gabriel Méndez, Miguel A Nacenta, and Sebastien Vandenheste. 2016. iVoLVER: Interactive Visual Language for Visualization Extraction and Reconstruction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, San Jose, CA, USA, 4073–4085. https://doi.org/10.1145/2858036.2858435

[27] Charles Joseph Minard. 1845. Route from Dijon to Mulhouse.

[28] Charles Joseph Minard. 1869. Figurative Map of the successive losses in men of the French Army in the Russian campaign 1812-1813.

[29] Hieu T. Nguyen and Arnold Smeulders. 2004. Active Learning Using Pre-clustering. In *Proceedings of the International Conference on Machine learning*. ACM Press, New York, New York, USA, 79. https://doi.org/10.1145/1015330.1015349

[30] Florence Nightingale. 1858. *Mortality of the British Army: At Home, at Home and Abroad, and during the Russian War, as Compared with the Mortality of the Civil Population in England*. London: Printed by Harrison and Sons.

[31] Florence Nightingale. 1858. *Notes on Matters Affecting the Health, Efficiency, and Hospital Administration of the British Army*. London: Printed by Harrison.

[32] William Playfair. 1786. *Commercial and Political Atlas: Representing, by Copper-Plate Charts, the Progress of the Commerce, Revenues, Expenditure, and Debts of England, during the Whole of the Eighteenth Century*. London: Corry.

[33] William Playfair. 1801. *Statistical Breviary: Shewing, on a Principle Entirely New, the Resources of Every State and Kingdom in Europe*. Wallis, London.

[34] Jorge Poco and Jeffrey Heer. 2017. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Computer Graphics Forum* 36, 3 (jun 2017), 353–363. https://doi.org/10.1111/cgf.13193

[35] Jorge Poco, Angela Mayhua, and Jeffrey Heer. 2018. Extracting and Retargeting Color Mappings from Bitmap Images of Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (jan 2018), 637–646. https://doi.org/10.1109/TVCG.2017.2744320

[36] Dimitrios Rafailidis, Apostolos Axenopoulos, Jonas Etzold, Stavroula Manolopoulou, and Petros Daras. 2014. Content-based tag propagation and tensor factorization for personalized item recommendation based on social tagging. *ACM Transactions on Interactive Intelligent Systems* 3, 4 (jan 2014), 1–27. https://doi.org/10.1145/2487164

[37] Ankit Rohatgi. 2011. WebPlotDigitizer. https://automeris.io/WebPlotDigitizer

[38] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. 2008. LabelMe : A Database and Web-Based Tool for Image Annotation. *International Journal of Computer Vision* 77, 1-3 (2008), 157–173. https://doi.org/10.1007/s11263-007-0090-8

[39] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. 2011. ReVision: Automated Classification, Analysis and Redesign of Chart Images. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 393–402. https://doi.org/10.1145/2047196.2047247

[40] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi. 2016. FigureSeer: Parsing Result-Figures in Research Papers. In *Proceedings of the European Conference on Computer Vision*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 664–680. https://doi.org/10.1007/978-3-319-46478-7_41

[41] John Snow. 1854. Cholera Map.

[42] John Snow. 1855. *On the Mode of Communication of Cholera*. John Churchill.

[43] Jinhui Tang, Qiang Chen, Meng Wang, Shuicheng Yan, Tat-Seng Chua, and Ramesh Jain. 2013. Towards Optimizing Human Labeling for Interactive Image Tagging. *ACM Transactions on Multimedia Computing, Communications, and Applications* 9, 4 (aug 2013), 1–18. https://doi.org/10.1145/2501643.2501651

[44] Yuandong Tian, Wei Liu, Rong Xiao, Fang Wen, and Xiaoou Tang. 2007. A Face Annotation Framework with Partial Clustering and Interactive Labeling. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8. https://doi.org/10.1109/CVPR.2007.383282

[45] Wei Tong and Rong Jin. 2007. Semi-Supervised Learning by Mixed Label Propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 651–656. http://dl.acm.org/citation.cfm?id=1619645.1619750

[46] Bas Tummers. 2006. DataThief III. http://datathief.org/

[47] Zuobing Xu, Ram Akella, and Yi Zhang. 2007. Incorporating Diversity and Density in Active Learning for Relevance Feedback. In *Proceedings of the European Conference on IR Research*. Springer-Verlag, Rome, Italy, 246–257. http://dl.acm.org/citation.cfm?id=1763653.1763684

[48] David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, USA, 189–196. https://doi.org/10.3115/981658.981684

[49] Yu Zhang, Bob Coecke, and Min Chen. 2019. On the Cost of Interactions in Interactive Visual Machine Learning. In *Proceedings of IEEE VIS Workshop on Evaluation of Interactive Visual Machine Learning Systems*. IEEE, Vancouver, BC, Canada.

[50] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Sch{\"{o}}lkopf. 2003. Learning with Local and Global Consistency. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*. MIT Press, Whistler, British Columbia, Canada, 321–328. http://dl.acm.org/citation.cfm?id=2981345.2981386

[51] Yan Ping Zhou and Chew Lim Tan. 2000. Hough Technique for Bar Charts Detection and Recognition in Document Images. In *Proceedings of International Conference on Image Processing*, Vol. 2. IEEE, 605–608 vol.2. https://doi.org/10.1109/ICIP.2000.899506

[52] Xiaojin Zhu and Zoubin Ghahramani. 2002. *Learning from Labeled and Unlabeled Data with Label Propagation.* Technical Report.

A BLOCK NEIGHBOURHOOD FEATURES

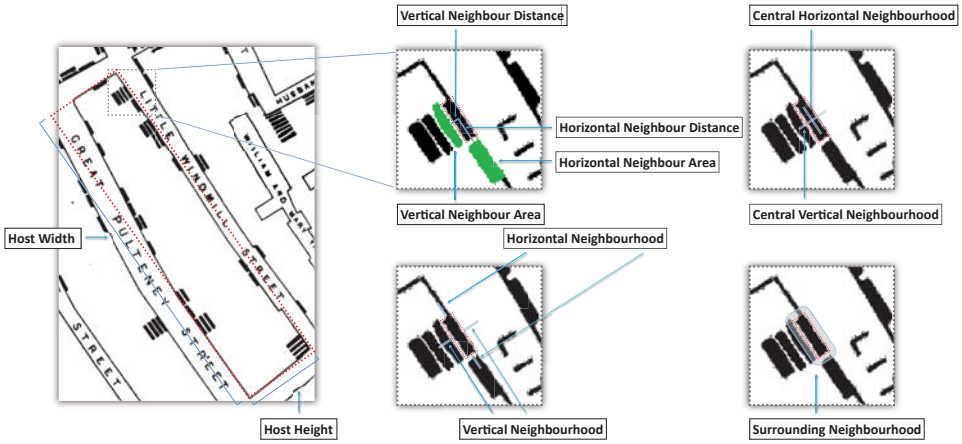


Fig. 12. Neighbourhood definitions and measurements. Horizontal neighbourhood refers to the union of 10 pixels horizontally extending outward the candidate in two directions. The definition of vertical and centralized neighbourhoods are similar. Surrounding neighbourhood refers to the union of pixels within 5 pixels' distance from the border of the candidate.

In some cases, true detections and false detections of blocks are not differentiable from their appearance. For example, dashed lines in the Cholera Map and blocks look identical. To handle such cases, we define neighbourhood features, to capture the image features around the candidate blocks. Fig. 12 illustrates some of the defined features.

Firstly, we compute neighbour distribution features to depict the distribution of other connected components around the candidate. The *horizontal neighbour distance* and *vertical neighbour distance* captures the distance to the nearest component in the horizontal direction and vertical direction respectively. The *surrounding neighbour number* feature denotes the number of adjacent components to the candidate.

Secondly, we compute neighbourhood foreground rate features to capture the distribution of foreground pixels in the neighbourhood of the candidate where foreground pixels are defined as the white pixels in the binarized image. *Surrounding foreground rate* is defined as the rate of foreground pixels within 5 pixels' distance from the candidate's border. *Horizontal foreground rate* and *vertical foreground rate* are defined as the rate of foreground pixels within 10 pixels' distance from the candidate's border in the horizontal direction and vertical direction, respectively. Similarly, *central horizontal foreground rate* and *central vertical foreground rate* measures the rate of foreground pixels in the two directions, while the difference is that central foreground rates concern the pixels within 10 pixels' distance from the candidate's centre instead of the border.

Thirdly, We also compute neighbour features to capture the appearance of the neighbours of the candidate. *Horizontal neighbour area* and *vertical neighbour area* measures the area of the closest neighbour in the horizontal and vertical direction. *Host width*, *host height*, *host area*, and *host solidity* measure the width, height, area, and solidity of the connected components containing the candidate.

B ROOT NEIGHBOURHOOD FEATURES

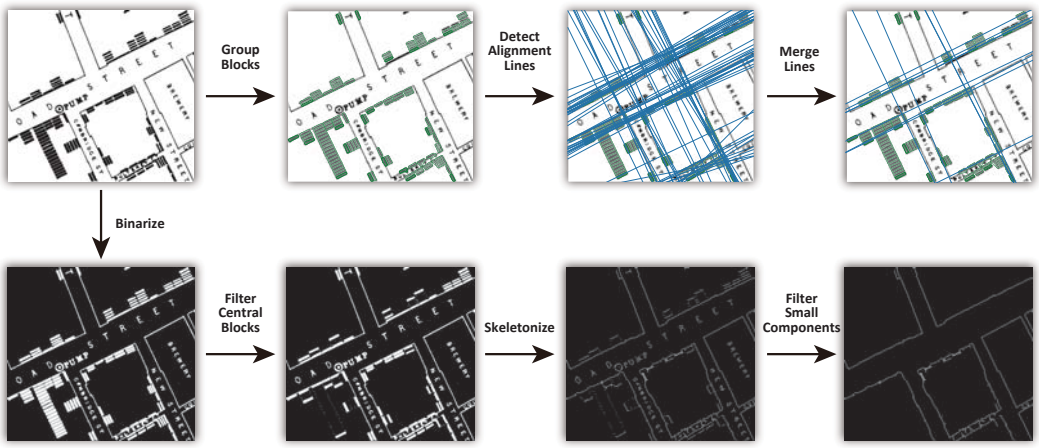


Fig. 13. The first row shows the process of computing alignment lines which are then used for computing the *distance to alignment line* feature. The second row shows the process of computing the skeletons which are then used for computing the *distance to skeleton* feature.

Distance to alignment line captures how well the candidate is aligned with other candidate roots. For each candidate root, we define the extending line of the medial axis of its closest bounding box as the *alignment line* that it generates. For all the alignment lines generated by all the candidate roots, we merge the alignment lines that are close to each other. Quantitatively, the pair of lines whose farthest pair of points is smaller than the minimum width and height of blocks are merged. For the merged lines, we refit the line with linear regression of the central point of the roots that generate the lines. Then, we filter the lines that cannot be merged with any other lines. This process of computing the alignment lines is shown in the first row of Fig. 13. With the merged lines (Fig. 13 top right), we can measure the *distance to alignment line* of each candidate root.

Distance to skeleton captures whether the candidate is close to a boundary line in the image. As shown in the second row of Fig. 13, to compute this feature, we detect the skeleton of the binarized image using Guo and Hall's thinning algorithm [13]. To remove noise for skeleton detection, we further adopt two strategies. Firstly, we filter all the pixels occupied by blocks that are not candidate roots before the skeleton detection. In this way, the pixels occupied by these blocks will not be mistaken for the skeleton. Secondly, after the skeleton detection, we filter the detected skeletons that are too small. We compute the connected components formed by the skeleton pixels using 8-connectivity and filter the connected components whose diagonal of the axis-aligned bounding box is shorter than twice the maximal width and height of blocks. Using the filtered skeleton, we measure the *distance to skeleton* of each candidate root.