This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

A Survey on Data Collection for Machine Learning

A Big Data - AI Integration Perspective

Yuji Roh, Geon Heo, Steven Euijong Whang, *Senior Member, IEEE*

Abstract—Data collection is a major bottleneck in machine learning and an active research topic in multiple communities. There are largely two reasons data collection has recently become a critical issue. First, as machine learning is becoming more widely-used, we are seeing new applications that do not necessarily have enough labeled data. Second, unlike traditional machine learning, deep learning techniques automatically generate features, which saves feature engineering costs, but in return may require larger amounts of labeled data. Interestingly, recent research in data collection comes not only from the machine learning, natural language, and computer vision communities, but also from the data management community due to the importance of handling large amounts of data. In this survey, we perform a comprehensive study of data collection from a data management point of view. Data collection largely consists of data acquisition, data labeling, and improvement of existing data or models. We provide a research landscape of these operations, provide guidelines on which technique to use when, and identify interesting research challenges. The integration of machine learning and data management for data collection is part of a larger trend of Big data and Artificial Intelligence (AI) integration and opens many opportunities for new research.

Index Terms—data collection, data acquisition, data labeling, machine learning

1 Introduction

We are living in exciting times where machine learning is having a profound influence on a wide range of applications from text understanding, image and speech recognition, to health care and genomics. As a striking example, deep learning techniques are known to perform on par with ophthalmologists on identifying diabetic eye diseases in images [1]. Much of the recent success is due to better computation infrastructure and large amounts of training data.

Among the many challenges in machine learning, data collection is becoming one of the critical bottlenecks. It is known that the majority of the time for running machine learning end-to-end is spent on preparing the data, which includes collecting, cleaning, analyzing, visualizing, and feature engineering. While all of these steps are time-consuming, data collection has recently become a challenge due to the following reasons.

First, as machine learning is used in new applications, it is usually the case that there is not enough training data. Traditional applications like machine translation or object detection enjoy massive amounts of training data that have been accumulated for decades. On the other hand, more recent applications have little or no training data. As an illustration, smart factories are increasingly becoming automated where product quality control is performed with machine learning. Whenever there is a new product or a new defect to detect, there is little or no training data to start with. The naïve approach of manual labeling may not be feasible because it is expensive and requires domain expertise. This problem applies to any novel application that benefits from machine learning.

Moreover, as deep learning [2] becomes popular, there is even more need for training data. In traditional machine learning, feature engineering is one of the most challenging steps where the user needs to understand the application and provide features used for training models. Deep learning, on the other hand, can automatically generate features, which saves us of feature engineering, which is a significant part of data preparation. However, in return, deep learning may require larger amounts of training data to perform well [3].

As a result, there is a pressing need of accurate and scalable data collection techniques in the era of Big data, which motivates us to conduct a comprehensive survey of the data collection literature from a data management point of view. There are largely three methods for data collection. First, if the goal is to share and search new datasets, then data acquisition techniques can be used to discover, augment, or generate datasets. Second, once the datasets are available, various data labeling techniques can be used to label the individual examples. Finally, instead of labeling new datasets, it may be better to improve existing data or train on top of trained models. These three methods are not necessarily distinct and can be used together. For example, one could search and label more datasets while improving existing ones.

An interesting observation is that the data collection techniques come not only from the machine learning community (including natural language processing and computer vision, which traditionally use machine learning heavily), but have also been studied for decades by the data management community, mainly under the names of data science and data analytics. Figure 1 shows an overview of

- *Y. Roh, G. Heo, and S. E. Whang are with the School of Electrical Engineering, KAIST, Daejeon, Korea.*
  E-mail: {yuji.roh, geon.heo, swhang}@kaist.ac.kr
- *Corresponding author: S. E. Whang*

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

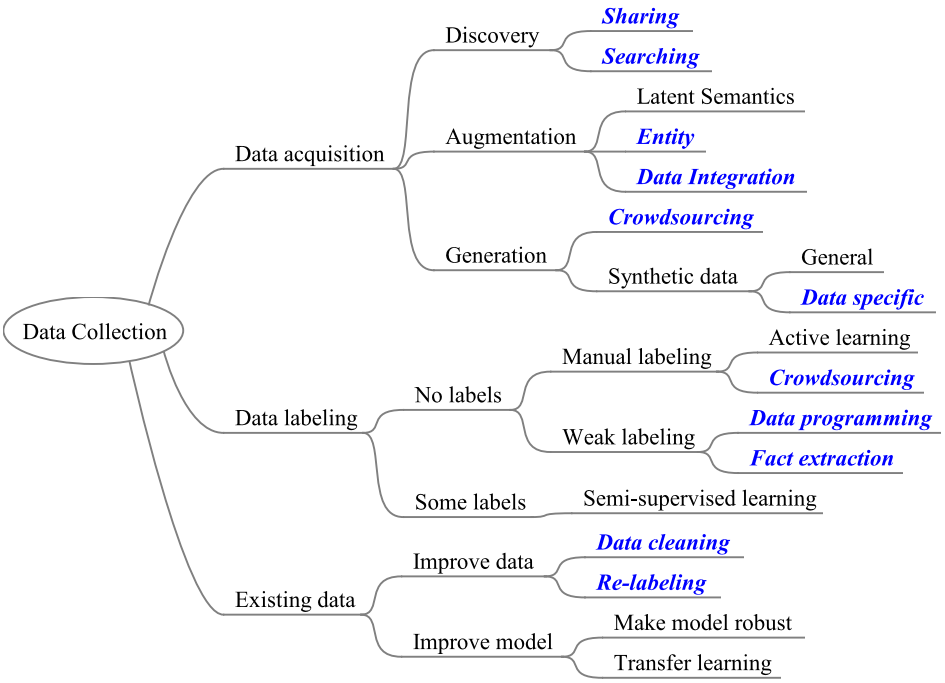


Fig. 1: A high level research landscape of data collection for machine learning. The topics that are at least partially contributed by the data management community are highlighted using blue italic text. Hence, to fully understand the research landscape, one needs to look at the literature from the viewpoints of both the machine learning and data management communities.

the research landscape where the topics that have contributions from the data management community are highlighted with blue italic text. Traditionally, labeling data has been a natural focus of research for machine learning tasks. For example, semi-supervised learning is a classical problem where model training is done on a small amount of labeled data and a larger amount of unlabeled data. However, as machine learning needs to be performed on large amounts of training data, data management issues including how to acquire large datasets, how to perform data labeling at scale, and how to improve the quality of large amounts of existing data become more relevant. Hence, to fully understand the research landscape of data collection, one needs to understand the literature from both the machine learning and data management communities.

While there are many surveys on data collection that are either limited to one discipline or a class of techniques, to our knowledge, this survey is the first to bridge the machine learning (including natural language processing and computer vision) and data management disciplines. We contend that a machine learning user needs to know the techniques on all sides to make informed decisions on which techniques to use when. In fact, data management plays a role in almost all aspects of machine learning [4], [5]. We note that many sub-topics including semi-supervised learning, active learning, and transfer learning are large enough to have their own surveys. The goal of this survey is not to go into all the depths of these sub-topics, but to focus on breadth and identify what data collection techniques are relevant for machine learning purposes and what research challenges exist. Hence, we will only cover the most representative work of the sub-topics, which are either the best-performing or most recent ones. The key audience of this survey can be researchers or practitioners that are starting to use data collection for machine learning and need an overall landscape introduction. Since the data collection techniques come from different disciplines, some may involve relational data while others non-relational data (e.g., images and text). Sometimes the boundary between operations (e.g., data acquisition and data labeling) is not clear cut. In those cases, we will clarify that the techniques are relevant in multiple operations.

Motivating Example To motivate the need to explore the techniques in Figure 1, we present a running example on data collection based on our experience with collaborating with the industry on a smart factory application. Suppose that Sally is a data scientist who works on product quality for a smart factory. The factory may produce manufacturing components like gears where it is important for them not to have scratches, dents, or any foreign substance. Sally may want to train a model on images of the components, which can be used to automatically classify whether each product has defects or not. This application scenario is depicted in Figure 3. A general decision flow chart of the data collection techniques that Sally can use is shown in Figure 2. Although the chart may look complicated at first glance, we contend that it is necessary to understand the entire research landscape to make informed decisions for data collection. In comparison, recent commercial tools [6]–[8] only cover a subset of all the possible data collection techniques. When using the chart, one can quickly narrow down the options in two steps by deciding whether to perform one of data acquisition, data labeling, or existing data improvements,

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

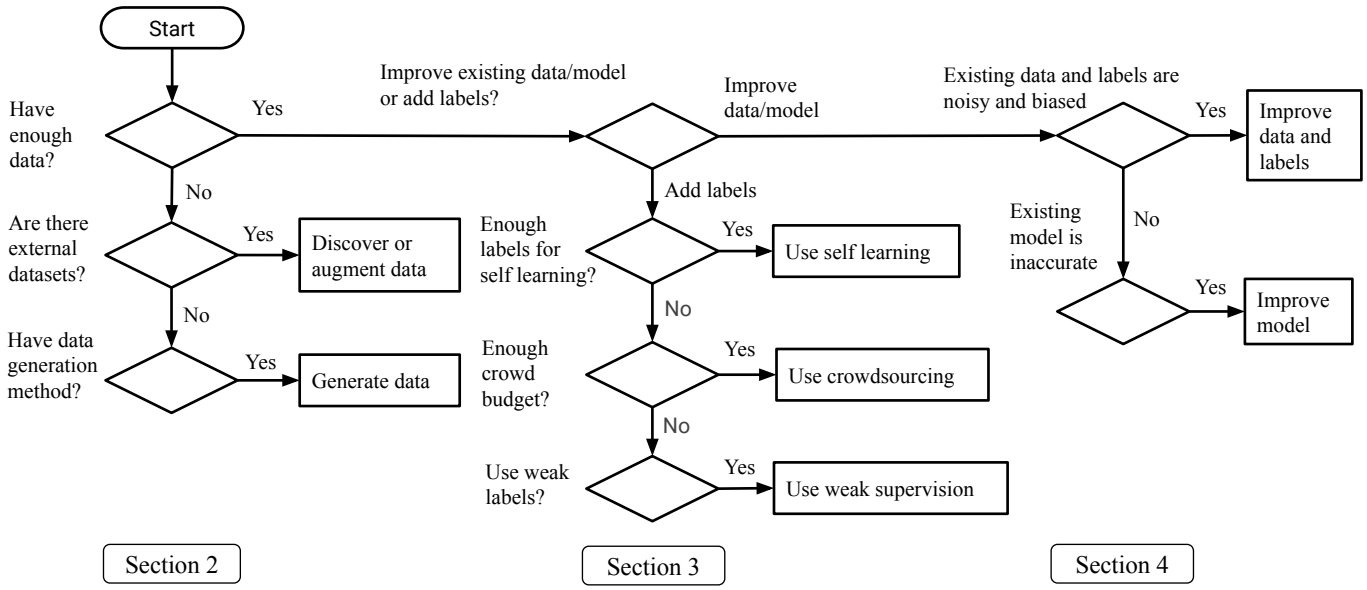


Fig. 2: A decision flow chart for data collection. From the top left, Sally can start by asking whether she has enough data. The following questions lead to specific techniques that can be used for acquiring data, labeling data, or improving existing data or models. This flow chart does not cover all the details in this survey. For example, data labeling techniques like self learning and crowdsourcing can be performed together as described in Section 3.2.1. Also, some questions (e.g., "Enough labels for self learning?") are not easy to answer and may require an in-depth understanding of the application and data. There are also techniques specific to the data type (images and text), which we detail in the body of the paper.

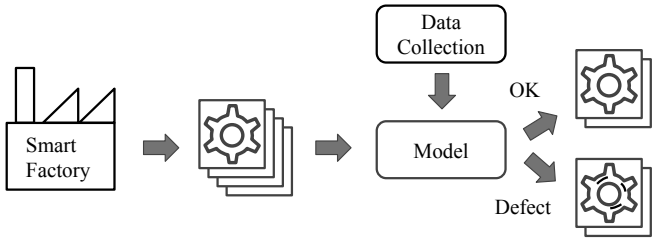


Fig. 3: A running example for data collection. A smart factory may produce various images of product components, which are classified as normal or defective by a convolutional neural network model. Unfortunately, with an application this specific, it is often difficult to find enough data for training the model.

and then choosing the specific technique to use for each operation. For example, if there is no data, then Sally could generate a dataset by installing camera equipment. Then if she has enough budget for human computation, she can use crowdsourcing platforms like Amazon Mechanical Turk to label the product images for defects. We will discuss more details of the flow chart in the following sections.

The rest of the paper is organized as follows:

- We review the data acquisition literature, which can be categorized into data discovery, data augmentation, and data generation. Many of the techniques require scalable solutions and have thus been studied by the data management community (Section 2).
- We review the data labeling literature and group the techniques into three approaches: utilizing existing labels, using crowdsourcing techniques, and using weak supervision. While data labeling is traditionally a machine learning topic, it is also studied in the data management community as scalability becomes an issue (Section 3).
- We review techniques for improving existing data or models when acquiring and labeling new data is not the best option. Improving data quality through cleaning is a traditional data management topic where recent techniques are increasingly focusing on machine learning applications (Section 4).
- We put all the techniques together and provide guidelines on how to decide which data collection techniques to use when (Section 5).
- Based on the current research landscape, we identify interesting future research challenges (Section 6).

2 Data Acquisition

The goal of data acquisition is to find datasets that can be used to train machine learning models. There are largely three approaches in the literature: data discovery, data augmentation, and data generation. Data discovery is necessary when one wants to share or search for new datasets and has become important as more datasets are available on the Web and corporate data lakes [19], [75]. Data augmentation complements data discovery where existing datasets are enhanced by adding more external data. Data generation can be used when there is no available external dataset, but it is possible to generate crowdsourced or synthetic datasets instead. The following sections will cover the three operations in more detail. The individual techniques are classified in Table 1.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

Task	Approach	Techniques
Data discovery	Sharing	Collaborative Analysis [9]–[11]
		Web [12]–[17]
		Collaborative and Web [18]
	Searching	Data Lake [19], [19]–[23]
		Web [24]–[34]
Data augmentation		Deriving Latent Semantics [35]–[37]
		Entity Augmentation [30], [31]
		Data Integration [38]–[44]
Data generation	Crowdsourcing	Gathering [45]–[54]
		Processing [49], [50], [55], [56]
	Synthetic Data	Generative Adversarial Networks [57]–[62]
		Policies [63], [64]
		Image [65]–[71]
		Text [72]–[74]

TABLE 1: A classification of data acquisition techniques. Some of the techniques can be used together. For example, data can be generated while augmenting existing data.

2.1 Data Discovery

Data discovery can be viewed as two steps. First, the generated data must be indexed and published for sharing. Many collaborative systems are designed to make this process easy. However, other systems are built without the intention of sharing datasets. For these systems, a *post-hoc* approach must be used where metadata is generated after the datasets are created, without the help of the dataset owners. Next, someone else can search the datasets for their machine learning tasks. Here the key challenges include how to scale the searching and how to tell whether a dataset is suitable for a given machine learning task. While most of the data discovery literature came from the data management community for data science and data analytics, they are also relevant in a machine learning context. However, another challenge in machine learning is data labeling, which we cover in Section 3.

2.1.1 Data Sharing

We study data systems that are designed with dataset sharing in mind. These systems may focus on collaborative analysis, publishing on the Web, or both.

Collaborative Analysis In an environment where data scientists are collaboratively analyzing different versions of datasets, DataHub [9]–[11] can be used to host, share, combine, and analyze them. There are two components: a dataset version control system inspired by Git (a version control system for code) and a hosted platform on top of it, which provides data search, data cleaning, data integration, and data visualization. A common use case of DataHub is where individuals or teams run machine learning tasks on their own versions of a dataset and later merge with other versions if necessary.

Web A different approach of sharing datasets is to publish them on the Web. Google Fusion Tables [12]–[14] is a cloud-based service for data management and integration. Fusion Tables enables users to upload structured data (e.g., spreadsheets) and provides tools for visually analyzing, filtering, and aggregating the data. The datasets that are published through Fusion Tables on the Web can be crawled by search engines and show up in search results. The datasets are therefore primarily accessible through Web search. Fusion Tables has been widely used in data journalism for creating interactive maps of data and adding them in articles. In addition, there are many data marketplaces including CKAN [15], Quandl [16], and DataMarket [17] where users can buy and sell datasets or find public datasets.

Collaborative and Web More recently, we are seeing a merging of collaborative and Web-based systems. For example, Kaggle [18] makes it easy to share datasets on the Web and even host data science competitions for models trained on the datasets. A Kaggle competition host posts a dataset along with a description of the challenge. Participants can then experiment with their techniques and compete with each other. After the deadline passes, a prize is given to the winner of the competition. Kaggle currently has thousands of public datasets and code snippets (called kernels) from competitions. In comparison to DataHub and Fusion Tables, the Kaggle datasets are coupled with competitions and are thus more readily usable for machine learning purposes.

2.1.2 Data Searching

While the previous data systems are platforms for sharing datasets, as a next logical step, we now explore systems that are mainly designed for searching datasets. This setting is common within large companies or on the Web.

Data Lake Data searching systems have become more popular with the advent of *data lakes* [19], [75] in corporate environments where many datasets are generated internally, but they are not easily discoverable by other teams or individuals within the company. Providing a way to search datasets and analyze them has significant business value because the teams or individuals do not have to make redundant efforts to re-generate the datasets for their machine learning tasks. Most of the recent data lake systems have come from the industry. In many cases, it is not feasible for all the dataset owners to publish datasets through one system. Instead, a post-hoc approach becomes necessary where datasets are

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

processed for searching after they are created, and no effort is required on the dataset owner's side.

As an early solution for data lakes, IBM proposed a system [19] that enables datasets to be curated and then searched. IBM estimates that 70% of the time spent on analytic projects is concerned with discovering, cleaning, and integrating datasets that are scattered among many business applications. Thus, IBM takes the stance of creating, filling, maintaining, and governing the data lake where these processes are collectively called *data wrangling*. When analyzing data, users do not perform the analytics directly on the data lake, but extract data sets and store them separately. Before this step, the users can do a preliminary exploration of datasets, e.g., visualizing them to determine if the dataset is useful and does not contain anomalies that need further investigation. While supporting data curation in the data lake saves users from processing raw data, it does limit the scalability of how many datasets can be indexed.

More recently, scalability has become a pressing issue for handling data lakes that consists of most datasets in a large company. Google Data Search (GOODS) [20] is a system that catalogues the metadata of tens of billions of datasets from various storage systems within Google. GOODS infers various metadata including owner information and provenance information (by looking up job logs), analyzes the contents of the datasets, and collects input from users. At the core is a central catalog, which contains the metadata and is indexed for data searching. Due to Google's scale, there are many technical challenges including scaling to the number of datasets, supporting a variety of data formats where the costs for extracting metadata may differ, updating the catalog entries due to the frequent churn of datasets, dealing with uncertainty in metadata discovery, computing dataset importance for search ranking, and recovering dataset semantics that are missing. To find datasets, users can use keywords queries on the GOODS frontend and view profile pages of the datasets that appear in the search results. In addition, users can track the provenance of a dataset to see which datasets were used to create the given dataset and those that rely on it.

Finally, expressive queries are also important for searching a data lake. While GOODS scales, one downside is that it only supports simple keyword queries. This approach is similar to keyword search in databases [76], [77], but the purpose is to find datasets instead of tuples. The DATA CIVILIZER system [21], [22] complements GOODS by focusing more on the discovery aspect of datasets. Specifically, DATA CIVILIZER consists of a module for building a linkage graph of data. Assuming that datasets have schema, the nodes in the linkage graph are columns of tables while edges are relationships like primary key-foreign key (PK-FK) relationships. A data discovery module then supports a rich set of *discovery queries* on the linkage graph, which can help users more easily discover the relevant datasets. DATARAMAN [23] specializes in extracting structured data from semi-structured log datasets in data lakes automatically by learning patterns. AURUM [78], [79] supports data discovery queries on semantically-linked datasets.

Web As the Web contains large numbers of structured datasets, there have been significant efforts to automatically extract the useful ones [32]–[34]. One of the most successful systems is WebTables [24], [25], which automatically extracts structured data that is published online in the form of HTML tables. For example, WebTables extracts all Wikipedia infoboxes. Initially, about 14.1 billion HTML tables are collected from the Google search web crawl. Then a classifier is applied to determine which tables can be viewed as relational database tables. Each relational table consists of a schema that describes the columns and a set of tuples. In comparison to the above data lake systems, WebTables collects structured data from the Web.

As Web data tends to be much more diverse than say those in a corporate environment, the table extraction techniques have been extended in multiple ways as well. One direction is to extend table extraction beyond identifying HTML tags by extracting relational data in the form of vertical tables and lists and leveraging knowledge bases [27], [28]. Table searching also evolved where, in addition to keyword searching, row-subset queries, entity-attribute queries, and column search were introduced [29]. Finally, techniques for enhancing the tables [30], [31] were proposed where entities or attribute values are added to make the tables more complete.

Recently, a service called Google Dataset Search [26] was launched for searching repositories of datasets on the Web. The motivation is that there are thousands of data repositories on the Web that contain millions of datasets that are not easy to search. Dataset Search lets dataset providers describe their datasets using various metadata (e.g., author, publication date, how the data was collected, and terms for using the data) so that they become more searcheable. In comparison to the fully-automatic WebTables, dataset providers may need to do some manual work, but have the opportunity to make their datasets more searcheable. In comparison to GOODS, Dataset Search targets the Web instead of a data lake.

2.2 Data Augmentation

Another approach to acquiring data is to augment existing datasets with external data. In the machine learning community, adding pre-trained embeddings is a common way to increase the features to train on. In the data management community, entity augmentation techniques have been proposed to further enrich existing entity information. Data integration is a broad topic and can be considered as data augmentation if we are extending existing datasets with newly-acquired ones.

2.2.1 Deriving Latent Semantics

A common data augmentation is to derive latent semantics from data. A popular technique is to generate and use embeddings that represent words, entities, or knowledge. In particular, word embeddings have been successfully used to solve many problems in natural language processing (NLP). Word2vec [35] is a seminal work where, given a text corpus, a word is represented by a vector of real numbers that captures the linguistic context of the word in the corpus. The word vectors can be generated by training a shallow two-layer neural network to reconstruct the surrounding words in the corpus. There are two possible models for training

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

word vectors: Continuous Bag-of-Words (CBOW) and Skip-gram. While CBOW predicts a word based on its surrounding words, Skip-gram does the opposite and predicts the surrounding words based on a given word. As a result, two words that occur in similar contexts tend to have similar word vectors. A fascinating application of word vectors is performing arithmetic operations on the word vectors. For example, the result of subtracting the word vector of "king" by that of "queen" is similar to the result of subtracting the word vector of "man" by that of "woman". Since word2vec was proposed, there have been many extensions including GloVe [36], which improves word vectors by also taking into account global corpus statistics, and Doc2Vec [80], which generates representations of documents.

Another technique for deriving latent semantics is latent topic modeling. For example, Latent Dirichlet Allocation [37] (LDA) is a generative model that can be used to explain why certain parts of the data are similar using unobserved groups.

2.2.2 Entity Augmentation

In many cases, datasets are incomplete and need to be filled in by gathering more information. The missing information can either be values or entire features. An early system called Octopus [30] composes Web search queries using keys of the table containing the entities. Then all the Web tables in the resulting Web pages are clustered by schema, and the tables in the most relevant cluster are joined with the entity table. More recently, InfoGather [31] takes a holistic approach using Web tables on the Web. The entity augmentation is performed by filling in missing values of attributes in some or all of the entities by matching multiple Web tables using schema matching. To help the user decide which attributes to fill in, InfoGather identifies synonymous attributes in the Web tables.

2.2.3 Data Integration

Data integration can also be considered as data augmentation, especially if we are extending existing data sets with other acquired ones. Since this discipline is well established, we point the readers to some excellent surveys [40], [41]. More recently, an interesting line of work relevant to machine learning [42]–[44] observes that in practice, many companies use relational databases where the training data is divided into smaller tables. However, most machine learning toolkits assume that a training dataset is a single file and ignore the fact that there are typically multiple tables in a database due to normalization. The key question is whether joining the tables and augmenting the information is useful for model training. The Hamlet system [38] and its subsequent Hamlet++ systems [39] address this problem by determining if key-foreign key (KFK) joins are necessary for improving the model accuracy for various classifiers (linear, decision trees, non-linear SVMs, and artificial neural networks) and propose decision rules to predict when it is safe to avoid joins and, as a result, significantly reduce the total runtime. A surprising result is that joins can often be avoided without negatively influencing the model's accuracy. Intuitively, a foreign key determines the entire record of the joining table, so the features brought in by a join do not add a lot more information.

2.3 Data Generation

If there are no existing datasets that can be used for training, then another option is to generate the datasets either manually or automatically. For manual construction, crowdsourcing is the standard method where human workers are given tasks to gather the necessary bits of data that collectively become the generated dataset. Alternatively, automatic techniques can be used to generate synthetic datasets. Note that data generation can also be viewed as data augmentation if there is existing data where some missing parts needs to be filled in.

2.3.1 Crowdsourcing

Crowdsourcing is used to solve a wide range of problems, and there are many surveys as well [81]–[84]. One of the earliest and most popular platforms is Amazon Mechanical Turk [85] where tasks (called *HITs*) are assigned to human workers, and workers are compensated for finishing the tasks. Since then, many other crowdsourcing platforms have been developed, and research on crowdsourcing has flourished in the areas of data management, machine learning, and human computer interaction. There is a wide range of crowdsourcing tasks from simple ones like labeling images up to complex ones like collaboratively writing that involve multiple steps [86], [87]. Another important usage of crowdsourcing is data labeling (e.g., the ImageNet project), which we discuss in Section 3.2.

In this section, we narrow the scope and focus on crowdsourcing techniques that are specific to data generation tasks. A recent survey [88] provides an extensive discussion on the challenges for data crowdsourcing. Another survey [89] touches on the theoretical foundations of data crowdsourcing. According to both surveys, data generation using crowdsourcing can be divided into two steps: *gathering* data and *preprocessing* data.

Gathering data One way to categorize data gathering techniques is whether the tasks are procedural or declarative. A procedural task is where the task creator defines explicit steps and assigns them to workers. For example, one may write a computer program that issues tasks to workers. TurKit [45] allows users to write scripts that include HITs using a crash-and-return programming model where a script can be re-executed without re-running costly functions with side effects. AutoMan [46] is a domain-specific language embedded in Scala where crowdsourcing tasks can be invoked like conventional functions. Dog [47] is a high-level programming language that compiles into MapReduce tasks that can be performed by humans or machines. A declarative task is when the task creator specifies high-level data requirements, and the workers provide the data that satisfy them. For example, a database users may pose an SQL query like "SELECT title, director, genre, rating FROM MOVIES WHERE genre = 'action"' to gather movie ratings data for a recommendation system. Deco [48] uses a simple extension of SQL and defines precise semantics for arbitrary queries on stored data and data collected by the crowd. CrowdDB [49] focuses on the systems aspect of using crowdsourcing to answer queries that cannot be answered automatically.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

Another way to categorize data gathering is whether the data is assumed to be closed-world or open-world. Under a closed-world assumption, the data is assumed to be "known" and entirely collectable by asking the right questions. ASKIT! [51] uses this assumption and focuses on the problem of determining which questions should be directed to which users, in order to minimize the uncertainty of the collected data. In an open-world assumption, there is no longer a guarantee that all the data can be collected. Instead, one must estimate if enough data was collected. Statistical tools [52] have been proposed for scanning a single table with predicates like "SELECT FLAVORS FROM ICE_CREAM." Initially, many flavors can be collected, but the rate of new flavors will inevitably slow down, and statistical methods are used to estimate the future rate of new values.

Data gathering is not limited to collecting entire records of a table. CrowdFill [53] is a system for collecting parts of structured data from the crowd. Instead of posing specific questions to workers, CrowdFill shows a partially-filled table. Workers can then fill in the empty cells and also upvote or downvote data entered by other workers. CrowdFill provides a collaborative environment and allows the specification of constraints on values and mechanisms for resolving conflicts when workers are filling in values of the same record. ALFRED [54] uses the crowd to train extractors that can then be used to acquire data. ALFRED asks simple yes/no membership questions on the contents of Web pages to workers and uses the answers to infer the extraction rules. The quality of the rules can be improved by recruiting multiple workers.

Preprocessing data Once the data is gathered, one may want to preprocess the data to make it suitable for machine learning purposes. While many possible crowd operations have been proposed, the ones that are relevant include data curation, entity resolution, and joining datasets. Data Tamer [55] is an end-to-end data curation system that can clean and transform datasets and semantically integrate with other datasets. Data Tamer has a crowdsourcing component (called Data Tamer Exchange), which assigns tasks to workers. The supported operations are attribute identification (i.e., determine if two attributes are the same) and entity resolution (i.e., determine if two entities are the same). Corleone [56] is a hands-off crowdsourcing system, which crowdsources the entire workflow of entity resolution to workers. CrowdDB [49] and Qurk [50] are systems for aggregating, sorting, and joining datasets.

For both gathering and preprocessing data, quality control is an important challenge as well. The issues include designing the right interface to maximize worker productivity, managing workers who may have different levels of skills (or may even be spammers), and decomposing problems into smaller tasks and aggregating them. Several surveys [82]–[84] cover these issues in detail.

2.3.2 Synthetic Data Generation

Generating synthetic data along with labels is increasingly being used in machine learning due to its low cost and flexibility [90]. A simple method is to start from a probability distribution and generate a sample from that distribution using tools like scikit learn [91]. In addition, there are more advanced techniques like Generative Adversarial Networks (GANs) [2], [57], [61], [62] and application-specific generation techniques. We first provide a brief introduction of GANs and present synthetic data generation techniques on relational data. We then introduce recent augmentation techniques using policies. Finally, we introduce image and text data generation techniques due to their importance.

GANs The key approach of a GAN is to train two contesting neural networks: a generative network and a discriminative network. The generative network learns to map from a latent space to a data distribution, and the discriminative network discriminates examples from the true distribution from the candidates produced by the generative network.

The training of a GAN can be formalized as:

$$\min_G \max_D V(D,G)$$
$$V(D,G) = \mathop{\mathbb{E}}_{x\sim p_{data}(x)}[\log D(x)] + \mathop{\mathbb{E}}_{z\sim p_z(z)}[\log(1-D(G(z))]$$

where $p_{data}(x)$ is the distribution of the real data, $p_z(z)$ is the distribution of the generator, $G(z)$ is the generative network, and $D(x)$ is the discriminative network.

The objective of the generative network is to increase the error rate of the discriminative network. That is, the generative network attempts to fool the discriminative network into thinking that its candidates are from the true distribution. GANs have been used to generate synthetic images and videos that look realistic in many applications.

GANs have recently been used to generate synthetic relational data. A MEDGAN [58] generates synthetic patient records with high-dimensional discrete variable (binary or count) features based on real patient records. While GANs can only learn to approximate discrete patient records, the novelty is to also use an autoencoder to project these records into a lower dimensional space and then project them back to the original space. A TABLE-GAN [59] also synthesizes tables that are similar to the real ones, but with a focus on privacy perservation. In particular, a metric for information loss is defined, and two parameters are provided to adjust the information loss. The higher the loss, the more privacy the synthetic table has. A TGAN [60] focuses on simultaneously generating values for a mixture of discrete and continuous features.

Policies Another recent approach is to use human-defined policies [63], [64] to apply transformations to the images as long as they remain realistic. This criteria can be enforced by training a reinforcement learning model on a separate validation set.

Data-specific We now introduce data-specific techniques for generation. Synthetic image generation is a heavily-studied topic in the computer vision community. Given the wide range of vision problems, we are not aware of a comprehensive survey on synthetic data generation and will only focus on a few representative problems. In object detection, it is possible to learn 3D models of objects and give variations (e.g., rotate a car 90 degrees) to generate another realistic image [65], [66]. If the training data is a rapid sequence of images frames in time [67] the objects of a frame can be assumed to move in a linear trajectory between consecutive

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

frames. Text within images is another application where one can vary the fonts, sizes, and colors of the text to generate large amounts of synthetic text images [68], [69].

An alternative approach to generating image datasets is to start from a large set of noisy images and select the clean ones. Xia et al. [70] searches the Web for images with noise and then uses a density-based measure to cluster the images and remove outliers. Bai et al. [71] exploits large click-through logs, which contains queries of users and the images that were clicked by those users. A deep neural network is used to learn representations of the words and images and compute word-word and image-word similarities. The noisy images that have low similarities to their categories are then removed.

Generating synthetic text data has also been studied in the natural language processing community. Paraphrasing [72] is a classical problem of generating alternative expressions that have the same semantic meaning. For example "What does X do for a living?" is a paraphrase of "What is X's job?". We briefly cover two recent methods – one is syntax-based and the other semantics-based – that uses paraphrasing to generate large amounts of synthetic text data. Syntactically controlled paraphrase networks [73] (SCPNs) can be trained to produce paraphrases of a sentence with different sentence structures. Semantically equivalent adversarial rules for text [74] (SEARs) have been proposed for perturbing input text while preserving its semantics. SEARs can be used to debug a model by applying them on training data and seeing if the re-trained model changes its predictions. In addition, there are many paraphrasing techniques that are not covered in this survey.

3 Data Labeling

Once enough data has been acquired, the next step is to label individual examples. For instance, given an image dataset of industrial components in a smart factory application, workers can start annotating if there are any defects in the components. In many cases, data acquisition is done along with data labeling. When extracting facts from the Web and constructing a knowledge base, then each fact is assumed to be correct and thus implicitly labeled as true. When discussing the data labeling literature, it is easier to separate it from data acquisition as the techniques can be quite different.

We believe the following categories provide a reasonable view of understanding the data labeling landscape:

- *Use existing labels:* An early idea of data labeling is to exploit any labels that already exist. There is an extensive literature on semi-supervised learning where the idea is to learn from the labels to predict the rest of the labels.
- *Crowd-based:* The next set of techniques are based on crowdsourcing. A simple approach is to label individual examples. A more advanced technique is to use active learning where questions to ask are more carefully selected. More recently, many crowdsourcing techniques have been proposed to help workers become more effective in labeling.
- *Weak labels:* While it is desirable to generate correct labels all the time, this process may be too expensive. An alternative approach is to newly generate less than perfect labels (i.e., weak labels), but in large quantities to compensate for the lower quality. Recently, the latter approach is gaining more popularity as labeled data is scarce in many new applications.

Table 2 shows where different labeling approaches fit into the categories. In addition, each labeling approach can be further categorized as follows:

- *Machine learning task:* In supervised learning, the two categories are classification (e.g., determining whether a piece of text has a positive sentiment) and regression (e.g., estimating the salary of a person). Most of the data labeling research has been focused on classification problems rather than regression problems, possibly because data labeling is simpler in a classification setting.
- *Data type:* Depending on the data type (e.g., text, images, and graphs) the data labeling techniques differ significantly. For example, fact extraction from text is very different from object detection on images.

3.1 Utilizing existing labels

A common setting in machine learning is to have a small amount of labeled data, which is expensive to produce with humans, along with a much larger amount of unlabeled data. Semi-supervised learning techniques [143] exploit both labeled and unlabeled data to make predictions. In a transductive learning setting, the entire unlabeled data is available while in an inductive learning setting, some unlabeled data is available, but the predictions must be on unseen data. Semi-supervised learning is a broad topic, and we focus on a smaller branch of research called *self-labeled techniques* [96] where the goal is to generate more labels by trusting one's own predictions. Since the details are in the survey, we only provide a summary here. In addition to the general techniques, there are graph-based label propagation techniques that are specialized for graph data.

3.1.1 Classification

For semi-supervised learning techniques for classification, the goal is to train a model that returns one of multiple possible classes for each example using labeled and unlabeled datasets. We consider the best-performing techniques in a survey that focuses on labeling data [96], which are summarized in Figure 4. The performance results are similar regardless of using transductive or inductive learning.

The simplest class of semi-supervised learning techniques train one model using one learning algorithm on one set of features. For example, Self-training [92] initially trains a model on the labeled examples. The model is then applied to all the unlabeled data where the examples are ranked by the confidences in their predictions. The most confident predictions are then added into the labeled examples. This process repeats until all the unlabeled examples are labeled.

The next class trains multiple classifiers by sampling the training data several times and training a model for each sample. For example, Tri-training [93] initially trains three models on the labeled examples using Bagging for the ensemble learning algorithm. Then each model is updated

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

Category	Approach	Machine learning task	Data types	Techniques
Use Existing Labels	Self-labeled	classification	all	[92]–[96]
		regression	all	[97]–[99]
	Label propagation	classification	graph	[100]–[102]
Crowd-based	Active learning	classification	all	[103]–[109]
		regression	all	[110]
	Semi-supervised+Active learning	classification	text	[111], [112]
			image	[113]
			graph	[114]
	Crowdsourcing	classification	all	[50], [54], [115]–[122]
		regression	all	[123]
Weak supervision	Data programming	classification	all	[3], [124]–[127], [127]–[130]
	Fact extraction	classification	text	[131]–[142]

TABLE 2: A classification of data labeling techniques. Some of the techniques can be used for the same application. For example, for classification on graph data, both self-labeled techniques and label propagation can be used.

iteratively where the other two models make predictions on the unlabeled examples, and only the examples with the same predictions are used in conjunction with the original labeled examples to re-train the model. The iteration stops when no model changes. Finally, the unlabeled examples are labeled using majority voting where at least two models must agree with each other.

The next class uses multiple learning algorithms. For example, Democratic Co-learning [94] uses a set of different learning algorithms (in the experiments, they are Naive Bayes, C4.5, and 3-nearest neighbor) are used to train a set of classifiers separately on the same training data. Predictions on new examples are generated by combining the results of the three classifiers using weighted voting. The new labels are then added to the training set of the classifiers whose predictions are different from the majority results. This process repeats until no more data is added to the training data of a classifier.

The final class uses multiple views, which are subsets of features that are conditionally independent given the class. For example, Co-training [95] splits the feature set into two sufficient and redundant views, which means that one set of features is sufficient for learning and independent of learning with the other set of features given the label. For each feature set, a model is trained and then used to teach the model trained on the other feature set. The co-trained models can minimize errors by maximizing their agreements over the unlabeled examples.

According to the survey, these algorithms result in similar transductive or inductive accuracies when averaged on 55 datasets from the UCI [144] and KEEL dataset [145] repositories. Here accuracy is defined as the portion of classifications by the trained model that are correct. However, not all of these techniques are generally-applicable where one can plug in any machine learning algorithm and use any set of features. First, Co-training assumes that the feature can be divided into two subsets that are conditionally independent given the label (also called sufficient and redundant views), which is not always possible. Second, Democratic Co-learning assumes three different machine learning algorithms, which is not always possible if there is only one algorithm to use.

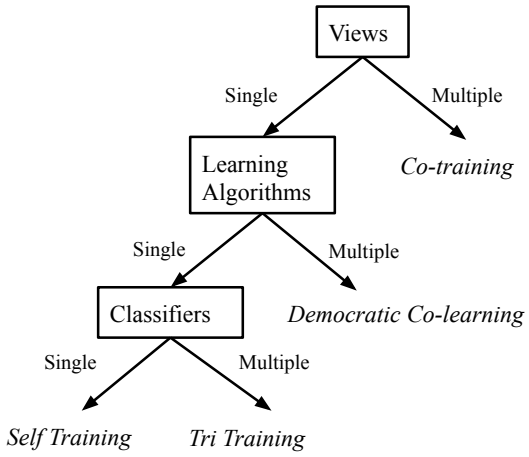


Fig. 4: A simplified classification of semi-supervised learning techniques for self labeling according to a survey [96] using the best-performing techniques regardless of inductive or transductive learning.

3.1.2 Regression

Relatively less research has been done for semi-supervised learning for regression where the goal is to train a model that predicts a real number given an example. Co-regularized least squares regression [97] is a least squares regression algorithm based on the co-learning approach. Another co-regularized framework [98] utilizes sufficient and redundant views similar to Co-training. Co-training regressors [99] uses two k-nearest neighbor regressors with different distance metrics. In each iteration, a regressor labels the unlabeled data that can be labeled most confidently by the other regressor. After the iterations, the final prediction of an example is made by averaging the regression estimates by the two regressors. Co-training Regressors can be extended by using any other base regressor.

3.1.3 Graph-based Label Propagation

Graph-based label propagation techniques also start with limited sets of labeled examples, but exploit the graph structure of examples based on their similarities to infer the labels of the remaining examples. For example, if an

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

image is labeled as a dog, then similar images down the graph can also be labeled as dogs with some probability. The further the distance, the lower the probability of label propagation. Graph-based label propagation has applications in computer vision, information retrieval, social networks, and natural language processing. Zhu et al. [101] proposed a semi-supervised learning based on a Gaussian random field model where the unlabeled and labeled examples form a weighted graph. The mean of the field is characterized in terms of harmonic functions and can be efficiently computed using matrix methods or belief propagation. The MAD-Sketch algorithm [102] was proposed to further reduce the space and time complexities of graph-based SSL algorithms using count-min sketching. In particular, the space complexity per node is reduced from $O(m)$ to $O(\log m)$ under certain conditions where m is the number of distinct labels, and a similar improvement is achieved for the time complexity. Recently, a family of algorithms called EXPANDER [100] were proposed to further reduce the space complexity per node to $O(1)$ and compute the MAD-Sketch algorithm in a distributed fashion.

3.2 Crowd-based techniques

The most accurate way to label examples is to do it manually. A well known use case is the ImageNet image classification dataset [146] where tens of millions of images were organized according to a semantic hierarchy by WordNet using Amazon Mechanical Turk. However, ImageNet is an ambitious project that took years to complete, which most machine learning users cannot afford for their own applications. Traditionally, active learning has been a key technique in the machine learning community for carefully choosing the right examples to label and thus minimize cost. More recently, crowdsourcing techniques for labeling have been proposed where there can be many workers who are not necessarily experts in labeling. Hence, there is more emphasis on how to assign tasks to workers, what interfaces to use, and how to ensure high quality labels. Recent commercial tools vary in what services they provide for labeling. For example, Amazon SageMaker [8] supports labeling based on active learning, Google Cloud AutoML [6] provides a manual labeling service, and Microsoft Custom Vision [7] requires labels from the user. While crowdsourcing data labeling is closely related to crowdsourcing data acquisition, the individual techniques are different.

3.2.1 Active Learning

Active learning focuses on selecting the most "interesting" unlabeled examples to give to the crowd for labeling. The workers are expected to be very accurate, so there is less emphasis on how to interact with those with less expertise. While some references view active learning as a special case of semi-supervised learning, the key difference is that there is a human-in-the-loop. The key challenge is choosing the right examples to ask given a limited budget. One downside of active learning is that the examples are biased to the training algorithm and cannot be reused. Active learning is covered extensively in other surveys [105], [147], and we only cover the most prominent techniques here.

Uncertain Examples Uncertainty Sampling [103] is the simplest in active learning and chooses the next unlabeled example that the model prediction is most uncertain. For example, if the model is a binary classifier, uncertainty sampling chooses the example whose probability is nearest to 0.5. If there are more than three class labels, we could choose the example whose prediction is the least confident. The downside of this approach is that it throws away the information of all the other possible labels. So an improved version called margin sampling is to choose the example whose probability difference between the most and second-most probable labels is the largest. This method can be further generalized using entropy as the uncertainty measure where entropy is an information-theoretic measure for the amount of information to encode a distribution.

Query-by-Committee [104] extends uncertainty sampling by training a committee of models on the same labeled data. Each model can vote when labeling each example, and the most informative example is considered to be the one where the most models disagree with each other. More formally, this approach minimizes the *version space*, which is the space of all possible classifiers that give the same classification results as (and are thus consistent with) the labeled data. The challenge is to train models that represent different regions of the version space and have some amount of disagreement. Various methods have been proposed [105], but there does not seem to be a clear winner. One general method is called query-by-bagging [106], which uses bagging as an ensemble learning algorithm and trains models on bootstrap samples. There is no general agreement on the best number of models to train, which is application-specific.

Both uncertainty sampling and query-by-committee focus on individual examples instead of the entire set of examples, they run into the danger of choosing examples that are outliers according to the example distribution. Density weighting [107] is a way to improve the above techniques by choosing instances that are not only uncertain or disagreeing, but also representative of the example distribution.

Decision Theoretic Approaches Another line of active learning performs decision-theoretic approaches. Decision theory is a framework for making decision under uncertainty using states and actions to optimize some objective function. In the context of active learning, the objective could be to choose an example that maximizes the estimated model accuracy [108]. Another possible objective is reducing generalization error [109], which is estimated as follows: if the measure is log loss, then the entropy of the predicted class distribution is considered the error rate; if the measure is 0-1 loss, the maximum probability among all classes is the error rate. Each example to label is chosen by taking a sample of the unlabeled data and choosing the example that minimizes the estimated error rate.

Regression Active learning techniques can also be extended to regression problems. For uncertainty sampling, instead of computing the entropy of classes, one can compute the output variance of the predictions and select the examples with the highest variance. Query-by-committee can also be extended to regression [110] by training a committee of models and selecting the examples where the variance among the

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

committee's predictions is the largest. This approach is said to work well when the bias of the models is small. Also, this approach is said to be robust to overspecification, which reduces the chance of overfitting.

Self and Active Learning Combined The data labeling techniques we consider are complementary to each other and can be used together. In fact, semi-supervised learning and active learning have a history of being used together [111]–[114], [148]. A key observation is that the two techniques solve opposite problems where semi-supervised learning finds the predictions with the highest confidence and adds them to the labeled examples while active learning finds the predictions with the lowest confidence (using uncertainty sampling, query-by-committee, or density-weighted method) and sends them for manual labeling.

There are various ways semi-supervised learning can be used with active learning. McCallum and Nigam [111] improves the Query-By-Committee (QBC) technique and combines it with Expectation-Maximization (EM), which effectively performs semi-supervised learning. Given a set of documents for training data, active learning is done by selecting the documents that are closer to others (and thus representative), but have committee disagreement, for labeling. In addition, the EM algorithm is used to further infer the rest of the labels. The active learning and EM can either be done separately or interleaved. Tomanek and Hahn [112] propose semi-supervised active learning (SeSAL) for sequence labeling tasks, which include POS tagging, chunking, or named entity recognition (NER). Here the examples are sequences of text. The idea is to use active learning for the subsequences that have the highest training utility within the selected sentences and use semi-supervised learning to automatically label the rest of the subsequences. The utility of a subsequence is highest when the current model is least confident about the labeling.

Zhou et al. [113] proposes the semi-supervised active image retrieval (SSAIR) approach where the focus is on image retrieval. SSAIR is inspired by the co-training method where initially two classifiers are trained from the labeled data. Then each learner passes the most relevant/irrelevant images to the other classifier. The classifiers are then retrained with the additional labels, and their results are combined. The images that still have low confidence are selected to be labeled by humans.

Zhu et al. [114] combines semi-supervised and active learning under a Gaussian random field model. The labeled and unlabeled examples are represented as vertices in a graph where edges are weighted by similarities between examples. This framework enables one to compute the next question that minimizes the expected generalization error efficiently for active learning. Once the new labels are added to the labeled data, semi-supervised learning is performed using harmonic functions.

3.2.2 Crowdsourcing

In comparison to active learning, the crowdsourcing techniques here are more focused on running tasks with many workers who are not necessarily labeling experts. As a result, workers may make mistakes, and there is a heavy literature [81]–[84], [115], [149], [150] on improving the interaction with workers, evaluating workers so they are reliable, reducing any bias that the workers may have, and aggregating the labeling results while resolving any ambiguities among them.

User Interaction A major challenge in user interaction is to effectively provide instructions to workers on how to perform the labeling. The traditional approach is to provide some guidelines for labeling to the workers up front and then let them make a best effort to follow them. However, the guidelines are often incomplete and do not cover all possible scenarios, leaving the workers in the dark. Revolt [115] is a system that attempts to fix this problem through collaborative crowdsourcing. Here workers work in three steps: *Voting* where workers vote just like in traditional labeling, *Explaining* where workers justify their rational for labeling, and *Categorize* where workers review explanations from other workers and tag any conflicting labels. This information can then be used to make post-hoc judgements of the label decision boundaries. Another approach is to provide better tools to assist workers to organize their concepts, which may evolve as more examples are labeled [117].

In addition, providing the right labeling interface is critical for workers to perform well. The challenge is that each application may have a different interface that works best. We will not cover all the possible applications, but instead illustrate a line of research for the problem of entity resolution where the goal is to find records in a database that refer to the same real-world entity. Here the label is whether two (or more) records are the same or not. Just for this problem, there is a line of research on providing the best interface for comparisons. CrowdER [118] provides two types of interfaces to compare records: pair-based and cluster-based. Qurk [50] uses a mapping interface where multiple records on one side are matched with records on the other side. Qurk uses a combination of comparison or rating tasks to accelerate labeling.

Quality control Controlling the quality of data labeling by the crowd is important because the workers may vary significantly in their abilities to provide labels. A simple way to ensure quality is to repeatedly label the same example using multiple workers and perhaps take a majority voting at the end. However, there are more sophisticated approaches as well. Get another label [119] and Crowdscreen [120] actively solicit labels while Karger et al. [121] passively collects data and runs the expectation maximization algorithm. Vox Populi [122] proposes techniques for pruning low-quality workers that can achieve better labeling quality without having to repeatedly label examples.

Scalability Scaling up crowdsourced labeling is another important challenge. While traditional active learning techniques were proposed for this purpose, more recently the data management community has started to apply systems techniques for further scaling the algorithms to large datasets. In particular, Mozafari et al. [116] proposes active learning algorithms that can run in parallel. One algorithm (called *Uncertainty*) selects examples that the current classifier is most uncertain about. A more sophisticated algorithm (called *MinExpError*) combines the current model's accuracy with the uncertainty. A key idea is the use of bootstrap the-

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

ory, which makes the algorithms applicable to any classifier and also enables embarassingly-parallel processing.

Regression In comparison to crowdsourcing research for classification tasks, less attention has been given to regression tasks. Marcus et al. [123] solves the problem of selectivity estimation in a crowdsourced database. The goal is to estimate the fraction of records that satisfy some property by asking workers questions.

3.3 Weak supervision

As machine learning is used in a wider range of applications, it is mostly the case that there is not enough labeled data. For example, in a smart factory setting, any new product will have no labels for training a model for quality control. As a result, weak supervision techniques [151]–[153] have become increasingly popular where the idea is to semi-automatically generate large quantities of labels that are not as accurate as manual labels, but good enough for the trained model to obtain a reasonably-high accuracy. This approach is especially useful when there are large amounts of data, and manual labeling becomes infeasible. In the next sections, we discuss the recently proposed data programming paradigm and fact extraction techniques.

3.3.1 Data Programming

As data labeling at scale becomes more important especially for deep learning applications, data programming [126] has been proposed as a solution for generating large amounts of weak labels using multiple labeling functions instead of individual labeling. Figure 5 illustrates how data programming can be used for Sally's smart factory application. A labeling function can be any computer program that either generates a label for an example or refrains to do so. For example, a labeling function that checks if a tweet has a positive sentiment may check if certain positive words appear in the text. Since a single labeling function by itself may not be accurate enough or not be able to generate labels for all examples, multiple labeling functions are implemented and combined into a generative model, which is then used to generate large amounts of weak labels with reasonable quality. Alternatively, voting methods like majority voting can be used to combine the labeling functions. Finally, a noise-aware discriminative model is trained on the weak labels. Data programming has been implemented in the state-of-the-art Snorkel system [127], which is becoming increasingly popular in the industry [128], [129].

Data programming has advantages both in terms of model accuracy and usability. A key observation for generating weak labels is that training a discriminative model on large amounts of weak labels may result in higher accuracy than training with fewer manual labels. In terms of usability, implementing labeling functions can be an intuitive process for humans compared to feature engineering in traditional machine learning [154].

What makes data programming effective is the way it combines multiple labeling functions into the generative model by fitting a probabilistic graphical model. A naïve approach to combine labeling functions is to take a majority vote. However, this approach cannot handle pathological cases where many labeling functions are near identical,

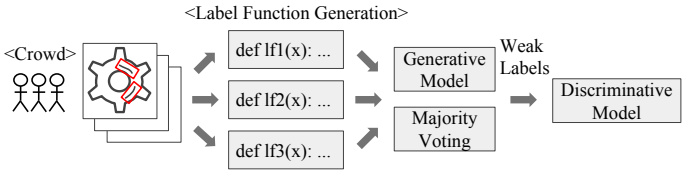


Fig. 5: A workflow of using data programming for a smart factory application. In this scenario, Sally is using crowdsourcing to annotate defects on component images. Next, the annotations can be automatically converted to labeling functions. Then the labeling functions are combined either into a generative model or using majority voting. Finally, the combined model generates weak labels that are used to train a discriminative model.

which defeats the purpose of majority voting. Instead, labeling functions that are more correlated with each other will have less influence on the predicted label. In addition, labeling functions that are outliers are also trained to have less influence in order to cancel out the noise. Theoretical analysis [3] shows that, if the labeling functions are reasonably accurate, then the predictions made by the generative model becomes arbitrarily close to the true labels.

Several systems have been developed for data programming. DeepDive [124] is a precursor of data programming and supports fast knowledge base construction on dark data using information extraction. DeepDive effectively uses humans to extract features, implement supervision rules using a declarative language (similar to labeling functions), and supports incremental maintenance of inference. DDLite [125] is the first system to use data programming and supports rapid prototyping of labeling functions through an interactive interface. The primary application is information extraction, and DDLite has been used to extract chemicals, diseases, and anatomical named entities. Compared to DeepDive, it has a simpler Python syntax and does not require complex setup involving databases. Snorkel [127], [155] is the most recent system for data programming. Compared to DDLite, Snorkel is a full-fledged product that is widely used in the industry. Snorkel enables users to use weak labels from all available weak label sources, supports any type of classifier, and provides rapid results in response to the user's input. More recently, Snorkel has been extended to solve massively multi-task learning [130].

Data programming is designed for classification and does not readily support regression. To make an extension to regression, the labeling functions must return real numbers instead of discrete values. In addition, the probabilistic graphical model used for training generative models must be able to return a continuous distribution of possible values for labels.

3.3.2 Fact Extraction

Another way to generate weak labels is use fact extraction. Knowledge bases contain facts that are extracted from various sources including the Web. A fact could describe an attribute of an entity (e.g., ⟨Germany, *capital*, Berlin⟩). The facts can be considered positively-labeled examples, which

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

can be used as seed labels for distant supervision [156] when generating weak labels. It is worth mentioning that fact extraction roots from the broader topic of information extraction where the goal is to extract structured data from the Web [157]. Early work include RoadRunner [139], which compares HTML pages to generates wrappers and KnowItAll [140], which uses extraction rules and a search engine to identify and rank facts. Since then, the subsequent works have become more sophisticated and also attempt to organize the extract facts into knowledge bases.

We now focus on fact extraction techniques for knowledge bases. If precision is critically important, then manual curation should be part of the knowledge base construction as in Freebase [131] and Google Knowledge Graph. Otherwise, the extraction techniques depend on the data source. YAGO [132], [133] extracts facts from Wikipedia using classes in WordNet. Ollie [134], ReVerb [135], and ReNoun [136] and open information extraction systems that apply patterns to Web text. Knowledge Vault [137] also extracts from Web content, but combines facts from text, tabular data, page structure, and human annotations. Biperpedia [138] extracts the attributes of entities from a query stream and Web text.

The Never-Ending Language Learner (NELL) system [141], [142] continuously extracts structured information from the unstructured Web and constructs a knowledge base that consists of entities and relationships. Initially, NELL starts with a seed ontology, which contains entities of classes (e.g., person, fruit, and emotion) and relationships among the entities (e.g., $playsOnTeam(athlete, sportsTeam)$ and $playsInstrument(musician, instrument)$). NELL analyzes hundreds of millions of Web pages and identifies new entities in the given classes as well as entity pairs of the relationships by matching patterns on their surrounding phrases. The resulting entities and relationships can then be used as the next training data for constructing even more patterns. NELL has been collecting facts continuously since 2010. The extraction techniques can be viewed as distant supervision generating weak labels.

4 Using Existing Data and Models

An alternative approach to acquiring new data and labeling it is to improve the labeling of any existing datasets or improving the model training. This approach makes sense for a number of scenarios. First, it may be difficult to find new datasets because the application is too novel or non-trivial for others to have produced datasets. Second, simply adding more data may not significantly improve the model's accuracy anymore. In this case, re-labeling or cleaning the existing data may be the faster way to increase the accuracy. Alternatively, the model training can be made more robust to noise and bias, or the model can be trained from an existing model using transfer learning techniques. In the following sections, we explore techniques for improving existing labels and improving existing models. The techniques are summarized in Table 3.

Task	Techniques
Improve Data	Data Cleaning [158]–[166]
	Re-labeling [119]
Improve Model	Robust Against Noise [167]–[171]
	Transfer Learning [172]–[178]

TABLE 3: A classification of techniques for improving existing data and models.

4.1 Improving Existing Data

A major problem in machine learning is that the data can be noisy and the labels incorrect. This problem occurs frequently in practice, so production machine learning platforms like TensorFlow Extended (TFX) [179] have separate components [180] to reduce data errors as much as possible though analysis and validation. In case the labels are also noisy, re-labeling the examples becomes necessary as well. We explore recent advances in data cleaning with a focus on machine learning and then techniques for re-labeling.

4.1.1 Data Cleaning

It is common for the data itself to be noisy. For example, some values may be out of range (e.g., a latitude value is beyond [-90, 90]) or use different units by mistake (e.g., some intervals are in hours while other are in minutes). There is a heavy literature on various integrity constraints (e.g., domain constraints, referential integrity constraints, and functional dependencies) that can improve data quality as well. HoloClean [158] is a state-of-art data cleaning system that uses quality rules, value correlations, and reference data to build a probabilistic model that captures how the data was generated. HoloClean then generates a probabilistic program for repairing the data. In addition, various interactive data cleaning tools [162]–[165] have been proposed to convert data into a better form for machine learning.

An interesting line of recent work is cleaning techniques with the explicit intention of improving machine learning results. ActiveClean [159] is a model training framework that iteratively suggests samples of data to clean based on how much the cleaning improves the model accuracy and the likelihood that the data is dirty. An analyst can then perform transformations and filtering to clean each sample. ActiveClean treats the training and cleaning as a form of stochastic gradiant descent and uses convex-loss models (SVMs, linear and logistic regression) to guarantee global solutions for clean models. BoostClean [160] solves an important class of inconsistencies where an attribute value is outside an allowed domain. BoostClean takes as input a dataset and a set of functions that can detect these errors and repair functions that can fix them. Each pair of detection and repair functions can produce a new model trained on the cleaned data. BoostClean uses statistical boosting to find the best ensemble of pairs that maximize the final model's accuracy. Recently, TARS [161] was proposed to solve the problem of cleaning crowdsourced labels using oracles. TARS provides two pieces of advice. First, given test data with noisy labels, it uses an estimation technique to predict how well the model may perform on the true labels. The estimation is shown to be unbiased, and confidence intervals are computed to bound the error. Second, given

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

training data with noisy labels, TARS determines which examples to send to an oracle in order to maximize the expected model improvement of cleaning each noisy label. More recently, MLCLEAN [166] has been proposed to integrate three data operations: traditional data cleaning, model unfairness mitigation where the goal is to remove data bias that causes model fairness, and data sanitization where the goal is to remove data poisoning.

4.1.2 Re-labeling

Trained models are only as good as their training data, and it is important to obtain high quality data labels. Simply labeling more data may not improve the model accuracy further. Indeed, Sheng et al. [119] shows that, if the labels are noisy, then the model accuracy plateaus from some point and does not increase further, no matter how many more labeling is done. The solution is to improve the quality of existing labels. The authors show that repeated labeling using workers of certain individual qualities can significantly improve model accuracy where a straightforward round robin approach already give substantial improvements, and being more selective in labeling gives even better results.

4.2 Improving Models

In addition to improving the data, there are also ways to improve the model training itself. Making the model training more robust against noise or bias is an active area of research. Another popular approach is to use transfer learning where previously-trained models are used as a starting point to train the current model.

4.2.1 Robust Against Noise and Bias

A common scenario in machine learning is that there is a large number of noisy or even adversarial labels and a relatively smaller number of clean labels. Simply discarding the noisy labels will result in reduced training data, which is not desirable for complex models. Hence, there has been extensive research (see the survey [181]) on how to make the model training still use noisy labels by becoming more robust. For specific techniques, Xiao et al. [167] propose a general framework for training convolutional neural networks on images with a small number of clean labels and many noisy labels. The idea is to model the relationships between images, class labels, and label noises with a probabilistic graphical model and integrate it into the model training. Label noise is categorized into two types: confusing noise, which is caused by confusing content in the images, and pure random noise, which is caused by technical bugs like mismatches between images and their surrounding text. The true labels and noise types are treated as latent variables, and an EM algorithm is used for inference. Webly supervised learning [168] is a technique for training a convolutional neural network on clean and noisy images on the Web. First, the model is trained on top-ranked images from search engines, which tend to be clean because they are highly-ranked, but also biased in the sense that objects tends to be centered in the image with a clean background. Then relationships are discovered among the clean images, which are then used to adapt the model to more noisier images that are harder to classify. This method suggests that it is worth training on easy and hard data separately.

Goodfellow et al. [171] take a different approach where they explain why machine learning models including neural networks may misclassify adversarial examples. While previous research attempts to explain this phenomenon by focusing on nonlinearity and overfitting, the authors show that it is the model's linear behavior in high-dimensional spaces that makes it vulnerable. That is, making many small changes on the features of an example can result in a large change to the output prediction. As a result, generating large amounts of adversarial examples becomes easier using linear perturbation.

Even if the labels themselves are clean, it may be the case that the labels are imbalanced. $SMOTE$ [170] performs over-sampling for minority classes that need more examples. Simply replicating examples may lead to overfitting, so the over-sampling is done by generating synthetic examples using the minority examples and their nearest neighbors. He and Garcia [169] provide a comprehensive survey on learning from imbalanced data.

4.2.2 Transfer Learning

Transfer learning is a popular approach for training models when there is not enough training data or time to train from scratch. A common technique is to start from an existing model that is well trained (also called a *source task*), one can incrementally train a new model (a *target task*) that already performs well. For example, a convolutional neural networks like AlexNet [182] and VGGNet [183] can be used to train a model for a different, but related vision problem. Recently, Google announced TensorFlow Hub [174], which enables users to easily re-use an existing model to train an accurate model, even with a small dataset. Also, Google Cloud AutoML [6] provides transfer learning as a service. From a data management perspective, an interesting question is how these existing tools can be extended to index the metadata of models and provide search as a service, just like for datasets. The metadata for models may be quite different than metadata for data because one needs to determine if a model can be used for transfer learning in her own application. In addition to using pre-trained models, another popular technique mainly used in Computer Vision is few-shot learning [175] where the goal is to extend existing models to handle new classes using zero or more examples. Since transfer learning is primarily a machine learning topic that does not significantly involve data management, we only summarize the high-level ideas based on surveys [172], [173]. There are studies of transfer learning techniques in the context of NLP [176], Computer Vision [177], and deep learning [184] as well.

An early survey of transfer learning [172] identifies three main research issues in transfer learning: what to transfer, how to transfer, and when to transfer. That is, we need to decide what part of knowledge can be transferred, what methods should be used to transfer the knowledge, and whether transferring this knowledge is appropriate and does not have any negative effect. *Inductive transfer learning* is used when the source task and target task are different while the two domains may or may not be the same. Here a task can be categorizing a document while a domain could

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

be a set of university webpages to categorize. *Transductive transfer learning* is used when the source and target tasks are the same, but the domains are different. *Unsupervised transfer learning* is similar to inductive transfer learning where the source and target tasks are different, but uses unsupervised learning tasks like clustering and dimensionality reduction.

The three approaches above can also be divided based on what to transfer. *Instance-based transfer learning* assumes that the examples of the source can be re-used in the target by re-weighting them. *Feature-representation transfer learning* assumes that the features that represent the data of the source task can be used to represent the data of the target task. *Parameter transfer learning* assumes that the source and target tasks share some parameters or prior distributions that can be re-used. *Relational knowledge transfer learning* assumes that certain relationships within the data of the source task can be re-used in the target task.

More recent surveys [173], [178] classify most of the traditional transfer learning techniques as *homogeneous transfer learning* where the feature spaces of the source and target tasks are the same. In addition, the surveys identify a relatively new class of techniques called *heterogeneous transfer learning* where the feature spaces are different, but the source and target examples are extracted from the same domain. Heterogeneous transfer learning largely falls into two categories: asymmetric and symmetric transformation. In an asymmetric approach, features of the source task are transformed to the features of the target task. In a symmetric approach, the assumption is that there is a common latent feature space that unifies the source and target features. Transfer learning has been successfully used in many applications including text sentiment analysis, image classification, human activity classification, software defect detection, and multi-language text classification.

5 Putting Everything Together

We now return to Sally's scenario and provide an end-to-end guideline for data collection (summarized as the workflow in Figure 2). If there is no or little data to start with then Sally would need to acquire datasets. She can either search for relevant datasets either on the Web or within the company data lake, or decide to generate a dataset herself by installing camera equipment for taking photos of the products within the factory. If the products also had some metadata, Sally could also augment that data with external information about the product.

Once the data is available, then Sally can choose among the labeling techniques using the categories discussed in Section 3. If there are enough existing labels, then self labeling using semi-supervised learning is an attractive option. There are many variants of self labeling depending on the assumptions on the model training as we studied. If there are not enough labels, Sally can decide to generate some using the crowd-based techniques using a budget. If there are only a few experts available for labeling, active learning may be the right choice, assuming that the important examples that influence the model can be narrowed down. If there are many workers who do not necessarily have expertise, general crowdsourcing methods can be used. If Sally does not have enough budget for crowd-based methods or if it is simply not worth the cost, and if the model training can tolerate weak labels, then weak supervision techniques like data programming and label propagation can be used.

If Sally has existing labels, she may also want to make sure whether they can be improved in quality. If the data is noisy or biased, then the various data cleaning techniques can be used. If there are existing models for product quality through tools like TensorFlow Hub [174], they can be used to further improve the model using transfer learning.

Through our experience, we also realize that it is not always easy to determine if there is enough data and labels. For example, even if the dataset is small or there are few labels, as long as the distribution of data is easy to learn, then automatic approaches like semi-supervised learning will do the job better than manual approaches like active learning. Another hard-to-measure factor is the amount of human effort needed. When comparing active learning versus data programming, we need to compare the tasks of labeling examples and implementing labeling functions, which are quite different. Depending on the application, implementing a program on examples can range from trivial (e.g., look for certain keywords) to almost impossible (e.g., general object detection). Hence, even if data programming is an attractive option, one must determine the actual effort of programming, which cannot be determined with a few yes or no questions.

Another thing to keep in mind is how the labeling techniques tradeoff accuracy and scalability. Manual labeling is obviously the most accurate, but least scalable. Active learning scales better than the manual approach, but is still limited to how fast humans can label. Data programming produces weak labels, which tend to have lower accuracy than manual labels. On the other hand, data programming can scale better than active learning assuming that the initial cost of implementing labeling functions and debugging them is reasonable. Semi-supervised learning obviously scales the best with automatic labeling. The labeling accuracy depends on the accuracy of the model trained on existing labels. Combining self labeling with active learning is a good example of taking the best of both worlds.

6 Future Research Challenges

Although data collection was traditionally a topic in the machine learning, as the amount of training data is increasing, data management research is becoming just as relevant, and we are observing a convergence of the two disciplines. As such, there needs to be more awareness on how the research landscape will evolve for both communities and more effort to better integrate the techniques.

Data Evaluation An open question is how to evaluate whether the right data was collected with sufficient quantity. First, it may not be clear if we have found the best datasets for a machine learning task and whether the amount of data is enough to train a model with sufficient accuracy. In some cases, there may be too many datasets, and simply collecting and integrating all of them may have a negative affect on model training. As a result, selecting the right datasets becomes an important problem. Moreover, if the datasets are dynamic (e.g., they are streams of signals from sensors) and change in quality, then the choice of datasets

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

may have to change dynamically as well. Second, many data discovery tools rely on dataset owners to annotate their datasets for better discovery, but more automatic techniques for understanding and extracting metadata from the data are needed.

While most of the data collection work assumes that the model training comes after the data collection, another important avenue is to augment or improve the data based on how the model performs. While there is a heavy literature on model interpretation [185], [186], it is not clear how to address feedback on the data level. In the model fairness literature [187], one approach to reducing unfairness is to fix the data. In data cleaning, ActiveClean and BoostClean are interesting approaches for fixing the data to improve model accuracy. A key challenge is analyzing the model, which becomes harder as models become more complicated.

Performance Tradeoff While traditional labeling techniques focus on accuracy, there is a recent push towards generating large amounts of weak labels. We need to better understand the tradeoffs of accuracy versus scalability to make informed decisions on which approach to use when. For example, simply having more weak labels does not necessarily mean the model's accuracy will eventually reach a perfect accuracy. At some point, it may be worth investing in humans or using transfer learning to make additional improvements. Such decisions can be made through some trial and error, but an interesting question is whether there is a more systematic way to do such evaluations.

Crowdsourcing Despite the many efforts in crowdsourcing, leveraging humans is still a non-trivial task. Dealing with humans involves designing the right tasks and interfaces, ensuring that the worker quality is good enough, and setting the right price for tasks. The recent data programming paradigm introduces a new set of challenges where workers now have to implement labeling functions instead of providing labels themselves. One idea is to improve the quality of such collaborative programming by making the programming of labeling functions drastically easier, say by introducing libraries or templates for programming.

Empirical comparison of techniques Although we showed a flowchart on when to use which techniques, it is far from complete, as many factors are application-specific and can only be determined by looking at the data and application. For example, if the model training can be done with a small number of labels, then we may not have to perform data labeling using crowdsourcing. In addition, the estimated human efforts in labeling and data programming may not follow any theoretical model in practice. For example, humans may find programming for certain applications much more difficult and time-consuming than other applications depending on their expertise. Hence, there needs to be more empirical research on the effectiveness of the techniques.

Generalizing and integrating techniques We observed that many data collection techniques were application or data type specific and were often small parts of a larger research. As machine learning becomes widely used in just about any application, there needs to be more effort in generalizing the techniques to other problems. In data labeling, most of the research effort has been focused on classification tasks and much less on regression tasks. An interesting question is which classification techniques can also be extended to regression. It is also worth exploring if application-specific techniques can be generalized further. For example, the NELL system continuously extracts facts from the Web indefinitely. This idea can possibly be applied to collecting any type of data from any source, although the technical details may differ. Finally, given the variety of techniques for data collection, there needs to be more research on end-to-end solutions that combine techniques for data acquisition, data labeling, and improvements of existing data and models.

7 Conclusion

As machine learning becomes more widely used, it becomes more important to acquire large amounts of data and label data, especially for state-of-the-art neural networks. Traditionally, the machine learning, natural language processing, and computer vision communities has contributed to this problem – primarily on data labeling techniques including semi-supervised learning and active learning. Recently, in the era of Big data, the data management community is also contributing to numerous subproblems in data acquisition, data labeling, and improvement of existing data. In this survey, we have investigated the research landscape of how all these technique complement each other and have provided guidelines on deciding which technique can be used when. Finally, we have discussed interesting data collection challenges that remain to be addressed. In the future, we expect the integration of Big data and AI to happen not only in data collection, but in all aspects of machine learning.

Acknowledgments

This research was supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), by SK Telecom, and by a Google AI Focused Research Award.

References

[1] "Deep learning for detection of diabetic eye disease," https://research.googleblog.com/2016/11/deep-learning-for-detection-of-diabetic.html.
[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.
[3] S. H. Bach, B. D. He, A. Ratner, and C. Ré, "Learning the structure of generative models without labeled data," in *ICML*, 2017, pp. 273–282.
[4] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data lifecycle challenges in production machine learning: A survey," *SIGMOD Rec.*, vol. 47, no. 2, pp. 17–28, Jun. 2018.
[5] ——, "Data management challenges in production machine learning," in *SIGMOD*, 2017, pp. 1723–1726.
[6] "Google cloud automl," https://cloud.google.com/automl/.
[7] "Microsoft custom vision," https://azure.microsoft.com/en-us/services/cognitive-services/custom-vision-service/.
[8] "Amazon sagemaker," https://aws.amazon.com/sagemaker/.
[9] A. Bhardwaj, A. Deshpande, A. J. Elmore, D. Karger, S. Madden, A. Parameswaran, H. Subramanyam, E. Wu, and R. Zhang, "Collaborative data analytics with datahub," *PVLDB*, vol. 8, no. 12, pp. 1916–1919, Aug. 2015.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

[10] A. P. Bhardwaj, S. Bhattacherjee, A. Chavan, A. Deshpande, A. J. Elmore, S. Madden, and A. G. Parameswaran, "Datahub: Collaborative data science & dataset version management at scale," in *CIDR*, 2015.

[11] S. Bhattacherjee, A. Chavan, S. Huang, A. Deshpande, and A. Parameswaran, "Principles of dataset versioning: Exploring the recreation/storage tradeoff," *PVLDB*, vol. 8, no. 12, pp. 1346–1357, Aug. 2015.

[12] A. Y. Halevy, "Data publishing and sharing using fusion tables," in *CIDR*, 2013.

[13] H. Gonzalez, A. Y. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, and W. Shen, "Google fusion tables: data management, integration and collaboration in the cloud," in *SoCC*, 2010, pp. 175–180.

[14] H. Gonzalez, A. Y. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, W. Shen, and J. Goldberg-Kidon, "Google fusion tables: web-centered data management and collaboration," in *SIGMOD*, 2010, pp. 1061–1066.

[15] "Ckan," http://ckan.org.

[16] "Quandl," https://www.quandl.com.

[17] "Datamarket," https://datamarket.com.

[18] "Kaggle," https://www.kaggle.com/.

[19] I. G. Terrizzano, P. M. Schwarz, M. Roth, and J. E. Colino, "Data wrangling: The challenging yourney from the wild to the lake," in *CIDR*, 2015.

[20] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang, "Goods: Organizing google's datasets," in *SIGMOD*, 2016, pp. 795–806.

[21] R. Castro Fernandez, D. Deng, E. Mansour, A. A. Qahtan, W. Tao, Z. Abedjan, A. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang, "A demo of the data civilizer system," in *SIGMOD*, 2017, pp. 1639–1642.

[22] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang, "The data civilizer system," in *CIDR*, 2017.

[23] Y. Gao, S. Huang, and A. G. Parameswaran, "Navigating the data lake with DATAMARAN: automatically extracting structure from log datasets," in *SIGMOD*, 2018, pp. 943–958.

[24] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: Exploring the power of tables on the web," *PVLDB*, vol. 1, no. 1, pp. 538–549, 2008.

[25] M. J. Cafarella, A. Y. Halevy, H. Lee, J. Madhavan, C. Yu, D. Z. Wang, and E. Wu, "Ten years of webtables," *PVLDB*, vol. 11, no. 12, pp. 2140–2149, 2018.

[26] "Google dataset search," https://www.blog.google/products/search/making-it-easier-discover-datasets/.

[27] H. Elmeleegy, J. Madhavan, and A. Halevy, "Harvesting relational tables from lists on the web," *The VLDB Journal*, vol. 20, no. 2, pp. 209–226, Apr. 2011.

[28] X. Chu, Y. He, K. Chakrabarti, and K. Ganjam, "Tegra: Table extraction by global record alignment," in *SIGMOD*, 2015, pp. 1713–1728.

[29] K. Chakrabarti, S. Chaudhuri, Z. Chen, K. Ganjam, and Y. He, "Data services leveraging bing's data assets," *IEEE Data Eng. Bull.*, vol. 39, no. 3, pp. 15–28, 2016.

[30] M. J. Cafarella, A. Halevy, and N. Khoussainova, "Data integration for the relational web," *PVLDB*, vol. 2, no. 1, pp. 1090–1101, Aug. 2009.

[31] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri, "Infogather: Entity augmentation and attribute discovery by holistic matching with web tables," in *SIGMOD*, 2012, pp. 97–108.

[32] N. N. Dalvi, R. Kumar, and M. A. Soliman, "Automatic wrappers for large scale web extraction," *PVLDB*, vol. 4, no. 4, pp. 219–230, 2011.

[33] P. Bohannon, N. N. Dalvi, Y. Filmus, N. Jacoby, S. S. Keerthi, and A. Kirpal, "Automatic web-scale information extraction," in *SIGMOD*, 2012, pp. 609–612.

[34] R. Baumgartner, W. Gatterbauer, and G. Gottlob, "Web data extraction system," in *Encyclopedia of Database Systems, Second Edition*, 2018.

[35] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[36] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014, pp. 1532–1543.

[37] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[38] A. Kumar, J. Naughton, J. M. Patel, and X. Zhu, "To join or not to join?: Thinking twice about joins before feature selection," in *SIGMOD*, 2016, pp. 19–34.

[39] V. Shah, A. Kumar, and X. Zhu, "Are key-foreign key joins safe to avoid when learning high-capacity classifiers?" *PVLDB*, vol. 11, no. 3, pp. 366–379, Nov. 2017.

[40] M. Stonebraker and I. F. Ilyas, "Data integration: The current status and the way forward," *IEEE Data Eng. Bull.*, vol. 41, no. 2, pp. 3–9, 2018.

[41] A. Doan, A. Y. Halevy, and Z. G. Ives, *Principles of Data Integration*. Morgan Kaufmann, 2012.

[42] S. Li, L. Chen, and A. Kumar, "Enabling and optimizing non-linear feature interactions in factorized linear algebra," in *SIGMOD*, 2019, pp. 1571–1588.

[43] L. Chen, A. Kumar, J. F. Naughton, and J. M. Patel, "Towards linear algebra over normalized data," *PVLDB*, vol. 10, no. 11, pp. 1214–1225, 2017.

[44] A. Kumar, J. F. Naughton, J. M. Patel, and X. Zhu, "To join or not to join?: Thinking twice about joins before feature selection," in *SIGMOD*, 2016, pp. 19–34.

[45] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller, "Turkit: Human computation algorithms on mechanical turk," in *UIST*, 2010, pp. 57–66.

[46] D. W. Barowy, C. Curtsinger, E. D. Berger, and A. McGregor, "Automan: A platform for integrating human-based and digital computation," in *OOPSLA*, 2012, pp. 639–654.

[47] S. Ahmad, A. Battle, Z. Malkani, and S. Kamvar, "The jabberwocky programming environment for structured social computing," in *UIST*, 2011, pp. 53–64.

[48] H. Park, R. Pang, A. G. Parameswaran, H. Garcia-Molina, N. Polyzotis, and J. Widom, "Deco: A system for declarative crowdsourcing," *PVLDB*, vol. 5, no. 12, pp. 1990–1993, 2012.

[49] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "Crowddb: Answering queries with crowdsourcing," in *SIGMOD*, 2011, pp. 61–72.

[50] A. Marcus, E. Wu, S. Madden, and R. C. Miller, "Crowdsourced databases: Query processing with people," in *CIDR*, 2011, pp. 211–214.

[51] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W. C. Tan, "Asking the right questions in crowd data sourcing," in *ICDE*, 2012, pp. 1261–1264.

[52] M. J. Franklin, B. Trushkowsky, P. Sarkar, and T. Kraska, "Crowdsourced enumeration queries," in *ICDE*, 2013, pp. 673–684.

[53] H. Park and J. Widom, "Crowdfill: collecting structured data from the crowd," in *SIGMOD*, 2014, pp. 577–588.

[54] V. Crescenzi, P. Merialdo, and D. Qiu, "Crowdsourcing large scale wrapper inference," vol. 33, pp. 1–28, 2014.

[55] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu, "Data curation at scale: The data tamer system," in *CIDR*, 2013.

[56] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu, "Corleone: Hands-off crowdsourcing for entity matching," in *SIGMOD*, 2014, pp. 601–612.

[57] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[58] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," in *MLHC*, 2017, pp. 286–305.

[59] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *PVLDB*, vol. 11, no. 10, pp. 1071–1083, 2018.

[60] L. Xu and K. Veeramachaneni, "Synthesizing tabular data using generative adversarial networks," *CoRR*, vol. abs/1811.11264, 2018.

[61] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2017.

[62] K. Kurach, M. Lucic, X. Zhai, M. Michalski, and S. Gelly, "The GAN landscape: Losses, architectures, regularization, and normalization," *CoRR*, vol. abs/1807.04720, 2018.

[63] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *CoRR*, vol. abs/1805.09501, 2018.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

[64] A. J. Ratner, H. R. Ehrenberg, Z. Hussain, J. Dunnmon, and C. Ré, "Learning to compose domain-specific transformations for data augmentation," in *NIPS*, 2017, pp. 3239–3249.
[65] X. Peng, B. Sun, K. Ali, and K. Saenko, "Learning deep object detectors from 3d models," in *ICCV*, 2015, pp. 1278–1286.
[66] S. Kim, G. Choe, B. Ahn, and I. Kweon, "Deep representation of industrial components using simulated images," in *ICRA*, 2017, pp. 2003–2010.
[67] T. Oh, R. Jaroensri, C. Kim, M. A. Elgharib, F. Durand, W. T. Freeman, and W. Matusik, "Learning-based video motion magnification," *CoRR*, vol. abs/1804.02684, 2018.
[68] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," *CoRR*, vol. abs/1406.2227, 2014.
[69] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *CVPR*, June 2016.
[70] Y. Xia, X. Cao, F. Wen, and J. Sun, "Well begun is half done: Generating high-quality seeds for automatic image dataset construction from web," in *ECCV*, 2014, pp. 387–400.
[71] Y. Bai, K. Yang, W. Yu, C. Xu, W. Ma, and T. Zhao, "Automatic image dataset construction from click-through logs using deep neural network," in *MM*, 2015, pp. 441–450.
[72] J. Mallinson, R. Sennrich, and M. Lapata, "Paraphrasing revisited with neural machine translation," in *EACL*. Association for Computational Linguistics, 2017, pp. 881–893.
[73] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer, "Adversarial example generation with syntactically controlled paraphrase networks," *CoRR*, vol. abs/1804.06059, 2018.
[74] M. T. Ribeiro, S. Singh, and C. Guestrin, "Semantically equivalent adversarial rules for debugging nlp models," in *ACL*, 2018.
[75] A. Y. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang, "Managing google's data lake: an overview of the goods system," *IEEE Data Eng. Bull.*, vol. 39, no. 3, pp. 5–14, 2016.
[76] J. X. Yu, L. Qin, and L. Chang, "Keyword search in relational databases: A survey," *IEEE Data Eng. Bull.*, vol. 33, no. 1, pp. 67–78, 2010.
[77] S. Chaudhuri and G. Das, "Keyword querying and ranking in databases," *PVLDB*, vol. 2, no. 2, pp. 1658–1659, 2009.
[78] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker, "Aurum: A data discovery system," in *ICDE*, 2018, pp. 1001–1012.
[79] R. C. Fernandez, E. Mansour, A. A. Qahtan, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang, "Seeping semantics: Linking datasets using word embeddings for data discovery," in *ICDE*, 2018, pp. 989–1000.
[80] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014, pp. 1188–1196.
[81] "Crowdsourced data management: Industry and academic perspectives," *Foundations and Trends in Databases*, vol. 6, 2015.
[82] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar, "Quality control in crowdsourcing systems: Issues and directions," *IEEE Internet Computing*, vol. 17, no. 2, pp. 76–81, March 2013.
[83] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, and M. Allahbakhsh, "Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 7:1–7:40, Jan. 2018.
[84] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," *IEEE TKDE*, vol. 28, no. 9, pp. 2296–2319, Sept 2016.
[85] "Amazon mechanical turk," https://www.mturk.com.
[86] J. Kim, S. Sterman, A. A. B. Cohen, and M. S. Bernstein, "Mechanical novel: Crowdsourcing complex work through reflection and revision," in *CSCW*, 2017, pp. 233–245.
[87] N. Salehi, J. Teevan, S. T. Iqbal, and E. Kamar, "Communicating context to the crowd for complex writing tasks," in *CSCW*, 2017, pp. 1890–1901.
[88] H. Garcia-Molina, M. Joglekar, A. Marcus, A. Parameswaran, and V. Verroios, "Challenges in data crowdsourcing," *IEEE TKDE*, vol. 28, no. 4, pp. 901–911, Apr. 2016.
[89] Y. Amsterdamer and T. Milo, "Foundations of crowd data sourcing," *SIGMOD Rec.*, vol. 43, no. 4, pp. 5–14, Feb. 2015.
[90] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *DSAA*, 2016, pp. 399–410.
[91] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[92] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *ACL*, 1995, pp. 189–196.
[93] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE TKDE*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.
[94] Y. Zhou and S. A. Goldman, "Democratic co-learning," in *IEEE ICTAI*, 2004, pp. 594–602.
[95] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *COLT*, 1998, pp. 92–100.
[96] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," *Knowl. Inf. Syst.*, vol. 42, no. 2, pp. 245–284, 2015.
[97] U. Brefeld, T. Gärtner, T. Scheffer, and S. Wrobel, "Efficient co-regularised least squares regression," in *ICML*, 2006, pp. 137–144.
[98] V. Sindhwani and P. Niyogi, "A co-regularized approach to semi-supervised learning with multiple views," in *Proceedings of the ICML Workshop on Learning with Multiple Views*, 2005.
[99] Z.-H. Zhou and M. Li, "Semi-supervised regression with co-training," in *IJCAI*, 2005, pp. 908–913.
[100] S. Ravi and Q. Diao, "Large scale distributed semi-supervised learning using streaming approximation," in *AISTATS*, 2016, pp. 519–528.
[101] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003, pp. 912–919.
[102] P. P. Talukdar and W. W. Cohen, "Scaling graph-based semi supervised learning to large number of labels using count-min sketch," in *AISTATS*, 2014, pp. 940–947.
[103] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR*, 1994, pp. 3–12.
[104] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *COLT*, 1992, pp. 287–294.
[105] B. Settles, *Active Learning*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.
[106] N. Abe and H. Mamitsuka, "Query learning strategies using boosting and bagging," in *ICML*, 1998, pp. 1–9.
[107] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *EMNLP*, 2008, pp. 1070–1079.
[108] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *NIPS*, 2007, pp. 1289–1296.
[109] N. Roy and A. McCallum, "Toward optimal active learning through sampling estimation of error reduction," in *ICML*, 2001, pp. 441–448.
[110] R. Burbidge, J. J. Rowland, and R. D. King, "Active learning for regression based on query by committee," in *IDEAL*, 2007, pp. 209–218.
[111] A. McCallum and K. Nigam, "Employing em and pool-based active learning for text classification," in *ICML*, 1998, pp. 350–358.
[112] K. Tomanek and U. Hahn, "Semi-supervised active learning for sequence labeling," in *ACL*, 2009, pp. 1039–1047.
[113] Z.-H. Zhou, K.-J. Chen, and Y. Jiang, "Exploiting unlabeled data in content-based image retrieval," in *ECML*, 2004, pp. 525–536.
[114] X. Zhu, J. Lafferty, and Z. Ghahramani, "Combining active learning and semi-supervised learning using gaussian fields and harmonic functions," in *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003, pp. 58–65.
[115] J. C. Chang, S. Amershi, and E. Kamar, "Revolt: Collaborative crowdsourcing for labeling machine learning datasets," in *CHI*, 2017, pp. 2334–2346.
[116] B. Mozafari, P. Sarkar, M. Franklin, M. Jordan, and S. Madden, "Scaling up crowd-sourcing to very large datasets: A case for active learning," *PVLDB*, vol. 8, no. 2, pp. 125–136, Oct. 2014.
[117] T. Kulesza, S. Amershi, R. Caruana, D. Fisher, and D. X. Charles, "Structured labeling for facilitating concept evolution in machine learning," in *CHI*, 2014, pp. 3075–3084.
[118] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "Crowder: Crowdsourcing entity resolution," *PVLDB*, vol. 5, no. 11, pp. 1483–1494, 2012.
[119] V. S. Sheng, F. Provost, and P. G. Ipeirotis, "Get another label? improving data quality and data mining using multiple, noisy labelers," in *KDD*, 2008, pp. 614–622.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

[120] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom, "Crowdscreen: algorithms for filtering data with humans," in *SIGMOD*, 2012, pp. 361–372.

[121] D. R. Karger, S. Oh, and D. Shah, "Iterative learning for reliable crowdsourcing systems," in *NIPS*, 2011, pp. 1953–1961.

[122] O. Dekel and O. Shamir, "Vox populi: Collecting high-quality labels from a crowd," in *COLT*, 2009.

[123] A. Marcus, D. R. Karger, S. Madden, R. Miller, and S. Oh, "Counting with the crowd," *PVLDB*, vol. 6, no. 2, pp. 109–120, 2012.

[124] C. Zhang, "Deepdive: A data management system for automatic knowledge base construction," Ph.D. dissertation, 2015.

[125] H. R. Ehrenberg, J. Shin, A. J. Ratner, J. A. Fries, and C. Ré, "Data programming with ddlite: putting humans in a different part of the loop," in *HILDA@SIGMOD*, 2016, p. 13.

[126] A. J. Ratner, C. D. Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," in *NIPS*, 2016, pp. 3567–3575.

[127] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *PVLDB*, vol. 11, no. 3, pp. 269–282, Nov. 2017.

[128] S. H. Bach, D. Rodriguez, Y. Liu, C. Luo, H. Shao, C. Xia, S. Sen, A. Ratner, B. Hancock, H. Alborzi, R. Kuchhal, C. Ré, and R. Malkin, "Snorkel drybell: A case study in deploying weak supervision at industrial scale," in *SIGMOD*, 2019, pp. 362–375.

[129] E. Bringer, A. Israeli, Y. Shoham, A. Ratner, and C. Ré, "Osprey: Weak supervision of imbalanced extraction problems without code," in *DEEM@SIGMOD*, 2019, pp. 4:1–4:11.

[130] A. Ratner, B. Hancock, J. Dunnmon, R. Goldman, and C. Ré, "Snorkel metal: Weak supervision for multi-task learning," in *DEEM@SIGMOD*, 2018, pp. 3:1–3:4.

[131] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *SIGMOD*, 2008, pp. 1247–1250.

[132] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," in *WWW*, 2007, pp. 697–706.

[133] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "YAGO3: A knowledge base from multilingual wikipedias," in *CIDR*, 2015.

[134] Mausam, M. Schmitz, R. Bart, S. Soderland, and O. Etzioni, "Open language learning for information extraction," in *EMNLP-CoNLL*, 2012, pp. 523–534.

[135] N. Sawadsky, G. C. Murphy, and R. Jiresal, "Reverb: Recommending code-related web pages," in *ICSE*, 2013, pp. 812–821.

[136] M. Yahya, S. Whang, R. Gupta, and A. Y. Halevy, "Renoun: Fact extraction for nominal attributes," in *EMNLP*, 2014, pp. 325–335.

[137] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *KDD*, 2014, pp. 601–610.

[138] R. Gupta, A. Halevy, X. Wang, S. E. Whang, and F. Wu, "Biperpedia: An ontology for search applications," *PVLDB*, vol. 7, no. 7, pp. 505–516, Mar. 2014.

[139] V. Crescenzi, G. Mecca, and P. Merialdo, "Roadrunner: Towards automatic data extraction from large web sites," in *VLDB*, 2001, pp. 109–118.

[140] O. Etzioni, M. J. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Web-scale information extraction in knowitall: (preliminary results)," in *WWW*, 2004, pp. 100–110.

[141] T. M. Mitchell, W. W. Cohen, E. R. H. Jr., P. P. Talukdar, J. Betteridge, A. Carlson, B. D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. A. Platanios, A. Ritter, M. Samadi, B. Settles, R. C. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling, "Never-ending learning," in *AAAI*, 2015, pp. 2302–2310.

[142] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *AAAI*, 2010.

[143] X. Zhu, "Semi-Supervised Learning Literature Survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep.

[144] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017.

[145] J. Alcal-Fdez, A. Fernndez, J. Luengo, J. Derrac, and S. Garca, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework." *Multiple-Valued Logic and Soft Computing*, no. 2-3, pp. 255–287.

[146] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.

[147] F. Ricci, L. Rokach, and B. Shapira, Eds., *Recommender Systems Handbook*. Springer, 2015.

[148] Y. Gu, Z. Jin, and S. C. Chiu, "Combining active learning and semi-supervised learning using local and global consistency," in *Neural Information Processing*, C. K. Loo, K. S. Yap, K. W. Wong, A. Teoh, and K. Huang, Eds. Springer International Publishing, 2014, pp. 215–222.

[149] V. Crescenzi, P. Merialdo, and D. Qiu, "Crowdsourcing large scale wrapper inference," *Distrib. Parallel Databases*, vol. 33, no. 1, pp. 95–122, Mar. 2015.

[150] M. Schaekermann, J. Goh, K. Larson, and E. Law, "Resolvable vs. irresolvable disagreement: A study on worker deliberation in crowd work," *PACMHCI*, vol. 2, no. CSCW, pp. 154:1–154:19, 2018.

[151] "Weak supervision: The new programming paradigm for machine learning," https://hazyresearch.github.io/snorkel/blog/ws_blog_post.html.

[152] A. J. Ratner, B. Hancock, and C. Ré, "The role of massively multi-task and weak supervision in software 2.0," in *CIDR*, 2019.

[153] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, 08 2017.

[154] H. R. Ehrenberg, J. Shin, A. J. Ratner, J. A. Fries, and C. Ré, "Data programming with ddlite: Putting humans in a different part of the loop," in *HILDA@SIGMOD*, 2016, pp. 13:1–13:6.

[155] A. J. Ratner, S. H. Bach, H. R. Ehrenberg, and C. Ré, "Snorkel: Fast training set generation for information extraction," in *SIGMOD*, 2017, pp. 1683–1686.

[156] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *ACL*, 2009, pp. 1003–1011.

[157] E. Ferrara, P. D. Meo, G. Fiumara, and R. Baumgartner, "Web data extraction, applications and techniques: A survey," *Knowl.-Based Syst.*, vol. 70, pp. 301–323, 2014.

[158] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "Holoclean: Holistic data repairs with probabilistic inference," *PVLDB*, vol. 10, no. 11, pp. 1190–1201, 2017.

[159] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg, "Activeclean: Interactive data cleaning for statistical modeling," *PVLDB*, vol. 9, no. 12, pp. 948–959, 2016.

[160] S. Krishnan, M. J. Franklin, K. Goldberg, and E. Wu, "Boostclean: Automated error detection and repair for machine learning," *CoRR*, vol. abs/1711.01299, 2017.

[161] M. Dolatshah, M. Teoh, J. Wang, and J. Pei, "Cleaning Crowdsourced Labels Using Oracles For Statistical Classification," School of Computer Science, Simon Fraser University, Tech. Rep., 2018.

[162] V. Raman and J. M. Hellerstein, "Potter's wheel: An interactive data cleaning system," in *VLDB*, 2001, pp. 381–390.

[163] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer, "Wrangler: interactive visual specification of data transformation scripts," in *CHI*, 2011, pp. 3363–3372.

[164] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer, "Profiler: integrated statistical analysis and visualization for data quality assessment," in *AVI*, 2012, pp. 547–554.

[165] W. R. Harris and S. Gulwani, "Spreadsheet table transformations from examples," in *PLDI*, 2011, pp. 317–328.

[166] K. H. Tae, Y. Roh, Y. H. Oh, H. Kim, and S. E. Whang, "Data cleaning for accurate, fair, and robust models: A big data - AI integration approach," in *DEEM@SIGMOD*, 2019.

[167] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification." in *CVPR*. IEEE Computer Society, pp. 2691–2699.

[168] X. Chen and A. Gupta, "Webly supervised learning of convolutional networks," in *ICCV*, 2015, pp. 1431–1439.

[169] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE TKDE*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[170] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, Jun. 2002.

[171] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *CoRR*, vol. abs/1412.6572, 2014.

[172] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE TKDE*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

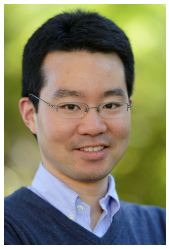
1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2946162, IEEE Transactions on Knowledge and Data Engineering

[173] K. R. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, p. 9, 2016.
[174] "Tensorflow hub," https://www.tensorflow.org/hub/.
[175] F. Li, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, 2006.
[176] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, "Transfer learning in natural language processing," in *NAACL-HLT*, 2019, pp. 15–18.
[177] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NIPS*, 2014, pp. 3320–3328.
[178] O. Day and T. M. Khoshgoftaar, "A survey on heterogeneous transfer learning," *J. Big Data*, vol. 4, p. 29, 2017.
[179] D. Baylor, E. Breck, H. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, C. Y. Koo, L. Lew, C. Mewald, A. N. Modi, N. Polyzotis, S. Ramesh, S. Roy, S. E. Whang, M. Wicke, J. Wilkiewicz, X. Zhang, and M. Zinkevich, "TFX: A tensorflow-based production-scale machine learning platform," in *KDD*, 2017, pp. 1387–1395.
[180] "Tensorflow data validation," https://www.tensorflow.org/tfx/data_validation/.
[181] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 5, pp. 845–869, 2014.
[182] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1106–1114.
[183] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
[184] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *ICANN*, 2018, pp. 270–279.
[185] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *KDD*, 2016, pp. 1135–1144.
[186] ——, "Anchors: High-precision model-agnostic explanations," in *AAAI*, 2018.
[187] E. Krasanakis, E. Spyromitros-Xioufis, S. Papadopoulos, and Y. Kompatsiaris, "Adaptive sensitive reweighting to mitigate bias in fairness-aware classification," in *WWW*, 2018, pp. 853–862.

Yuji Roh is an MS candidate at the School of Electrical Enginnering, Korea Advanced Institute of Science and Technology (KAIST). Her current research interests include Big Data - AI Integration, Big Data Analytics, and Fairness of AI. She received her BS degree in Electrical Engineering from KAIST in 2018.

Geon Heo is an MS candidate in the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST). His current research interests include Big Data AI integration, Big Data Analytics, and Big Data Systems. He received the BS degree from the School of Basic Science, Daegu Gyungbuk Institute of Science and Technology (DGIST), in 2018.



Steven Euijong Whang Steven Euijong Whang is an assistant professor at the School of Electrical Engineering, KAIST. His research interests are Big Data - AI Integration, Big Data Analytics, and Big Data Systems. Previously he was a Research Scientist at Google Research from Dec. 2012 to Jan. 2018 and co-developed the data infrastructure of the TensorFlow Extended (TFX) machine learning platform. Prof. Whang received his Ph.D. in computer science in 2012 from Stanford University and his B.S. in computer science from KAIST in 2003. He is a recipient of the Google AI Focused Research Award in 2018, the first in Asia. He is also an IEEE senior member.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.