Creating Hardware Component Knowledge Bases with Training Data Generation and Multi-task Learning

LUKE HSIAO and SEN WU, Stanford University, USA
NICHOLAS CHIANG, Henry M. Gunn High School, USA
CHRISTOPHER RÉ and PHILIP LEVIS, Stanford University, USA

Hardware component databases are vital resources in designing embedded systems. Since creating these databases requires hundreds of thousands of hours of manual data entry, they are proprietary, limited in the data they provide, and have random data entry errors.

We present a machine learning based approach for creating hardware component databases directly from datasheets. Extracting data directly from datasheets is challenging because: (1) the data is relational in nature and relies on non-local context, (2) the documents are filled with technical jargon, and (3) the datasheets are PDFs, a format that decouples visual locality from locality in the document. Addressing this complexity has traditionally relied on human input, making it costly to scale. Our approach uses a rich data model, weak supervision, data augmentation, and multi-task learning to create these knowledge bases in a matter of days.

We evaluate the approach on datasheets of three types of components and achieve an average quality of 77 F1 points—quality comparable to existing human-curated knowledge bases. We perform application studies that demonstrate the extraction of multiple data modalities including numerical properties and images. We show how different sources of supervision such as heuristics and human labels have distinct advantages that can be utilized together to improve knowledge base quality. Finally, we present a case study to show how this approach changes the way practitioners create hardware component knowledge bases.

CCS Concepts: • **Information systems** → *Data mining*; • **Hardware** → *Electronic design automation*;

Additional Key Words and Phrases: Knowledge base construction, design tools, machine learning

This article is an extended version of Hsiao et al. [16].

We gratefully acknowledge the support of DARPA under Nos. FA86501827865 (SDH) and FA86501827882 (ASED); NIH under No. U54EB020405 (Mobilize); NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); ONR under No. N000141712266 (Unifying Weak Supervision); the Moore Foundation; NXP; Xilinx; LETI-CEA; Intel; IBM; Microsoft; NEC; Toshiba; TSMC; ARM; Hitachi; BASF; Accenture; Ericsson; Qualcomm; Analog Devices; the Okawa Foundation; American Family Insurance; Google Cloud; Swiss Re; the HAI-AWS Cloud Credits for Research program, and members of the Stanford DAWN project: Teradata, Facebook, Google, Ant Financial, NEC, VMWare, and Infosys. We also acknowledge the support of the Intel/NSF CPS Security grant No. 1505728, the Stanford Secure Internet of Things Project, and the Stanford System X Alliance.

Authors' addresses: L. Hsiao, Stanford University, 353 Jane Stanford Way, Gates Bldg. 284, Stanford, CA 94305; email: lwhsiao@stanford.edu; S. Wu, Stanford University, 353 Jane Stanford Way, Gates Bldg. 411, Stanford, CA 94305; email: senwu@cs.stanford.edu; N. Chiang, 353 Jane Stanford Way, Gates Bldg 284, Stanford, CA 94305; C. Ré, Stanford University, 353 Jane Stanford Way, Gates Bldg 408, Stanford, CA 94305; email: chrismre@cs.stanford.edu; P. Levis, Stanford University, 353 Jane Stanford Way, Gates Bldg 409, Stanford, CA 94305; email: pal@cs.stanford.edu.

1539-9087/2020/09-ART42 $15.00
https://doi.org/10.1145/3391906

ACM Reference format:
Luke Hsiao, Sen Wu, Nicholas Chiang, Christopher Ré, and Philip Levis. 2020. Creating Hardware Component Knowledge Bases with Training Data Generation and Multi-task Learning. *ACM Trans. Embed. Comput. Syst.* 19, 6, Article 42 (September 2020), 26 pages.
https://doi.org/10.1145/3391906

1 INTRODUCTION

Creating embedded systems often requires developing new hardware. Searching for components that best meet system requirements constitutes a significant portion of design time. Downloading a datasheet is easy, but figuring out *which* datasheet to download is hard. Typically, the needed information is hidden in the datasheet itself, a complex document that is impenetrable to standard search engines. Requirements are typically multi-dimensional and quantitative, so selecting the right component involves ranges across multiple properties, such as voltage gain and non-textual information like packaging response graphs. Usually there are many (e.g., thousands) of different versions of a component with equivalent functionality but differences in cost, energy, or size. Hardware engineers today conduct component searches by visiting many different web search engines, delicately tuning parameters on each one to get a handful (not zero, not hundreds) of results, manually aggregating the results, then inspecting individual datasheets for information not accessible in these search engines.

This laborious process means that designing hardware requires a library of components in one's head, gained through deep experience. Without this experience, hardware design remains a formidable challenge: Maker forums have detailed discussions on picking the right transistor [36], and entire research papers hinge on careful component selection [17].

The challenge of hardware component selection stands in stark contrast to the ease of selecting good software libraries. Software library information is easily accessible and searchable: Searches yield easy-to-use libraries such as web servers, graphics, or data analysis. Since searches are textual, they can be easily answered by crawling documentation, package descriptions, or community boards such as Stack Overflow. Any given search typically yields only a small number of well-maintained libraries for a given purpose; there are not hundreds of graphing packages comparable to `matplotlib`[1] or hundreds of secure socket libraries comparable to `libssl`.[2]

Hardware component databases are valuable tools for hardware developers. As shown in Figure 1, applications and tools can use these knowledge bases to cross-validate existing databases to answer questions such as "which operational amplifiers should I use to build this gain circuit," or even to query non-textual data like product thumbnails.

Services such as Digi-Key,[3] Mouser,[4] and Parts.io[5] help hardware developers by building proprietary databases; they offer component search pages that drive billions of dollars in sales [11]. However, these databases are often created manually. People with sufficient technical expertise to understand a datasheet (e.g., whether V_{CC} and V_{DD} are interchangeable in a given setting) enter data by hand. Human data entry, however, is prone to both random errors and errors based on individual biases [14]. Further, these databases are incomplete. The cost of data entry results in databases that contain a limited subset of the available information (e.g., a few characteristics out of dozens within a single datasheet).

[1] https://matplotlib.org/.
[2] https://www.openssl.org/.
[3] https://www.digikey.com/.
[4] https://www.mouser.com/.
[5] https://parts.io/.

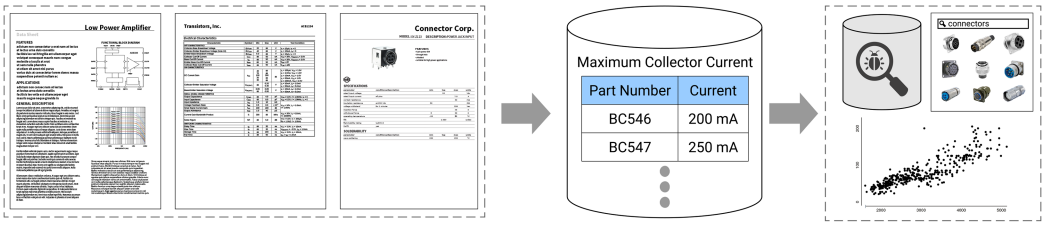
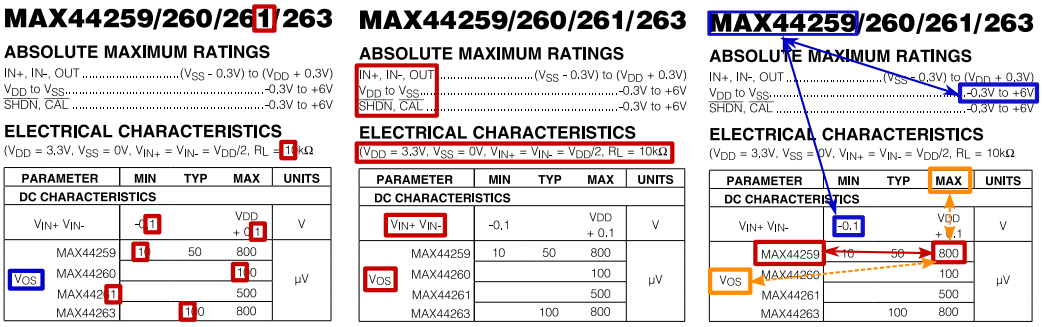


Fig. 1. Hardware component knowledge bases are populated from datasheets and serve valuable applications such as cross-validation, selecting components based on optimal electrical characteristics, or building rich search interfaces.



(a) **Relational data:** a keyword search for "V_{OS}" and "1" may match 1000s of documents as both terms are commonly used. Instead, engineers want to query relational data, e.g., whether a specific part has a minimum "V_{OS}" value of "1 µV".

(b) **Jargon:** datasheets use extensive technical jargon such as the symbols boxed in red. Understanding a datasheet requires both technical expertise and deep experience.

(c) **Input format:** PDF documents lack structural information (e.g., explicit tables), so relationships must be inferred from the rendering of the text, vectors, and images in a document using cues like alignments, proximity, and sizes.

Fig. 2. An example document highlighting the challenges of extracting information from PDF datasheets.

1.1 Learning to Construct Component Databases

This article proposes making hardware component information both accessible and affordable by creating databases from datasheets using state-of-the-art machine learning. This problem requires machine learning, because datasheets are complex, richly formatted documents that rely on many implicit signals and structures to communicate information. Addressing datasheet complexity has traditionally required manual human intervention. Information extraction from datasheets is complicated by three key challenges: relational data, jargon, and input format. Figure 2 shows examples of these challenges drawn from a sample datasheet.

First, hardware component information is relational in nature. Take, for example, the case where a user wants to search for quantitative values of a variety of electrical characteristics (Figure 2(a)). This type of query renders traditional search tools ineffectual, because text-based search alone cannot adequately express these complex relationships. Further, keyword searches commonly match thousands of documents.

Second, datasheets describe components using technical detail and jargon in a wide variety of ways (see Figure 2(b)). Extracting their data requires capturing this domain knowledge in a learning system and precludes relying on untrained crowdsourcing services such as Amazon Mechanical Turk.

Third, datasheets are distributed in Portable Document Format (PDF), and vendors vary significantly in how they present data using textual, structural, tabular, and visual cues (Figure 2(c)). These cues are understandable to humans but are challenging for machines to interpret. Further, the wide variety and non-uniformity of these cues make them impossible to address accurately by simply applying heuristics.

1.2 Proposed Approach

We propose a methodology for creating hardware component knowledge bases. Our methodology builds hardware component knowledge bases by reading thousands of PDF datasheets of multiple component types as input and populates relational databases as output.

We use three machine learning techniques to address the challenges of hardware datasheets. First, rather than modeling input as unstructured text, we use a rich data model that captures the multiple modalities of information provided in a PDF document. This allows us to encode features based on textual, structural, and visual information. Second, we use systematic training data generation, in the form of weak supervision and data augmentation, to efficiently translate domain knowledge into the large amount of data that is required to train a machine learning model on this task. Weak supervision and data augmentation provide multiple ways to combine and benefit from a wide variety of signals such as heuristics and expert human annotations. Third, we train a multi-task learning model that is robust to the data variety in hardware datasheets. This shifts database errors away from random, human errors toward more systematic errors that a machine learning approach can iteratively address and reduce. In addition, using a multi-task learning approach improves efficiency when extracting many characteristics from a datasheet by allowing these extraction tasks to be trained together using a shared feature space (Section 4.2.3).

Other domains have adopted automated methods for creating knowledge bases as a solution to making information accessible [28, 44]. These domains, however, select automated methods focused on unstructured text alone. In contrast, hardware datasheets are compiled for technical readers and include immense data variety typically rendered in dense numerical, graphical, and pictorial formats. The Fonduer framework [41], which provides a general data model for richly formatted documents, treats different types of documents indifferently, seeking to force them into a single framework. In this work, we build on Fonduer but take an opposite approach. Instead of seeking to force documents into a unified framework, we intimately study at onset the characteristics of the hardware datasheets, then carefully customize our methodology to respond to our findings. For example, within a datasheet, different electrical characteristics are expressed in highly similar ways (e.g., in tables with similar headers and structure); this fundamentally aligns with a multi-task learning approach [42]. Consequently, we utilize Fonduer as a tool to capture multimodal information, but we modify it to address the unique challenges of hardware datasheets to extract both textual and non-textual information. Further, we extend it to exploit the fundamental characteristics of the data with Emmental [40], a multi-task learning package. A complete description of the contributions beyond this prior work is in Section 2.

1.3 Contributions

This article makes these primary contributions:

(1) A methodology for creating hardware component knowledge bases using a rich data model, weak supervision, data augmentation, and multi-task learning (Section 3).
(2) The evaluation of this methodology on multiple hardware components, extracting both textual and non-textual information with an average quality of 77 F1 points. We improve on existing human-curated knowledge bases by 12 F1 points on average (Section 4).

(3) Application studies that highlight how these databases make hardware component selection easier (Section 4.3).
(4) A case study illustrating how our approach changes how hardware component knowledge bases are constructed (Section 5).

2 BACKGROUND AND RELATED WORK

Component databases are a key resource in embedded hardware development. Creating these databases is laborious and error-prone, often needing experts with technical knowledge to read datasheets and enter data. As a result, these databases are small unless they are proprietary databases owned by large component search companies. Their small size limits the practical utility of many tools [2, 19, 29]. For example, Drew et al. presented a tool for automatically checking breadboarded circuits, but its underlying knowledge base only supports six types of components [12]. Similarly, Ramesh et al. demonstrated that with a database of components, one can automatically produce an embedded device hardware design from software; however, they defer creating a sufficient library to future work [31].

Recent developments in machine learning and knowledge base construction have demonstrated success in automating the creation of queryable knowledge bases in domains such as paleontology, electronics, and genomics [41]. We build on this prior work and extend these techniques into the domain of supporting embedded system development by targeting hardware component knowledge bases, which have great value but are error-prone and laborious to produce.

2.1 Knowledge Base Construction

Knowledge base construction takes documents as input and outputs a database with a user-defined schema that is populated using information extracted from the input documents. We describe this process as follows:

A **mention**, m, represents a noun, i.e., a real-world person, place, or thing, which can be grouped and identified by its **mention type**, T. For example, "part number" is a mention type, while "BC546" is a corresponding mention. A relationship of n mentions is an n-ary **relation**, $R(m_1, m_2, \ldots, m_n)$, which corresponds to a **schema**, $S_R(T_1, T_2, \ldots, T_n)$. A **candidate** is an n-ary tuple, $c = (m_1, m_2, \ldots, m_n)$, which represents a potentially correct instance of a relation R. For instance, a "part number" and a "price" represent a relation with a schema, $S_R(T_1, T_2)$, where "BC546" and "$1.00" represent a candidate, $c = (m_1, m_2)$, of a 2-ary relation, $R(m_1, m_2)$.

To automate knowledge base construction, machine learning–based systems model this process as a classification task. Candidates are extracted from the input documents and assigned a Boolean random variable where a true value signals that the candidate is a valid instance of a relation. To make that determination, each candidate is assigned a set of **features** as signals for which Boolean value a classifier should assign. Then, these systems maximize the probability of correctly classifying each candidate based on its features and a set of examples, called **training data**.

Ultimately, a supervised machine learning algorithm requires three inputs: (1) candidates, (2) their features, and (3) training data. It then outputs a marginal probability for each of the input candidates. Finally, we **threshold** the output probabilities such that candidates whose probability exceeds the threshold are classified as true, and vice versa.

2.2 Training Data Generation

Training data is a vital input for knowledge base construction systems powered by machine learning. However, it is typically costly to obtain, because it requires domain experts to tediously

label data. *Weak supervision* and *data augmentation* have recently emerged as popular techniques for generating training data. Weak supervision generates training data from *unlabeled* inputs using multiple sources of potentially lower-quality labels, such as crowdsourcing [20, 45], existing knowledge bases [25], or heuristics [32]. Data augmentation generates training data from *labeled* inputs by applying transformations to existing, labeled seed data. These two techniques can be used alone or together to generate training data from diverse inputs, labeled or unlabeled, textual or non-textual.

For weak supervision, users encode domain expertise in the form of **labeling functions**. A labeling function receives each candidate as input, labeling it as true, false, or abstains from voting. Labeling functions can use arbitrary heuristics, which allow them to capture a variety of weak supervision approaches. Because each labeling function can abstain from voting, labeling functions will potentially cover different subsets of the input data and may conflict with each other due to the varying quality of the weak supervision sources. We follow the data programming paradigm [33] and use a generative probabilistic model to estimate the accuracy of each labeling function. These estimates are applied as weights to the output of each labeling function, resulting in a final probabilistic label for each candidate that serves as training data. Rather than relying solely on manual labels, with this approach, we combine manual labels with programmatic heuristics and iteratively generate large amounts of training data.

Weak supervision alone, however, is insufficient when explicit signals that can be used in labeling functions for unlabeled data are limited. This is the case, for example, in non-textual data like images. Using data augmentation to address these challenging datasets is a useful way to generate a large set of training data from a limited quantity of labeled examples.

Data augmentation allows users to generate new training data samples by encoding domain knowledge through **transformation functions**. These functions map a labeled data sample to transformed labeled samples by applying a label-variant or -invariant transformation. For example, a transformation function might rotate a given labeled image several ways (a label-invariant transformation) producing many labeled samples from a single labeled sample. There are a vast number of possible transformations that can be applied to any given data element. However, while some transformations improve quality, others are detrimental. Thus, data augmentation strategies impact scalability and quality [8, 9]. In this article, we use the approach proposed by Reference [43] to efficiently search over the space of transformations.

2.3 Multi-task Learning

Each learning task, such as extracting a single electrical characteristic, requires candidates, features, and training data to train a classifier. A traditional single-task learning approach treats each task independently such that each task has its own feature space and representation. In contrast, *multi-task learning* is a recent machine learning paradigm that leverages supervised data from related tasks simultaneously to create a single shared representation for multiple tasks. Learning multiple related tasks simultaneously often improves performance compared to handling each task independently [4, 10, 22, 24, 39]. A multi-task learning approach provides advantages by allowing data to be "pooled" together across many related tasks and results in a more efficient shared feature space.

Not all tasks benefit from multi-task learning. In some cases, using a multi-task learning approach can reduce performance due to interference between tasks and their data [1, 5]. However, in the domain of hardware component information, the tasks of extracting different electrical characteristics are highly related (e.g., extracting distinct characteristics, but relying on similar patterns in the datasheet).

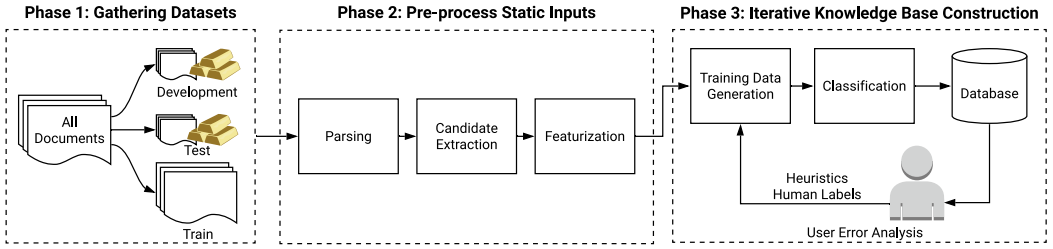


Fig. 3. An overview of our methodology for creating hardware component knowledge bases.

2.4 The Fonduer Framework

We use Fonduer as a tool to capture multimodal information from richly formatted data such as PDF datasheets [41]. In particular, Fonduer provides a rich data model for each document that we utilize for weak supervision in the form of labeling functions, and a feature library that captures signals from multiple modalities of information (e.g., textual, structural, tabular, and visual).

While Reference [41] showed that extracting information from PDF datasheets is possible by extracting a few numerical values from transistor datasheets, this article goes beyond their work in four ways. First, where Reference [41] showed that extracting information from richly formatted data is generally possible, we provide a practical methodology that details how to do so for hardware datasheets; at onset, we study the characteristics of this data and carefully tailor the techniques and implementation in response. Second, we show that our approach is generalizable by extracting hardware component information from three different types of components and by extracting both graphical and textual data, whereas Reference [41] only showed extraction of textual data from a single component. Third, where Reference [41] used single-task learning, we extend their framework to support a multi-task learning approach that better leverages the fundamental similarities between relations in hardware datasheets; this improves both efficiency and scalability as measured by runtime and memory utilization as additional data is extracted. Fourth, we demonstrate applications end-to-end, from dataset creation to application studies that use these knowledge bases, whereas Reference [41] focused on the creation of the knowledge base alone.

3 METHODOLOGY

We divide the process of creating hardware component knowledge bases into three phases: (1) gathering datasets, (2) pre-processing candidates and features as static inputs to a machine learning model, and (3) iterating until we achieve the desired quality (Figure 3). While these phases are broadly applicable to general document interpretation, in this section, we highlight specific challenges and hard-earned best practices that result from extracting information from hardware datasheets. The implementation of each block of the pipeline (e.g., parsing, candidate extraction, featurization) is detailed in Reference [41].

3.1 Phase 1: Gathering Datasets

Creating hardware component knowledge bases begins with a high-quality corpus of documents. Because datasheets are distributed as PDF documents, this phase requires extra preparation to get metadata for each document to arrive at a corpus that allows us to capture non-textual signals like document structure. In addition, because understanding datasheets requires technical expertise,

crowdsourcing ground truth labels from untrained services like Amazon Mechanical Turk results in significantly lower quality knowledge bases.

Acquiring PDF Document Metadata. Manufacturers distribute hardware datasheets as PDF documents that contain tables of relational information. However, unlike HTML or XML documents, which contain structural metadata, PDF documents only contain characters, vectors, and images, along with their rendering coordinates. While datasheets rely heavily on structured tables to present data, the underlying data format contains no explicit metadata about document structure like tables. Consequently, we require supplementary metadata in addition to the raw characters, vectors, and images contained within a document. To satisfy this requirement, we use Adobe Acrobat to acquire metadata by generating an HTML representation for each PDF document.[6] Though the conversion process may introduce noise, the HTML metadata provides valuable information about document structure that complements the visual information in the PDF document that is used in later phases.

Gathering Labels for Evaluation. To evaluate the quality of the final knowledge base, we must have *gold labels*, or ground truth labels, which we can compare against (e.g., by calculating an F1 score). A traditional data extraction pipeline might turn to crowdsourcing to obtain gold labels. However, because of the technical expertise or training required to interpret these datasheets, we find that untrained crowdsourcing is costly and often impractical due to the wide variety of inconsistencies and mistakes that must be corrected [15]. Instead, we recommend involving a domain expert to label data for a small but representative subset of the input corpus. This subset is further divided into a set used for error inspection during development and a set used to assess generalization during final validation. Collaborating with a domain expert also provides benefits for later phases, where insights from the domain expert can directly be leveraged as filters or labeling functions.

3.2 Phase 2: Pre-process Static Inputs

Machine learning algorithms require two static inputs: candidates and their features. The third input, training data, is iteratively generated and refined in Phase 3. To generate candidates and features, we must (1) parse the input corpus into a richly formatted data model, (2) extract candidates, and (3) featurize each of these candidates. In the domain of hardware datasheets, we also must account for implicit information, carefully avoid the combinatorial explosion of candidates, and leverage multimodal features.

Parsing. Manufacturers distribute datasheets as richly formatted PDF documents that convey information through textual, structural, tabular, and visual cues. Therefore, it is vital that we preserve as much of this rich metadata as possible when we parse these documents into a data model. Each subsequent step in the methodology relies on the data model. Implementations where input documents are parsed as unstructured text will lack information such as tabular or visual alignments, which are vital in determining whether a candidate is correct.

Candidate Extraction. Recall from Section 2 that we define candidates as an n-ary tuple of mentions, each of which belong to a particular mention type. To extract candidates, we first define mention types for each of the mentions in the candidate, then we extract the cross product of all mentions of each type to form candidates. Because of this cross product, there can be a combinatorial explosion of candidates, most of which are false. This is particularly prevalent when dealing

[6]Prior work has explored different approaches for extracting subsets of this metadata directly from PDF documents [6, 23, 26], but challenges remain.

with hardware datasheets, where mentions are often simply numerical values in a document. To combat this class imbalance and improve performance, we apply filters at both the mention and candidate levels. For example, if a mention type is a numerical value, we can filter at the mention level by constraining mentions to numerical values within a specific range, but this requires domain expertise to understand the valid range of values. At the candidate level, we can filter based on the candidate as a whole, e.g., discarding candidates in which all of its component mentions are not on the same page of the document. This highlights a fundamental tension between optimizing for system performance and optimizing for end-to-end quality. If we do not filter any candidates, there is an extreme class imbalance towards negative candidates that lowers end-to-end quality. Filtering improves performance by reducing the number of candidates considered and helps reduce the class imbalance. But, after a certain point, additional filtering lowers overall recall and, subsequently, also decreases end-to-end quality.

In addition, hardware datasheets often contain implicit information that should be extracted as a candidate. For example, rather than explicitly listing "BC546, BC547, BC548" as part numbers, a document header may simply contain only "BC546...8." We have extended the Fonduer framework to support implicit candidates so for simple patterns like this, we can expand the text into implicit candidates that are only stored after passing through all filters. However, more complex implicit information remains a challenge (Section 6.2).

Featurization. Next, we featurize each of the extracted candidates using all of the modalities of features provided by Fonduer [41]. Fonduer leverages its data model to compute features that capture signals from multiple modalities of information, such as structural, tabular, and visual features in addition to standard natural-language features such as part-of-speech and named-entity-recognition tags. It then creates a vector for each candidate indicating which of the features each candidate expresses. In simpler domains such as plain text articles, anything beyond textual features may be unnecessary. However, when working with hardware datasheets, we find that final end-to-end quality is best when features from all modalities are present.

3.3 Phase 3: Iterative Knowledge Base Construction

Finally, to generate the data used to train a machine learning classifier, we use two systematic approaches. First, we use labeling functions to unify multiple sources of supervision, such as heuristics and human labels. This enables systematic capture of domain expertise to generate training data from unlabeled sources.[7] This is especially useful when explicit data signals dictate specific label assignments. To mitigate the quality variability of each of these sources, we iteratively refine them, and also our training data, eventually achieving acceptable data quality.

Second, we use transformation functions to generate training data from labeled sources in cases where the data lacks explicit signals that can be used in weak supervision (e.g., for non-textual data like images). Applying data augmentation enables us to expand a small amount of labeled samples into a larger set of training data. We utilize the data augmentation strategy proposed by Dauphin [43].

Using this iteratively generated, larger training dataset, we then train a multi-task discriminative model to create a final knowledge base. With this approach, we have a classic classification problem and can apply logistic regression[8] for text-based relation extraction and a convolutional neural network for image-based relation extraction.

[7]We apply task-specific labeling functions to different tasks separately.
[8]Due to the high sparsity of the features, we use the sparse version of logistic regression to reduce memory usage during training.

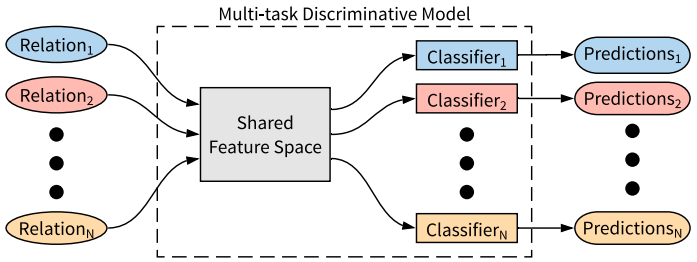


Fig. 4. The multi-task discriminative model architecture, where all tasks share a feature space but use individual classifiers.

To aid in this process, we provide four best practices for developing labeling and transformation functions for hardware datasheets. First, use labeling functions that operate on multiple modalities of information. For example, do not rely on labeling functions that use tabular information alone to determine alignments; use visual alignment as well. Using multiple modalities helps leverage the redundant information in the underlying data, resulting in more robust supervision.

Second, class imbalance (where there are many more negative candidates than positive candidates) is a prevalent challenge. Because of this imbalance, we suggest that labeling functions output true-else-abstain or false-else-abstain, and not output true-else-false or vice versa. Labeling functions that do not abstain label the entire input set, resulting in large numbers of candidates with conflicting labels, which lowers the computed weight for that labeling function. Instead, repurpose accurate labeling functions that label true-else-false as filters during candidate extraction.

Third, when debugging and developing labeling functions, evaluate their effectiveness on the development set, not the test set. Tune and refine the labeling functions by inspecting the true positive, false positive, and false negative candidates. Collaborate with a domain expert to understand how best to capture their expertise. Labeling functions that help reduce class imbalance are preferable; we recommend including only labeling functions with an accuracy of greater than 50%. Typically, high quality is achievable with fewer than 20 accurate labeling functions.

Fourth, use transformation functions that capture domain knowledge from a variety of different perspectives. The goal of these transformation functions is to generate training data that is as representative of your dataset as possible. For example, a user might capture domain knowledge about images in transformation functions that apply rotations, mirroring, rescaling, and blurring to get a better representation of how an image might appear. Because transformation functions are often composed together, a wider variety of transformation functions allows generation of training data that more completely represents the dataset.

We also observe that many relations in hardware component datasets are expressed in a similar way such that their indicating signals may benefit from other similar relations. For example, maximum and minimum storage temperatures are usually expressed in tables using similar formats. Consequently, we replace Fonduer's single-task discriminative model with a multi-task model that learns from multiple relations simultaneously due to the unique characteristics of the hardware domain. For our multi-task model, we implement a shared feature space among multiple relations by using a model with hard parameter sharing [35], i.e., where each task uses an individual classifier, but all tasks share the same feature space and weights. This architecture is shown in Figure 4, where each relation's training data is input to the multi-task discriminative model. Within the model, all relations use a shared feature space and are then classified individually (in our case, using logistic regression) to produce probabilistic predictions for each relation. We implement multi-task learning by incorporating Emmental [40], a multi-task learning package.

Table 1. Summary of the Datasets Used in Our Evaluation Based on Their Size on Disk, Number of Documents, Average Number of Pages per Document, and the Number of Relations Extracted

Dataset	Size	#Docs	#Pgs/Doc	#Rels
Bipolar Junction Transistors	3 GB	6.9 K	5.5	4
Circular Connectors	3 GB	5.1 K	3.2	1
Operational Amplifiers	5 GB	3.3 K	23.3	2

4 EVALUATION

In this section, we evaluate our methodology and examine end-to-end quality and scalability. We perform application studies that illustrate how these datasets can be used to make hardware component selection easier.

4.1 Evaluation Setup

We evaluate our methodology using three distinct hardware component datasets: bipolar junction transistors, operational amplifiers, and circular connectors. We extract relations of electrical characteristics from each dataset.

4.1.1 Datasets. Table 1 shows a summary of our three datasets, primarily sourced from Digi-Key. All of the documents in each dataset are processed to evaluate end-to-end quality (Section 4.2.1). Datasheets were selected by downloading all of the PDF datasheets available on Digi-Key in the respective product category. In addition, the operational amplifier and transistor datasets were augmented with a small number of documents from Octopart[9] and Parts.io. Datasheets that were duplicates, corrupted (i.e., could not be processed by Adobe Acrobat), encrypted,[10] or required optical character recognition (OCR) were filtered out.

These datasets represent immense data variety in terms of both format and style from many manufacturers, who used over 285 unique versions of software tools to author these datasheets, ranging from general purpose tools such as Microsoft Word and OpenOffice, to more specialized tools such as TopLeaf, QuarkXPress, and AutoCAD. These tools also suggest that manufacturers authored these documents in a wide variety of ways, from manual editing in a word processor to automatically generating documentation using custom tools built on PDF libraries such as iText and FPDF. Table 2 shows the tools that were used to author the most documents in our datasets.

Transistors. Transistors are one of the most commonly used and fundamental electrical components. Posts on selecting the correct transistor frequently appear on maker forums [36]. We select transistor datasheets from over 20 unique manufacturers and extract four binary relations primarily contained within tables: minimum and maximum storage temperatures, polarity, and maximum collector-emitter voltages, along with their associated part numbers. Our output is four database tables with the schema (document, part number, attribute value, probability). *We use this dataset to evaluate how our methodology performs using heuristics as a weak supervision source.*

Operational Amplifiers. Huang et al. required an operational amplifier with very specific characteristics [17]. To find potential parts, Huang et al. scraped Digi-Key to explore the trade-off between

[9] https://octopart.com/.

[10] PDFs can be encrypted and password-protected in two ways. User passwords require a password to open a PDF for reading. Master passwords require a password to change permission settings and can be used to restrict printing, editing, and copying content in a PDF. Both PDFs secured with a user password or those that restrict printing and copying content could not be processed by Adobe Acrobat.

Table 2. Top Eight Software Tools Used to Create the 15.3 K Datasheets in Our Total Dataset

Software	# Docs	% Dataset
Apache FOP	1.5 K	9.8
BroadVision QuickSilver	1.4 K	9.2
iText	1.1 K	7.2
Acrobat Distiller	1.0 K	6.5
WinDev	1.0 K	6.5
TurnKey TopLeaf	0.9 K	5.9
Microsoft Word + Adobe PDFMaker	0.9 K	5.9
FPDF	0.4 K	2.6

two electrical characteristics. In contrast, we create that knowledge base using our machine learning approach. Operational amplifiers are more complex and described by datasheets that are 4× longer on average than transistors. We assess datasheets from over 30 unique manufacturers and extract two unary relations, the gain bandwidth product and the quiescent current, to compare our result with that in Reference [17]. Our output is two database tables with the schema (document, attribute value, probability). *We use this dataset to evaluate our methodology when using human labels as a weak supervision source.*

Circular Connectors. Circular connectors, the third largest category of items on Digi-Key with over 490K products from 50 manufacturers, provides a diverse dataset. The sheer number of circular connectors makes it difficult to quickly find the right one, especially since the most important information about circular connectors are not numerical values, but how they look. To that end, we extract a single, non-textual, unary relation—thumbnail images—and output a database table with the schema (document, thumbnail, probability). *We use this dataset to evaluate our methodology when using data augmentation to extract non-textual information.*

4.1.2 Evaluation Metric. We evaluate the end quality of our knowledge bases using *precision*, *recall*, and *F1* score, defined as follows:

$$\texttt{precision} = \frac{tp}{tp + fp} \qquad \texttt{recall} = \frac{tp}{tp + fn} \qquad \texttt{F1} = 2 \times \frac{\texttt{precision} \times \texttt{recall}}{\texttt{precision} + \texttt{recall}},$$

where:

tp = True positives. How many candidates predicted to be positive are true.
fp = False positives. How many candidates predicted to be positive are false.
fn = False negatives. How many candidates predicted to be negative are true.

4.1.3 Implementation Details. We implemented our approach in Python, using Fonduer 0.8.2, Emmental 0.0.6, and PostgreSQL 11.5 for database operations. We used a machine with four physical CPUs (each of which was a 14-core 2.4 GHz Xeon E4–4657L), 1 TB of memory, 2× NVIDIA GeForce TITAN X GPUs, 12× 3 TB disk drives, and Ubuntu 16.04.4 as the operating system. We used Adobe Acrobat Pro to generate an HTML representation for each PDF document to support structural features.

4.2 Evaluation Results

We perform several experiments to evaluate our methodology in terms of end-to-end quality and performance.

Table 3. End-to-end Quality in Terms of Precision, Recall, and F1 Score for Each Dataset

Dataset	Relation	Prec.	Rec.	F1
Trans.	Min. Storage Temp.	0.98	0.58	0.73
	Max. Storage Temp.	0.96	0.61	0.76
	Polarity	0.86	0.92	0.90
	Max. Collector-Emitter Volt.	0.86	0.78	0.83
Op. Amps.	Gain Bandwidth Product	0.74	0.73	0.75
	Quiescent Current	0.72	0.60	0.66
Circ. Conn.	Product Thumbnails	0.68	0.86	0.76

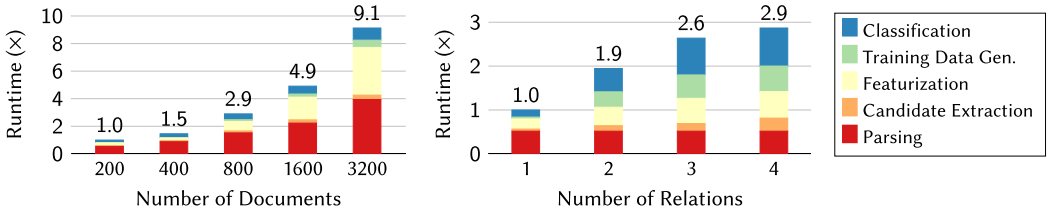
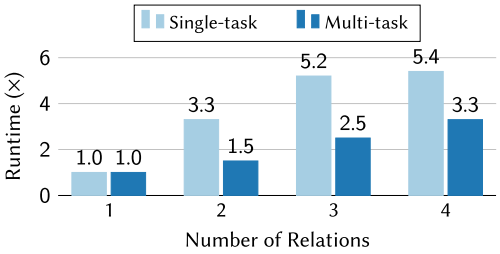


Fig. 5. End-to-end runtime for each computational stage when scaling the number of documents (left) and number of relations from 1 K documents (right).

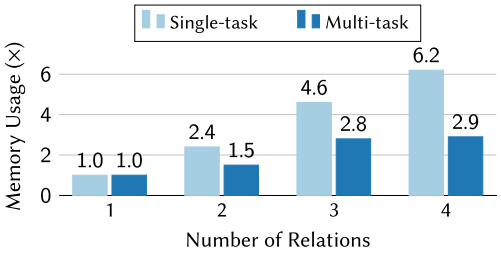
4.2.1 End-to-end Quality. Table 3 shows the precision, recall, and F1 score of each of the relations we extract from each dataset. We achieve, on average, 77 F1 points. Unlike large manually created knowledge bases that have been cultivated and curated for years, our knowledge bases were created in a matter of days. As expected, we achieve lower F1 scores for relations that are more complex. In the transistor dataset, for example, we achieve 90 F1 points for transistor polarities, which take one of two values: "NPN" or "PNP," and usually apply to all parts on a datasheet. However, our score for operational amplifier quiescent currents is only 66 F1 points. This is because quiescent current typically differs for each part listed in a datasheet and is often associated using visual alignments alone, which makes it more sensitive to noise.

In addition, we find that using heuristics as a weak supervision source generally results in higher precision but lower recall (as shown in the transistor dataset) while using human labels as a weak supervision source results in a relatively higher recall-to-precision ratio (as shown in the operational amplifier dataset). The reason for this is twofold: First, providing supervision using heuristics inherently targets specific patterns or features in the data (e.g., that two words are in the same tabular row). Consequently, the machine learning model learns a signal more precisely, since the heuristic is applied systematically across the dataset, but may return lower recall, since other signals are not directly considered. In contrast, providing human annotations for supervision inherently targets specific candidates, not patterns. As a result, human labels will typically cover a more broad set of features resulting in higher recall, yet potentially at the cost of lower precision (see Section 4.4).

4.2.2 End-to-end Scalability and Performance. We perform two experiments to evaluate the scalability and performance of our methodology. In Figure 5, we show examples of the end-to-end runtime that each computational step of our methodology requires. By rerunning training data generation and classification alone, we incrementally refine our generated training data. This



(a) A multi-task approach improves classification runtime compared to single-task as the number of relations increases.



(b) A multi-task approach improves memory usage compared to single-task as the number of relation increases.

Fig. 6. Performance of a multi-task discriminative model when scaling the number of relations.

fine tuning allows amortization of the costs of parsing our corpus, extracting candidates, and featurizing those candidates.

Parsing scales with the number of input documents, while candidate extraction, featurization, training data generation, and classification scale with the number of candidates. Figure 5 (left) shows the relative end-to-end runtime when scaling the number of input documents from the transistor dataset. In Figure 5 (right), we measure the end-to-end runtime of each computational step when increasing the number of relations extracted from 1 K documents of the transistor dataset. Increasing the number of relations or the number of documents parsed are proxies for increasing the number of candidates. From these figures, we see that our methodology scales sublinearly with both documents and relations.

The end-to-end runtime across our datasets was on the order of 10s of hours; this allows us to create hardware component knowledge bases in a matter of days. Our implementation has significant room for optimization to reduce the iteration time required to build these knowledge bases. Optimization efforts could lower system requirements below the thresholds currently needed to process large datasets.

4.2.3 Comparing Multi-task and Single-task Learning. A common use case for building hardware component knowledge bases is to extract many relations from a single type of component datasheet. For example, extracting several electrical characteristics at once for each datasheet in a dataset. In this common case, while each relation represents distinct information, the way that this information is expressed within a datasheet often shares features such as tabular structure, alignments, and keywords.

This observation suggests that extracting hardware component information aligns well with a multi-task learning approach. One of the advantages of a multi-task learning approach is that all tasks are trained and classified using a shared feature representation. Pragmatically, this results in reduced overhead for adding additional tasks, since classification only needs to be performed on a single feature representation, rather than needing to be performed on a unique feature representation for each task.

To evaluate the benefits of a multi-task approach, we compare the runtime, memory usage, and quality of a multi-task approach with those of a single-task approach when scaling the number of relations. In Figure 6(a), we compare the runtime for classification for 1 K documents of the transistor dataset using a single-task and a multi-task learning approach. On average, a multi-task approach reduces the runtime for classification by 2.2 ×. Similarly, we find that a multi-task approach reduces memory usage by 2 × as a result of using a shared feature space for all of the tasks (Figure 6(b)). Finally, in terms of quality, we find that a multi-task approach matches the quality of a single-task approach on these four transistor relations, with an average of 80 F1 points. By

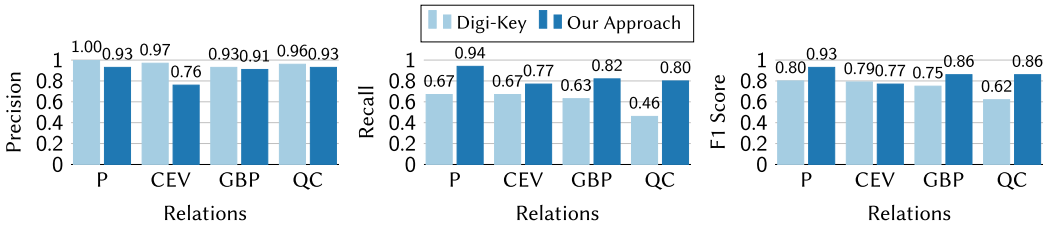


Fig. 7. Quality of our approach vs. Digi-Key for the relations available to compare against ground truth. The polarity, maximum collector-emitter voltage, gain bandwidth product, and quiescent current relations are abbreviated as P, CEV, GBP, and QC, respectively.

Symbol	Rating			Unit
V_{CEO}	Collector-Emitter Voltage	TIP29, TIP30	40	V
		TIP29A, TIP30A	60	
V_{CBO}	Collector-Base Voltage	TIP29B, TIP30B	80	
		TIP29C, TIP30C	100	

(a) Non-textual signals like alignments are vital in associating parts and attribute values. For example, in this example table, horizontal alignment is key for associating specific parts (e.g., TIP29B), with their ratings (e.g., 80).

Parameter	Symbol	Limit	Unit
Collector-base voltage	V_{CBO}	-160	V
Collector-emitter voltage	$V_{(BR)CEX}$	-160	V
Collector-emitter voltage	V_{CEO}	-145	V

(b) Ambiguous datasheets lead to human errors. In this example table, the term "collector-emitter voltage" is used ambiguously between the rows in blue and red. A reader must rely on the different symbols to disambiguate the values.

Fig. 8. Example datasheet tables showcasing challenges faced while extracting parts and attribute values.

leveraging multi-task learning, we improve the performance of our approach without sacrificing quality.

4.2.4 The Benefits of a Machine Learning Approach. The process of manually creating large knowledge bases like Digi-Key is expensive and prone to human error. To better understand the benefits of a more automated approach, we compare our generated knowledge bases with Digi-Key on four relations from a small set of transistor and operational amplifier datasheets using ground truth labeled by domain experts. Of the relations we extract, only these four relations are present in Digi-Key's database. Consequently, only these four relations can be directly compared.

On average, we improve on the quality of Digi-Key's knowledge base by 12 F1 points, primarily by improving recall by 23% while only losing 8% in precision (Figure 7). Digi-Key outperforms our approach in terms of F1 score when extracting maximum collector-emitter voltages. This is primarily due to noise introduced during PDF parsing (see Section 6). For example, in Figure 8(a), some PDF parsers may ignore vertical alignments for the cells boxed in blue and instead collapse all those values into a single sentence per cell. This results in inaccurate structural information, which makes it challenging to correctly associate part numbers with their attribute values.

By inspecting these discrepancies, we find that errors in Digi-Key data for these relations fall into three categories:

(1) Recall: In 66% of these discrepancies, Digi-Key only extracts a subset of the parts or values described in the datasheet. For example, a datasheet may express multiple valid gain values based on how an amplifier is configured, but Digi-Key only extracts one of multiple valid gain values for each amplifier.
(2) Neglecting hierarchy: In 29% of discrepancies, Digi-Key ignores part family information. For example, failing to relate a value to a part family as a whole (e.g., BC546) when all the children of that part family (e.g., BC546A, BC546B, BC546C) share a value.
(3) Inconsistent interpretation: 5% of the discrepancies occur because Digi-Key interprets a value inconsistently. For example, both $V_{(BR)CEX}$ and V_{CEO} can be generally referred to as

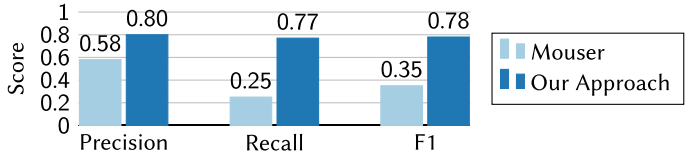


Fig. 9. Quality of our approach vs. Mouser on extracting quiescent current from 100 operational amplifier datasheets.

a "collector-emitter voltage." Ambiguity may cause human annotators to unintentionally extract the wrong value (Figure 8(b)).

Importantly, these error classes are not systematic; they do not follow a regular, consistently applied pattern. Further, these error classes can also vary depending on the individual inputting data.

In contrast, a machine learning approach shifts the class of errors from random to systematic, which can be readily identified and reduced. We inspect the discrepancies and find that our errors can also be classified into three categories:

(1) Heuristic errors: 48% of the discrepancies occur due to weak supervision that systematically fails to properly account for certain key indicators. For example, a subset of datasheets may use a different keyword than the rest to describe a particular electrical characteristic, which we fail to account for in our labeling functions.
(2) Noisy PDF parsing: 39% of the discrepancies occur due to errors introduced when parsing the PDF documents. The resulting noise prevents us from utilizing key document features that would have otherwise enabled us to correctly extract values. For example, failing to maintain the structure of a document's tables introduces errors in tabular relationships within the datasheet.
(3) Stringent filters: 13% of the discrepancies occur due to over-specific filters that fail to capture all values for an attribute. For example, a filter that assumes collector-emitter voltages end in 0 or 5 will filter collector-emitter voltages that end in 2 even if they are valid.

To better understand how our approach compares with other public databases, we also evaluate our trained model on a sample of 100 Mouser operational amplifier datasheets by extracting quiescent currents (Figure 9). We find that, like our comparison with Digi-Key, we significantly improve the recall of the extracted relation—by over 3× in this case. This improvement is caused primarily because Mouser, like Digi-Key, often only extracts a subset of the valid quiescent current values in the datasheet. Unlike our Digi-Key comparison, we improve on precision by 1.4×. We find that Mouser's labels for quiescent current values are often inconsistent. For example, although the majority of the reported values are typical operating values for quiescent current, a large portion of their labels will instead inconsistently report the maximum value. Further, because only a single value is extracted for each datasheet, this inconsistency decreases recall when an incorrect maximum value is labeled and the correct typical value is omitted. These improvements result in an F1 improvement from using our approach of 2.2×. We could not compare our results with services like Parts.io, which do not publish their databases of component information.

Using information posted on websites such as Digi-Key or Mouser is common practice, but this study illuminates how supplier summaries are limited when used as hardware component knowledge bases. Specifically, supplier summaries are, often by design, not exhaustive. Instead, supplier summaries focus on the components stocked and sold by the supplier and may only represent a biased fraction of all available components. While supplier summaries typically maintain very

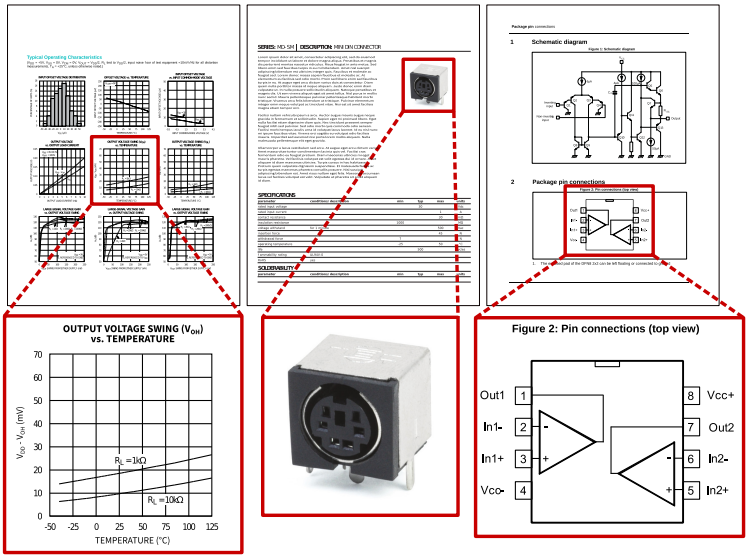


Fig. 10. Examples of non-textual information from real datasheets that applications could utilize. When electrical characteristic summaries are insufficient, designers may want to refer to the detailed characteristic curves (left). When selecting connectors, a preview of a product thumbnail is much faster to check than reading detailed measurements (middle). When prototyping circuits, matching up a component's footprint with its pin diagram is critical (right).

high precision, this selectivity also significantly limits their recall and, consequently, their use as a general hardware component knowledge base.

4.3 Application Studies

In this section, we study two example applications powered by our hardware component knowledge bases and demonstrate how these machine-generated knowledge bases make hardware component selection easier.

4.3.1 Enhancing Catalogs with Non-textual Information. Traditional information extraction systems focus on extracting textual information. However, many component selection and design applications utilize both textual and non-textual information. As shown in Figure 10, datasheets also contain this valuable information, often exclusively, in the form of figures, images, and diagrams.

We extend Fonduer to go beyond traditional text extraction by leveraging the raw pixel-level information contained in hardware datasheets. We extract all images from the datasheets (which are stored in the Fonduer data model) as candidates. This allows us to apply image processing, such as a convolutional neural network, to extract signals from this pixel-level information. There are many classes of applications that may benefit from extraction of this non-textual information, such as automatically generating computer-aided design models from diagrams, gathering schematics and pinouts for embedded design generation [31], or augmenting search engines with product thumbnails.

As an example application, we demonstrate how our approach can be applied to extract non-textual data like images of product thumbnails. With over 495K products listed, circular connectors are the third largest product category on Digi-Key. Unlike transistors or operational amplifiers, one of the most important characteristics of circular connectors is their appearance. Without hardware knowledge bases that contain thumbnails, finding a compatible connector would either

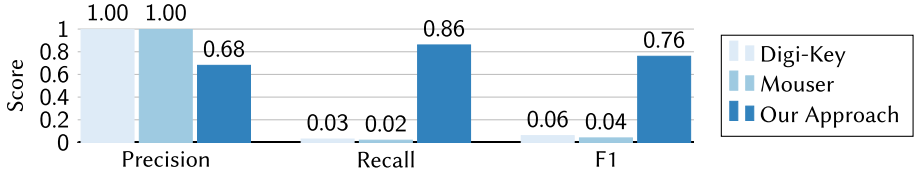


Fig. 11. Even when assuming perfect precision, our approach improves on both Digi-Key and Mouser by 12× in F1 score.

require searching for and tediously inspecting the contents of each datasheet or knowing its exact type beforehand. Major services such as Digi-Key and Mouser try to include thumbnails for all of the components they sell. However, collecting these thumbnails often requires more effort than numerical data entry, resulting in incomplete databases due to the cost. Of the 495K circular connectors in Digi-Key's catalog, less than 3% have thumbnails. Similarly, only 2% of the 823K circular connectors in Mouser's catalog have thumbnails.

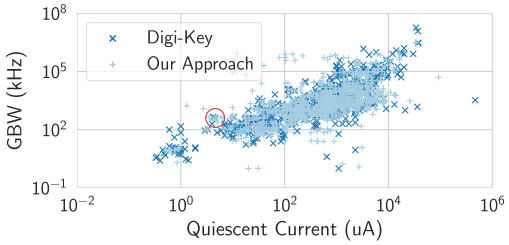
To apply our approach to product thumbnail image extraction, we select a pre-trained ResNet-18 network provided from `torchvision`[11] and fine tune the model's weights based on our dataset. We also augment our training data using techniques described in Reference [43], such as rotations, mirroring, and rescaling. We then use the convolutional neural network to classify which images are product thumbnails for each document. We achieve 86% recall and 68% precision (76 F1 points) on this task to produce a database of product thumbnails that can be displayed with each component. This far simplifies the search for the correct connector. To facilitate a comparison, we estimate an upper bound for the quality of both the Digi-Key and Mouser catalogs of circular connector thumbnails by assuming they have perfect precision (Figure 11) and then compare their metrics against the metrics we calculate on our subset of test documents. Our automated approach improves on the F1 score of both Digi-Key and Mouser by 12× or more.

Our methodology is a new approach to automating the extraction of both textual and non-textual information from hardware datasheets. This example application demonstrates that there are troves of valuable information still locked away in datasheets that can be automatically extracted with our approach in a straightforward way. Although challenges with extraction of non-textual information remain, such as graph interpretation and vector graphics, our approach is extensible and may be augmented to handle these in the future.

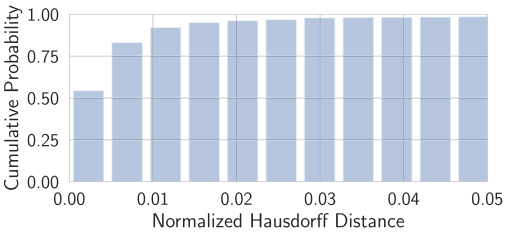
4.3.2 Electrical Characteristic Analysis. Analyzing electrical characteristics is a key part of the process of selecting hardware components. Huang et al. found that the key to their sensor device was the capacity to detect an ultrasonic signal reliably and accurately within a constrained power budget [17]. To accomplish this, they needed a series of operational amplifiers that could provide 1000× gain and were highly motivated to minimize the number of operational amplifiers used in their circuit to minimize noise and physical size.

To aid their search, they performed a survey of operational amplifiers by scraping data from Digi-Key and plotted the gain bandwidth product against the quiescent supply current of each amplifier. Their data is shown in Figure 12(a) in dark blue (×). We extract the same two electrical characteristics from our dataset of operational amplifier datasheets. We filter the gain bandwidth and quiescent current using thresholds of 0.50 and 0.75, respectively. Then, we combine the filtered characteristics based on their reading-order appearance in their datasheets and plot our results in light blue (+). The data from Digi-Key contains 10,877 points, 620 of which are unique. The data from our approach contains 2,280 points, 1,026 of which are unique. Our plot contains fewer points,

[11] https://github.com/pytorch/vision.



(a) We compare our approach with [17]. We find that our data largely overlaps and captures the component selected in [17], circled in red. However, unlike Digi-Key, which has been human curated over many years, our data is extracted directly from PDF datasheets in a matter of days.



(b) Cumulative distribution function of the normalized Hausdorff distance between our approach and Digi-Key. Over 95 % of our data is within 0.018 normalized distance to Digi-Key's data, with a maximum distance of 0.21.

Fig. 12. Comparing the data gathered using our approach against Digi-Key's public data.

because we only extract data from a subset of the total documents on Digi-Key; we exclude those that required OCR, were corrupt, or were encrypted (Section 4.1.1).

To quantify the similarity in the shape of these data points, we leverage the Hausdorff distance, which is a commonly used metric for comparing how similar two point sets are in a metric space [18, 37]. We first normalize the data into the range $[0, 1]$ by taking the logarithm of the data, subtracting by the minimum values of each dimension, and then dividing by the maximum values of each dimension before computing the Hausdorff distance. Then, the normalized Hausdorff distance, D_h, is the maximal Euclidean distance, dist, between any point of one set to the nearest point in another set, normalized by the maximum possible distance, z, where $z = \sqrt{2}$, since the data itself is normalized to the range $[0, 1]$.

$$D_h(A, B) = \frac{1}{z} \max_{a \in A}(\min_{b \in B}(\text{dist}(a, b))) \tag{1}$$

A normalized Hausdorff distance of 0 indicates that every point exactly overlaps between the two point sets, and the distance will approach 1 as the point sets are separated.

We compute the distribution of the distance between each point produced by our approach and the Digi-Key set and plot the cumulative distribution function in Figure 12(b). We find our data largely overlaps with Digi-Key. 95% of our data points are less than 0.018 normalized Hausdorff distance from the Digi-Key set with a maximum distance of 0.21.

Huang et al. ultimately selected a Micrel MIC861/863, which has a gain of 400 kHz and a quiescent current of 4.6 μA. Using our dataset, we are able to identify—with the correct gain bandwidth product and quiescent current values—the same operational amplifiers meeting their design constraints (circled in red in Figure 12(a)). However, rather than using a proprietary knowledge base built over decades using large amounts of manually input data, our approach used a database created from the ground up using training data generation and multi-task learning in a matter of days.

By inspecting a sample of the discrepancies between the two approaches, we find differences that result from both errors in Digi-Key's database (e.g., marking a value as kHz rather than MHz) and errors in our output (e.g., misclassifying an erroneous current value as quiescent current). However, in a machine learning approach, these errors are more systematic in nature and so can be more readily identified and corrected (Section 4.2.4). While both Digi-Key and our database are imperfect, this first-order analysis suggests that our approach can create queryable knowledge bases at scale and with human-like quality, which can serve as a powerful foundation for analysis tools. Our approach can also be applied to new domains, where existing databases may not exist.

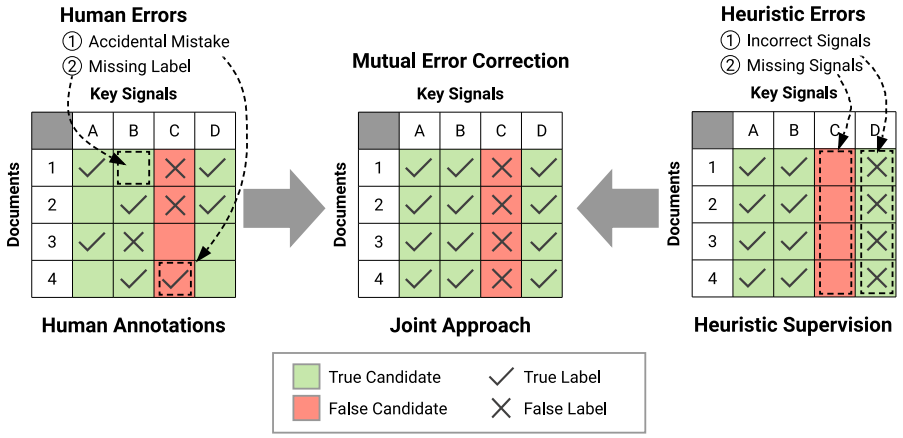


Fig. 13. Our methodology benefits from using both the recall of human annotations and the systematic consistency and precision of a training data generation techniques like weak supervision.

4.4 Discussion

Traditionally, machine learning systems rely on large amounts of manually labeled data. Because human annotations are expensive and time-consuming, this need for human interjection often limits the total training data available. This explains the rise in popularity of techniques like weak supervision, which typically use heuristics to programmatically generate training data.

We observe that heuristic-based supervision and human annotations have distinct characteristics that give rise to unique advantages. Human annotators operate in the context of candidates; they identify specific instances of true or false candidates. This provides sparse but specific information that is agnostic to the underlying features or patterns of the candidates. In contrast, heuristic-based supervision operates in the context of patterns. For example, entire subsets of candidates might be labeled true or false based on a pattern they share. As a result, these heuristics operate precisely based on the underlying features.

The characteristics of these approaches are shown in Figure 13. In this figure, each square represents a candidate. Each candidate in a document has key signals, or features, that correspond to underlying patterns associated with the candidate. Human annotations provide true and false labels for individual candidates and often cover a wide range of signals that a machine learning model can learn from. In contrast, heuristics are applied strictly based on key signals and only label those exact signals with precision. In support of this intuition, we find that the heuristic supervision used in the transistor dataset results in higher precision, while the human annotations used in the operational amplifier dataset results in higher recall (Table 3).

Figure 14 also shows a demonstration of this intuition using the maximum collector-emitter voltage for transistors as an example. Here, we use a small development set of 100 documents that have been manually annotated, and a larger training set of 1 K documents that are unlabeled. We plot the F1 score achieved on a held-out set of test documents when using: (1) human annotations from the small development set to train our classifier (labeled "human"), (2) heuristics on the training and development sets (labeled "heuristics"), and (3) a joint approach of using human annotations for the development set augmented with the heuristics applied to the training set (labeled "joint"). We then sweep over different threshold values where candidates with a probability over the threshold are labeled true and calculate the resulting F1 score. We find that the joint approach

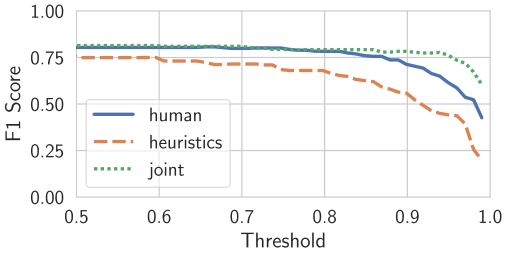


Fig. 14. Utilizing both human annotations and heuristics as weak supervision sources can help improve quality by jointly leveraging the precision of heuristics and recall of human annotations.

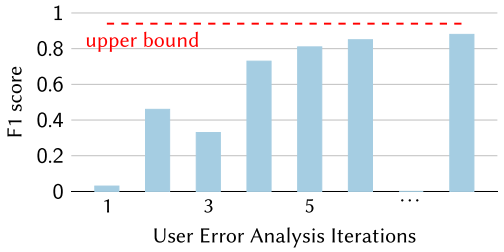


Fig. 15. Developers iteratively refine training data by adding, modifying, or removing labeling functions to increase quality. These iterations can be performed quickly without needing to repeatedly run the full extraction pipeline.

results in the highest F1 scores for larger threshold values, followed by human annotations, and then heuristics.

While these benefits are dependent on the datasets and quality of heuristics and human annotations, this highlights the value of using an approach that can support both sources of supervision. In many cases, a joint approach leveraging the benefits of both supervision from heuristics and supervision from human annotations successfully incorporates the advantages of each individual approach. When certain candidates are difficult to label heuristically, even a small amount of precise human annotations provides sufficient signal to a machine learning model for it to learn systematically.

5 CASE STUDY: A DAY IN THE LIFE OF A DEVELOPER

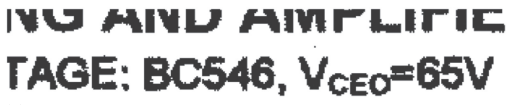
To more clearly illustrate the value of our methodology as applied to the application development process, we describe two common tasks: extracting a new relation and serving queries in production.

5.1 Extracting a New Relation

A key idea of our methodology is changing where developers spend time from manually annotating training data to programmatically creating their training data using weak supervision and data augmentation (Section 3.3). Training data generation and classification are a relatively small portion of end-to-end runtime (Section 4.2.2), allowing developers to quickly and iteratively add and tune their generated training data in the form of labeling and transformation functions to improve the quality of their databases.

Figure 15 shows how F1 score might evolve using weak supervision as an example. A developer incrementally adds, modifies, or deletes labeling functions, including a low-quality labeling function (i.e., one with less than 50% accuracy) in iteration 3, that lowers the F1 score. By analyzing the false positive and negative candidates and evaluating the accuracy of these functions, developers can make incremental adjustments to systematically address their error classes.

As indicated by the dashed line in Figure 15, there is an upper bound on the achievable quality. This limit depends on the quality of the data and the ability to map labeling functions to cues in the underlying data model. For example, PDF parsing errors may introduce significant noise and negatively impact quality (Sections 4.2.4 and 6). While the upper bound cannot be known *a priori*, developers find diminishing returns as they approach this limit and can analyze the errors



(a) Scanned documents are often noisy and can introduce errors after OCR.

(b) Vector-drawn text can be misleading, since vectors are typically used to draw diagrams, but are sometimes used to manually draw text characters.

Parameter	Conditions	AD620A Min	Typ	Max
Common-Mode Rejection				
1 kΩ Source Imbalance	V_{CM} = 0 V to ±10 V			
G = 1		73	90	
OUTPUT				
Output Swing	R_L = 10 kΩ			
	V_S = ±2.3 V to ±5 V	−V_S + 1.1		+V_S − 1.2
DYNAMIC RESPONSE				
Small Signal −3 dB Bandwidth				
G = 1			1000	

(c) Breaking cell boundaries, or where table borders and alignments are loosely followed, often introduce errors during PDF parsing.

Fig. 16. Real-world examples of formatting challenges that the techniques of our methodology do not address.

to triage root causes. In our experience, fewer than 20 iterations are necessary to approach the upper bound.

Takeaway. To extract a new relation, a developer can quickly iterate by running only a small portion of the pipeline (training data generation and classification) to generate large amounts of training data. Utilizing error analysis, a developer can identify and systematically address errors to improve quality.

5.2 Serving Queries with a Trained Model

Subsequent to the development phase are two common serving tasks. First, from the perspective of an *application developer*, the output of the trained model can be used to populate a relational database. This database of hardware component information can then be directly queried and integrated into new applications. Second, from the perspective of an *database developer*, the database can be quickly scaled using additional documents. The developer can use the pipeline and trained model to receive new documents as input and directly execute parsing, candidate extraction, featurization, and classification using the trained model to add more entries into the database. This methodology replaces the traditional approach of using hundreds of hours of manual data entry with a more automated approach of using training data generation to produce a high-quality model that extracts relations at scale.

Takeaway. In production, developers can build new applications using the familiar backend of a relational database. Furthermore, this database is easy to scale; it can automatically be expanded by running additional input documents through an already-trained pipeline.

6 FUTURE WORK

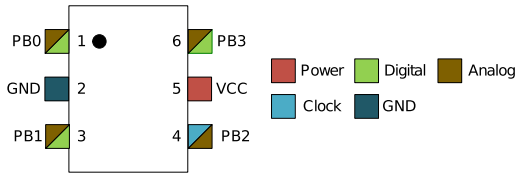
Our methodology is effective for building hardware component knowledge bases of relational information, such as electrical characteristics and their values listed in datasheets. However, below we highlight several significant limitations to guide future work.

6.1 Parsing PDF Documents

PDF documents are the de facto standard for publishing hardware component information. As the primary input, any noise or errors introduced during the PDF parsing process propagate through the rest of the pipeline and negatively affect quality. In the context of datasheets, PDF parsing tools do not fare well in at least three challenging scenarios.

CLASSIFICATION		A	B
h_{FE}	BC856	125 ~ 250	220 ~ 475
	BC857	125 ~ 250	220 ~ 475
	BC858	125 ~ 250	220 ~ 475

(a) Relationships to specific parts are implied by column headers alone. This example table is specifying that BC856A, BC857A, and BC858A have an h_{FE} of 125~250, while those with a B suffix have a value of 220~475.



(b) Relationships may also be specified using color matching, which is intuitive to humans readers, but complex to capture programmatically in a data model.

Fig. 17. Real-world examples of implicit relationships that are not addressed by these techniques.

First, PDF documents consisting of scanned images requiring OCR (e.g., Figure 16(a)) introduce noise that is difficult to eliminate downstream. For example, OCR software may interpret a scanned document containing the text "50 °C" as "50 OC," "500C," or even as "5000," depending on the quality of the original scan and the quality of the OCR software.

Second, because PDF merely specifies characters, vectors, or images and the locations to render them, even native-digital PDF documents that include text as vectors rather than characters can cause OCR issues. For example, some manufacturers publish datasheets where, rather than using text characters, they draw text using vectors, resulting in documents that contain little to no text (Figure 16(b)).

Third, manufacturers author datasheets using a gamut of software tools and design them to be understood visually by human readers. Recall that our methodology leverages structural information such as tabular alignment in a document. To extract this metadata, we pair PDF documents with a more structured format like HTML. When datasheets break the assumptions of common PDF parsing tools, such as allowing content to cross cell borders (Figure 16(c)), parsers introduce additional errors that degrade the resultant data quality.

Researchers continue to propose a variety of techniques designed to address challenging aspects of parsing PDFs, including text extraction [30, 38], table extraction [21, 27], and figure extraction [7, 34]. However, we find that mainstream tools inadequately address the scenarios described above.

6.2 Understanding Implicit Relationships

Also remaining for future work is understanding information that is implicitly expressed in a document. For example, rather than explicitly listing "BC546A, BC546B, BC547A, BC547B" as part numbers, a document header may simply contain only "BC546...547A/B." Here, we must implicitly understand how to expand and associate these part numbers and suffixes. Some datasheets also use these suffixes in isolation to reference a family of part numbers (Figure 17(a)). This challenge is exacerbated when relationships are expressed using color coding or symbolic legends (Figure 17(b)).

6.3 Open Information Extraction

Our methodology extracts precise, pre-defined relations from a corpus of documents. This requires the user to explicitly define relations to extract and create new labeling functions, transformation functions, and gold data for each relation. This process scales linearly with the number of target relations. In response, researchers have proposed techniques in *open information extraction* for extracting large sets of relations without requiring pre-defined specifications [3, 13]. However, these techniques do not yet support richly formatted data such as hardware component datasheets.

7 CONCLUSION

Embedded system design productivity benefits from hardware component information that is both available and accessible. Unfortunately, troves of hardware component information is inaccessibly locked away in datasheets. We present a methodology for creating queryable hardware component knowledge bases directly from their PDF datasheets. We use state-of-the-art machine learning techniques based on weak supervision, data augmentation, and multi-task learning to overcome some of the known challenges of extracting relational information from richly formatted datasheets.

Our approach leverages domain expertise from both heuristics and human labels. Utilizing training data generation, we combine benefits from the sparse but accurate signals of human annotations with the precise and systematic application of heuristics to yield a more robust and effective method of creating knowledge bases.

We evaluate our methodology by applying it to a dataset of over 15K PDF datasheets for transistors, operational amplifiers, and circular connectors. We extract multiple relations and multiple modalities of information from these datasheets such as numerical values from tables, text from paragraph descriptions, and product thumbnail images, achieving an average of 77 F1 points. On average, we improve recall by 23% at a cost of 8% in precision and find that our methodology improves on existing human-curated knowledge bases by 12 F1 points. Finally, we demonstrate examples of real-world applications that meet or exceed the quality of existing applications but can be created in a matter of days.

ACKNOWLEDGMENTS

We thank Mark Horowitz and Björn Hartmann for their feedback on early versions of this work. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, NIH, ONR, or the U.S. Government.

REFERENCES

[1] Héctor Martínez Alonso and Barbara Plank. 2016. When is multitask learning effective? Semantic sequence prediction under varying data conditions. *arXiv preprint arXiv:1612.02251* (2016).

[2] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. Trigger-action-circuits: Leveraging generative design to enable novices to design and build circuitry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 331–342.

[3] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Vol. 1. 344–354.

[4] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2007. Multi-task feature learning. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 41–48.

[5] Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303* (2017).

[6] Hui Chao and Jian Fan. 2004. Layout and content extraction for PDF documents. In *Proceedings of the International Workshop on Document Analysis Systems*. Springer, 213–224.

[7] Christopher Andreas Clark and Santosh Divvala. 2015. Looking beyond text: Extracting figures, tables and captions from computer science papers. In *Proceedings of the Workshops at the 29th Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI'15)*.

[8] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501* (2018).

[9] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. 2019. RandAugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719* (2019).

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[11] Dave Doherty. 2019. About Digikey. Retrieved from https://www.digikey.com/en/resources/about-digikey.
[12] Daniel Drew, Julie L. Newcomb, William McGrath, Filip Maksimovic, David Mellis, and Björn Hartmann. 2016. The toastboard: Ubiquitous instrumentation and automated checking of breadboarded circuits. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 677–686.
[13] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, et al. 2011. Open information extraction: The second generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*.
[14] Benoît Frénay and Michel Verleysen. 2014. Classification in the presence of label noise: A survey. *IEEE Trans. Neur. Netw. Learn. Syst.* 25, 5 (2014), 845–869.
[15] Hector Garcia-Molina, Manas Joglekar, Adam Marcus, Aditya Parameswaran, and Vasilis Verroios. 2016. Challenges in data crowdsourcing. *IEEE Trans. Knowl. Data Eng.* 28, 4 (2016), 901–911.
[16] Luke Hsiao, Sen Wu, Nicholas Chiang, Christopher Ré, and Philip Levis. 2019. Automating the generation of hardware component knowledge bases. In *Proceedings of the 20th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*. ACM, 163–176.
[17] William Huang, Ye-Sheng Kuo, Pat Pannuto, and Prabal Dutta. 2014. Opo: A wearable sensor for capturing high-fidelity face-to-face interactions. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 61–75.
[18] Daniel P. Huttenlocher, Gregory A. Klanderman, and William A. Rucklidge. 1993. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.* 9 (1993), 850–863.
[19] Antonio Iannopollo, Stavros Tripakis, and Alberto Sangiovanni-Vincentelli. 2019. Constrained synthesis from component libraries. *Sci. Comput. Prog.* 171 (2019), 21–41.
[20] Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. 2015. Comprehensive and reliable crowd assessment algorithms. In *Proceedings of the IEEE 31st International Conference on Data Engineering*. IEEE, 195–206.
[21] Ertugrul Kara, Mark Traquair, Burak Kantarci, and Shahzad Khan. 2019. Deep learning for recognizing the anatomy of tables on datasheets. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'19)*. IEEE, 1–6.
[22] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504* (2019).
[23] Ying Liu, Kun Bai, Prasenjit Mitra, and Clyde Lee Giles. 2007. Tableseer: Automatic table metadata extraction and searching in digital libraries. In *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'07)*. ACM, 91–100.
[24] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730* (2018).
[25] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, Vol. 2*. Association for Computational Linguistics, 1003–1011.
[26] Ermelinda Oro and Massimo Ruffolo. 2009. Trex: An approach for recognizing and extracting tables from PDF documents. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR'09)*. IEEE, 906–910.
[27] Martha O. Perez-Arriaga, Trilce Estrada, and Soraya Abad-Mota. 2016. TAO: System for table detection and extraction from PDF documents. In *Proceedings of the 29th International Florida Artificial Intelligence Research Society Conference (FLAIRS'16)*.
[28] Shanan E. Peters, Ce Zhang, Miron Livny, and Christopher Ré. 2014. A machine reading system for assembling synthetic paleontological databases. *PLOS One* 9, 12 (2014).
[29] Raf Ramakers, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2016. Retrofab: A design tool for retrofitting physical interfaces using actuators, sensors, and 3D printing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, 409–419.
[30] Cartic Ramakrishnan, Abhishek Patnia, Eduard Hovy, and Gully A. P. C. Burns. 2012. Layout-aware text extraction from full-text PDF of scientific articles. *Source Code Biol. Med.* 7, 1 (2012), 7.
[31] Rohit Ramesh, Richard Lin, Antonio Iannopollo, Alberto Sangiovanni-Vincentelli, Björn Hartmann, and Prabal Dutta. 2017. Turning coders into makers: The promise of embedded design generation. In *Proceedings of the 1st Annual ACM Symposium on Computational Fabrication*. ACM, 4.
[32] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2020. Snorkel: Rapid training data creation with weak supervision. *The Very Large Data Bases (VLDB) J.* 29, 2 (2019), 709–730. DOI: https://doi.org/10.1007/s00778-019-00552-1

[33] Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 3567–3575.

[34] Sagnik Ray Choudhury, Prasenjit Mitra, and Clyde Lee Giles. 2015. Automatic extraction of figures from scholarly documents. In *Proceedings of the ACM Symposium on Document Engineering*. ACM, 47–50.

[35] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[36] StackExchange. 2015. Choosing the right transistor for a switching circuit. Retrieved from https://electronics.stackexchange.com/questions/29029/choosing-the-right-transistor-for-a-switching-circuit.

[37] Abdel Aziz Taha and Allan Hanbury. 2015. An efficient algorithm for calculating the exact Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.* 37, 11 (2015), 2153–2163.

[38] Jörg Tiedemann. 2014. Improved text extraction from PDF documents for large-scale natural language processing. In *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 102–112.

[39] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* (2018).

[40] Sen Wu. 2019. Emmental: A framework for building multi-modal multi-task learning systems. Retrieved from https://github.com/SenWu/emmental.

[41] Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. 2018. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 1301–1316.

[42] Sen Wu, Hongyang Zhang, and Christopher Ré. 2020. Understanding and improving information transfer in multi-task learning. In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=SylzhkBtDB.

[43] Sen Wu, Hongyang Zhang, Gregory Valiant, and Christopher Ré. 2020. On the generalization effects of linear transformations in data augmentation. In *Proceedings of the International Conference on Machine Learning*.

[44] Ce Zhang, Vidhya Govindaraju, Jackson Borchardt, Tim Foltz, Christopher Ré, and Shanan Peters. 2013. GeoDeepDive: Statistical inference using familiar data-processing languages. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 993–996.

[45] Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I. Jordan. 2014. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 1260–1268.

Received October 2019; revised March 2020; accepted March 2020