A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching

Vamsi Meduri
Arizona State University
vmeduri@asu.edu

Lucian Popa
IBM Research, Almaden
lpopa@us.ibm.com

Prithviraj Sen
IBM Research, Almaden
senp@us.ibm.com

Mohamed Sarwat
Arizona State University
msarwat@asu.edu

ABSTRACT

Entity Matching (EM) is a core data cleaning task, aiming to identify different mentions of the same real-world entity. Active learning is one way to address the challenge of scarce labeled data in practice, by dynamically collecting the necessary examples to be labeled by an Oracle and refining the learned model (classifier) upon them. In this paper, we build a unified active learning benchmark framework for EM that allows users to easily combine different learning algorithms with applicable example selection algorithms. The goal of the framework is to enable concrete guidelines for practitioners as to what active learning combinations will work well for EM. Towards this, we perform comprehensive experiments on publicly available EM datasets from product and publication domains to evaluate active learning methods, using a variety of metrics including EM quality, #labels and example selection latencies. Our most surprising result finds that active learning with fewer labels can learn a classifier of comparable quality as supervised learning. In fact, for several of the datasets, we show that there is an active learning combination that beats the state-of-the-art supervised learning result. Our framework also includes novel optimizations that improve the quality of the learned model by roughly 9% in terms of F1-score and reduce example selection latencies by up to 10× without affecting the quality of the model.

ACM Reference Format:
Vamsi Meduri, Lucian Popa, Prithviraj Sen, and Mohamed Sarwat. 2020. A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3318464.3380597

1 INTRODUCTION

Entity matching (EM) is an important step in data cleaning where the goal is to link different mentions of the same real-world entity. Since many real-world downstream applications can benefit from clean data, improving EM continues to be a topic of fervent research. In particular, a popular approach to EM has been to formulate it as an instance of binary classification: Given relations D_1, D_2 assign one of *match* or *non-match* to each pair of tuples $r \in D_1, s \in D_2$ where r and s represent entity mentions.

Learning a binary classifier usually entails labeled training data upfront (supervised learning), which is a significant investment in terms of human labeling effort. Active learning [27] is a popular alternative that can avoid such prohibitive costs and has a history of application in EM going back almost two decades (early attempts include Sarawagi and Bhamidipaty [26], Tejada et al. [30]). In contrast to supervised learning, active learning employs an *example selector* that chooses the pair of mentions whose labels refine the quality of the classifier learned thus far. By restricting itself to informative pairs of mentions only, active learning hopes to achieve high quality EM while incurring less human labeling effort.

While previous work has evaluated supervised learning with classifiers of different flavors on the EM task (e.g., [21]) and built frameworks such as Magellan [20] that enable supervised learning-based EM workflows, the same cannot be said for active learning. Lacking comprehensive comparative evaluations, it is difficult to say which combinations of classifiers and example selectors work well on the EM task given that several such combinations have been tried in the past. Query-by-committee (QBC) [13, 28] is a specific example selector which has been tried in conjunction with decision trees [30], support vector machines and naive Bayes classifiers [26]. Mozafari et al. [22] propose to implement QBC in a learner-agnostic manner such that the example selector is completely decoupled from the classifier being used. While this makes implementation easier, the question remains whether or not we can gain improved EM quality if the example selector were learner-aware. While QBC has seen sustained use [22, 26, 30], the active learning literature offers other learner-aware example selectors based on margin [31] which has not seen much use in EM. Mozafari et al. is the only previous work we are aware of that compares against margin example selector while Sarawagi and Bhamidipaty mention it but do not evaluate it.

In this paper, we build a comprehensive framework for benchmarking active learning-based EM. Currently, our framework includes representative classifiers of four major types including linear classifiers (e.g., support vector machines), tree-based classifiers (e.g., random forests), non-linear classifiers (e.g., feed-forward neural networks) and rule-based classifiers, and three different types of example selectors including QBC, margin-based and heuristic example selectors (as proposed in Qian et al. [25] to learn rule-based classifiers at scale). While QBC is implemented in a learner-agnostic manner with bootstrap [12, 22] which allows combining QBC with *any* classifier, margin-based and heuristic example selectors are learner-aware. While not all combinations of classifier and example selector make sense (e.g., some of the heuristic example selectors are explicitly designed for rule-based classifiers), our framework allows maximum plug-and-play ability using which we evaluate active learning approaches on several publicly available EM datasets spread across *product* and *publication* domains.

Our main empirical results show that there is little to choose between margin-based example selection and learner-agnostic QBC in terms of quality of EM achieved, but the former usually results in lower example selection latencies. This situation might change however (depending on the dataset), if we learn a learner-aware ensemble of classifiers with margin-based example selection in which case, EM quality in terms of F1-score might outperform QBC. Our best results appear with learner-aware ensembles and QBC (sans the learner-agnostic committee creation). In particular, learner-aware QBC and random forests, which are natural ensembles of decision trees, invariably produce the best quality EM approaching F1-scores of 100%. This highlights the benefits of systematically implementing active learning EM algorithms within the same framework and evaluating them on a level playing field.

Thus, the primary contribution of this work is to build a benchmark framework for active learning-based EM, using which we provide guidelines to practitioners on the combination of learner and example selector that performs the best on various evaluation metrics such as EM quality, latency, #labels and interpretability. Without this framework, we would end up re-implementing the entire, end-to-end active learning pipeline separately for each combination of learner and example selector. Instead, this framework allows for the necessary components to be plugged-in as shown in Fig. 2, thereby requiring minimum or no changes to the remaining components in the pipeline.

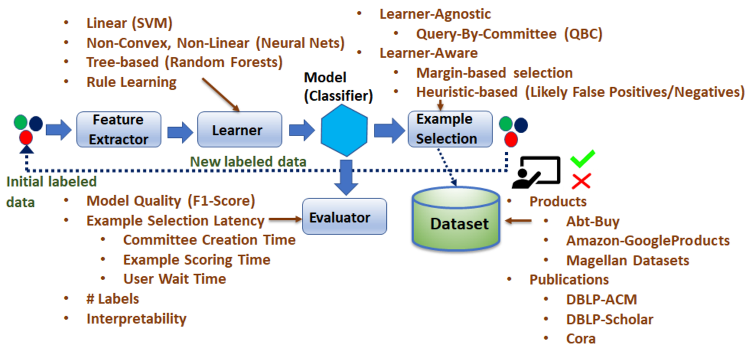
Since both high matching quality and low latency are crucial for active learning scenarios, we propose two general enhancements - learning active ensembles of highly precise classifiers incrementally over several learning iterations for enhanced F1-scores, and blocking to speed up example selection in the context of black-box mathematical models such as Linear SVMs. Given that such enhancements exist to optimize example selection for rule-based classifiers [4, 25], it is essential that these are extended to other learners while evaluating the learners against each other.
Following is a summary of our contributions.

- We develop a unified active learning benchmark framework that allows users to easily combine multiple learning models with several example selectors for EM.
- We conduct an exhaustive experimental study to compare various active learning methods for EM on multiple, publicly available datasets across two domains using our benchmark framework. Our experiments evaluate different approaches on EM quality, example selection latencies and #labels.
- Our framework includes various novel optimizations such as being able to learn ensembles of classifiers with active learning. Other classifier-specific optimizations include the usage of blocking with linear classifiers to reduce example selection latencies .
- We find that random forests with learner-aware QBC can routinely achieve near-perfect EM quality (progressive F1-score close to 100%) on all the datasets we experiment with, while being 10-100x faster w.r.t. example selection latencies than learner-agnostic techniques.

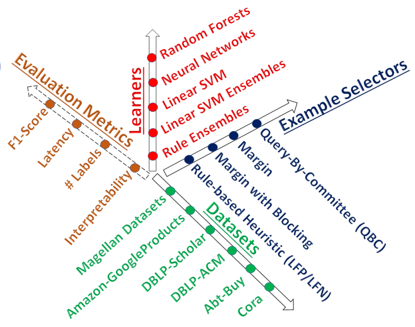
  This significantly improves upon previous EM works, both of the active learning and supervised learning variety, especially on datasets from the product domain.
- While previous work [22] has reported QBC to outperform margin-based example selection, we find that there is little to choose between the two in terms of EM quality, and that the latter can outperform the former if ensembles are learned.
- To estimate the effect of labeling errors in crowd-sourcing situations, we evaluate all our active learning approaches with noisy Oracles.
- We compare rules and tree-based models on an interpretability metric defined by Singh et al. [29]. We infer that although tree-based ensembles perform the best on EM quality, they sacrifice interpretability.

2 RELATED WORK

In this section, we review other example selectors from the active learning literature, how these relate to EM and prior art from areas related to active learning in EM such as crowd-sourcing. Under strong assumptions about data distribution,



(a) Our Unified Active Learning Benchmark Framework



(b) 4D view of Unified Active Learning

Figure 1

the earliest active learning algorithms such as *selective sampling* [8], *query-by-committee* (QBC) [13, 28] and margin-based example selection[15, 31] have been shown to either learn the optimal classifier, or reduce the number of candidate classifiers by a fixed fraction with each labeled example. It is unclear whether such theoretical results hold in practice as QBC and margin-based example selection are reduced to heuristics in this significantly more challenging setting of EM [11]. There exist other active learning algorithms such as IWAL (importance weighted active learning) [6] and ConvexHull [5] which either choose a poor objective of label prediction accuracy (instead of F1-score) for EM which is pervasive of class skew or incur excessive labels in practice.

Several prior works on EM have explored the use of crowd-sourcing however, the focus is usually not on learning an EM model but to reduce the number of labels asked from the crowd [7, 19, 32–37]. Due to the lack of a reusable EM model, one drawback of such approaches is having to incur costs associated with crowd-sourcing labels every time an instance of EM needs to be solved. Our framework is meant for learning a non-trivial EM model with active learning and we emulate crowdsourcing by modeling imperfect Oracles without label correction methods such as majority voting or label inference. Corleone [14] (and its more scalable version Falcon [10]) use random forests due to their interpretable properties to mine the blocking functions automatically, and to perform EM while incurring the least monetary cost for labeling. In our experiments, we too pit random forests against rules to compare them in terms of interpretability. But more importantly, our goal underlying the inclusion of random forests into our framework is to find out how well they can perform EM and and how many labeled examples they incur via active learning.

Currently, our EM framework includes feed-forward neural networks admittedly simpler than recently proposed deep learning architectures that perform EM with representation learning [18, 23]. We evaluate the performance of non-convex non-linear classifiers against other kinds of (shallow) classifiers when learned with active learning. But currently, as we shall see in our experiments, the EM results acquired with complex architectures are clearly behind our best approaches (see Fig. 16 where we compare Mudgal et al. [23] with learner-aware QBC on random forests).

3 BENCHMARK OVERVIEW

Figure 1a presents the system architecture of our unified active learning benchmark framework. In contrast to supervised learning which requires a significant amount of up-front training data, active learning requires a limited amount of initial labeled data (∼ 30 examples in our framework) from which the learner produces an initial model. The example selector chooses ambiguous, unlabeled examples that the model finds hard to predict the label for and queries an *Oracle* (human or ground truth) for those labels. The newly labeled data is added to the cumulative set of training data obtained thus far upon which a refined model is learned. In each active learning iteration, the learned model is evaluated by an evaluator w.r.t. a variety of metrics pertaining to label prediction quality, informative example selection latency, model interpretability and #labels which will be explained in detail. We have four basic components in our framework - *feature extractor, learner, example selector* and *evaluator*. We use the Object-Oriented paradigm of inheritance to model each component as a base class and extend it into a child class to support specialized functionalities.

Feature Extractor: We apply a blocking function as a pre-processing step to eliminate obvious non-matches among the Cartesian product of record pairs created from the tables to be matched. We obtain the feature vectors by applying 21 similarity functions from Java Simmetrics library [1] on all the matching schema attributes across the two tables. If one or both of the pre-aligned attributes of a record pair are null or missing, the similarity evaluates to 0. We use the same set of feature vectors across all the classifiers in the framework barring rule-based models from Qian et al. [25] which only support 3 (equality, Jaro-Winkler and Jaccard) out of the 21 similarity functions. While linear, non-convex non-linear and tree-based classifiers use floating point feature vectors (an example dimension can be JaccardSim(left-table.attr,right-table.attr)), rule-based models evaluate each similarity function on a discrete set of thresholds in (0,1] and create Boolean feature dimensions (e.g., JaccardSim(left-table.attr,right-table.attr)$\geq \tau$ with τ from 0.1 to 1.0).

Learner and Example Selector: As mentioned in section 1, we support a learner from each of the following diverse categories - linear, non-convex non-linear, tree-based and rule-based classifiers. Figure 2 shows how we derive a sub-class for each classifier from the learner base class. Since the base class hosts the common functionalities across all the learners, each sub-class only needs to contain methods specific to a learner. On similar lines, we support a learner-agnostic example selector and two learner-aware selection strategies. While the learner-agnostic selection strategy of query-by-committee (QBC) can be applied to any classifier, random forests inherently learn a committee of trees in a learner-aware manner. Therefore, a relaxed variant of QBC

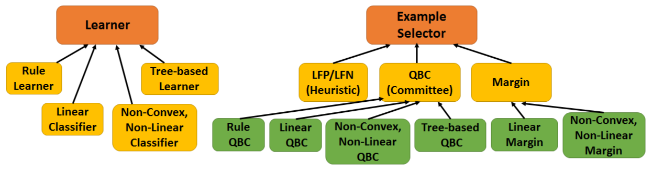


Figure 2: Class Hierarchy of Learners & Selectors

is applied to such tree-based learners. In contrast, learner-aware selection strategies can work only with specific learners. For instance, margin-based selection is compatible with linear and non-convex non-linear classifiers (and is extended accordingly in Figure 2) but not with random forests or rules. Heuristic-based technique of LFP/LFN is devised only for the rule-based classifier in Qian et al. [25] and does not have any child classes. Our framework records the compatibilities between specific example selectors and classifiers through the class hierarchy shown in the figure.

Evaluator: We evaluate the active learning methods on quality, latency, #labels and interpretability.

Quality: The quality of the model is determined by the usefulness of the examples retrieved by the example selector. In each active learning iteration, once we obtain a refined model, we test it on the entire set of data (both labeled and unlabeled pairs obtained post-blocking). Matching pairs get a label of 1 and non-matching pairs are assigned 0 as the label. Precision, recall and F1-score are computed based on the number of matching pairs predicted accurately.

Latency: The time taken by an example selector to retrieve the ambiguous examples in each iteration together with the training time of the model on the cumulative set of labeled examples determine the overall user wait time. The example selection time for QBC can be broken down into committee creation time, which is the time taken to create a committee of classifiers and example scoring time which is the time taken to compute the disagreement (entropy) metric for all the unlabeled examples and pick the most ambiguous ones out of them. For learner-aware approaches such as margin and LFP/LFN the latency only comprises the example scoring time as there is no classifier committee to be created. For tree-based approaches, the committee of random trees is created during the training phase. Hence, the example selection time for random forests is the time required to compute the entropy among the committee of trees. This will be further described in the subsequent sections.

#Labels: This is the minimum number of labeled examples required by each active learning method to learn a model that converges to its best achievable quality. If adding more labels no longer changes the quality of the model learned in terms of its Test F1-scores, the model can be deemed to have reached its convergent state. The lower the #labels, the more effective is the active learning strategy used. If all the unlabeled examples are required to achieve the best possible classifier, it means that the active learning policy used is ineffective and it is better to resort to supervised learning instead, in such scenarios.

Interpretability: This is a metric that determines how readable and interpretable the model is to the end user. Concise rules are preferred over mathematical models by humans especially in scenarios where explainability takes precedence over model quality or effectiveness. Interpretability is defined as being inversely proportional to the number of *atoms* in a rule [29], where an atom is defined as a Boolean predicate that consists of a similarity function applied over an attribute pair accompanied by a threshold. Since random forests are ensembles of decision trees which consist of similar logical predicates, we compare tree-based approaches to the rule-based models [25] w.r.t. interpretability.

4 COMPARED APPROACHES

In this section, we describe the various active learning methods, i.e., example selection policies and how they are applied to each learner we implement in our benchmark. As mentioned in sections 1 and 3, we categorize the example selectors as being learner-agnostic or learner-aware. While query-by-committee (QBC) is a learner-agnostic approach and can be applied to all classifiers, margin and Likely False Positives/Negatives (LFP/LFN) are learner-aware strategies.

4.1 Query-by-committee (QBC)

Mozafari et al. [22] propose query-by-committee (QBC) as a generic strategy that can be applied to any learner. Variants of it have been proposed earlier in Sarawagi and Bhamidipaty [26]. QBC formulates the ambiguous example space based on the disagreement among a committee of classifiers regarding the labels of examples. As illustrated in Figure 3, QBC draws

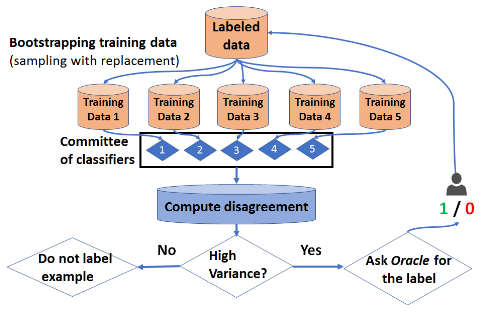


Figure 3: query-by-committee

B (=5 in the figure) bootstrap training samples with replacement out of the aggregate labeled data from which a committee of B classifiers is learned. Each classifier in the committee predicts the labels of all the unlabeled examples and disagreement is computed based on the entropy among the classifiers upon the assigned label to each example. In lieu of entropy, we use variance defined by Mozafari et al. [22] over an an unlabeled example Ex_i as $Variance(Ex_i) = \frac{P_i}{C}(1 - \frac{P_i}{C})$ where C is #classifiers in the committee and P_i is #classifiers which assign a positive class label (matching pair in the context of EM) to Ex_i.

Examples with the highest variance are passed for labeling after which they are included into the aggregate training set. When several examples have the same measure of high disagreement, a random subset of those examples is selected. A way to reduce randomness is to increase the # classifiers in the committee to get fewer examples with the same variance. However there are two hindrances: 1) larger bootstrap committees take longer to train, 2) not every classifier in the committee can be unique as the samples are drawn from the same training set and may contain overlapping examples. In general, larger committees are expected to select more informative examples than smaller committees.

4.1.1 Tree-based Classifiers. As mentioned before, QBC is learner-agnostic and can be applied to all learners such as linear, non-convex non-linear and rule-based classifiers. However, tree-based classifiers such as random forests naturally learn an ensemble of trees in a learner-aware manner during their training phase. Hence, the overhead of creating a committee of classifiers from re-sampled labeled data is unnecessary. We directly use the decision trees in a random forest as the classifier committee to compute the variance on the set of unlabeled examples in each active learning iteration. We use the same settings as the Corleone [14] system to implement the learner for random forests in our benchmark framework. Each random forest contains random decision trees of unlimited depth and uses a random subset of $\log_2(Dim+1)$ features for node splitting from a total of Dim features. Although Corleone uses 10 decision trees per forest, we allow #trees to be parameterized.

4.2 Margin

Margin measures the confidence of a classifier based on how far its predicted labels are from the decision boundary. Although the notion of margin has been originally proposed for linear classifiers, non-convex variants of margin [24] have also been proposed. We apply margin as an active learning strategy to both linear and non-convex non-linear classifiers.

4.2.1 Linear Classifiers. In the ML literature, version space of linear learners can be defined as the candidate set of classifiers that can separate the positive from the negative training examples in the aggregate set of labeled data. Margin-based selection sorts unlabeled examples based on their informativeness and selects those examples whose inclusion into the labeled data leads to a drastic reduction of the version space in each active learning iteration. This results in an earlier convergence to the ideal classifier than committee-based strategies. Margin-based selection for linear classifiers has been theoretically proved to aggressively halve the version space in each active learning iteration in the binary classification scenario [31] thus terming it as an aggressive strategy while naming committee-based techniques like QBC as passive strategies in the ML literature [15].

Margin for a binary linear classifier is defined as the distance of a feature vector to the separating hyperplane and the strategy picks the unlabeled examples which are closest to the hyperplane. Margin can be approximated by the magnitude of the dot product of a feature vector X with the separating hyperplane unit weight vector W added to normalized bias b as $W.X + b$. The sign of the dot product is ignored because ambiguous examples are chosen from both the classes. It is less likely although possible, that two distinct feature vectors fetch the same dot product, thus making margin-based selection more deterministic than QBC.

4.2.2 Non-Convex Non-Linear Classifiers. We use a neural network with a single hidden layer as a non-convex non-linear classifier implemented in our framework. During the forward pass of the training phase, we feed the aggregate labeled data at the input layer of the neural network. Given N labeled record pairs each with a feature vector of Dim

dimensions and a label of 1 to indicate matching or 0 for non-matching, they are passed to a hidden layer which converts each of these *Dim*-dimensional vectors into *h*-dimensional vectors using an affine combination of hidden-weights with the input features followed by a ReLU activation function. *h* is the number of neurons in the hidden layer. The intermediate feature vectors from the hidden layer are normalized using a batch-normalization layer [16] before passing them to the output layer. At the output layer, the intermediate feature vectors are converted from *h* dimensions into a single dimension using an affine layer. The affine output is

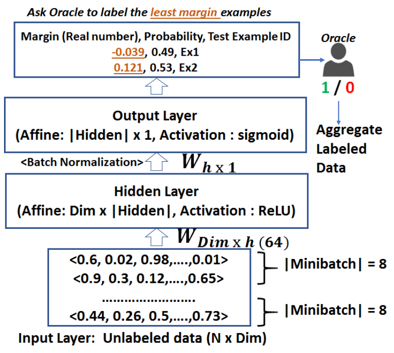


Figure 4: Margin-based selection for Neural Networks

termed as the margin (see margin definition for non-convex classifiers [24]) which is passed to the sigmoid activation function that emits an output probability. If the output probability > 0.5, the record pair is labeled to be matching else, non-matching. We use L2-loss function and Stochastic Gradient Descent (SGD) with momentum as the optimization function to update the weights during the backpropagation phase. We use 50 epochs and a mini-batch size of 8 during training. For SGD, we use a learning rate of 0.001, a decay constant of 0.99 and a momentum of 0.95. We also use drop-out regularization by turning off half of the hidden nodes randomly during training to prevent overfitting. We could see more stability in the neural network predictions because of batch-normalization and drop-out regularization.

Once a trained neural network is obtained in each active learning iteration, we pass the unlabeled examples to the input layer as shown in Figure 4. At the output layer once we obtain the margin and the output probability, we pass the top-K examples with the least margin to the Oracle for labeling and include them in the labeled data. The ambiguity of an example can be inferred directly from the output probability itself. If it is close to 0.5, the classifier is most ambiguous about its label. This intuitive logic can be used to cross-verify the theoretical margin definition from Nguyen and Sanner [24]. Since margin obtained from the affine output layer is fed as an input to the sigmoid function, the lower the margin, the closer to 0.5 its sigmoid evaluation would be.

4.3 Likely False Positives / Negatives (LFP/LFN)

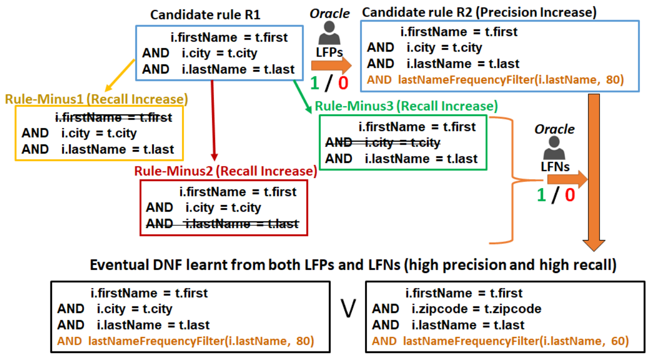


Figure 5: LFP/LFN heuristic for Rule-based Learners

LFP/LFN is an example selection heuristic devised for rule-based learning [25]. Active learning is used to learn entity matching rules expressed as monotone DNF formulas, that is, disjunctions of conjunctive rules constructed from individual atomic predicates. An example of a conjunctive candidate rule matching user profiles across two distinct social media platforms P1 and P2 may be P1.firstName = P2.FName AND P1.lastName = P2.LName AND P1.city = P2.city, based on equality of first and last names and cities. In order to improve precision of the candidate rule, LFP/LFN picks matches predicted by the rule on the unlabeled data that are likely to be non-matches (by using a feature similarity heuristic), and passes these Likely False Positives (LFPs) to the Oracle for labeling.

As a result of such labeling, in the next iteration, the system will learn a higher precision rule. For example, a new, more selective predicate may be added to the earlier conjunctive candidate rule: lastNameFrequencyFilter(P1.lastName,80), filtering out the most frequently occurring last names (e.g., in the top 80 percentile). Similarly, LFP/LFN also identifies pairs of records that are *not* predicted to be matching by an existing rule but are likely to be matches. These are the Likely False Negatives (or LFNs) which are again labeled by the Oracle. The LFNs are obtained by executing relaxed variants of the candidate rule *R* called *Rule-Minus* rules (see Figure 5); by dropping predicates from *R*, these relaxed rules find missed positive examples, and ultimately enhance recall. New conjunctive rules are thus learned from labeled LFPs and LFNs leading to enhanced precision and recall.

5 TIME AND QUALITY ENHANCEMENTS

We propose two enhancements, blocking and active ensembles, to improve the runtime and quality of example selection strategies for active learning. While generally applicable to any underlying active learning model and any example selector, we describe them in the context of margin-based selection for linear classifiers.

5.1 Blocking

Blocking has been used for EM [21] to prune, out of the Cartesian product of all possible pairs of records, those pairs that are unlikely to be matches. In contrast to the work so far, we propose blocking for the specific purpose of discovering ambiguous examples for selection strategies without having to compute the ambiguity metric for the entire space of unlabeled data. Hence, unlabeled examples that are unlikely to be ambiguous will be preemptively ignored using blocking.

While most blocking techniques were devised for rule-based learners, Jain et al. [17] propose two variants of Locality Sensitive Hashing (LSH) to speed up margin-based selection for linear classifiers. Contrary to their approach, the blocking technique we propose forgoes even a full feature vector construction on each unlabeled example and avoids dot product computations aggressively. We apply our learner-aware blocking on top of margin strategy and not for QBC because a majority of the time in QBC is spent in the construction of a classifier committee which is dependent on the already labeled data. So pruning unlabeled data gives meager benefits for QBC.

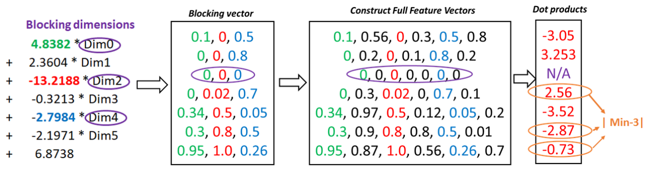


Figure 6: Multiple blocking dimensions for SVMs

Margin-based example selection computes ambiguity based on the distance of each unlabeled example from the separating hyperplane. Our blocking technique skips the margin (dot product) computation for examples whose feature dimensions evaluate to 0 because if all the dimensions of a feature vector X are 0s, $W.X + b = b$ i.e., margin equals bias b whose sign decides the class label of X without ambiguity. However, instead of constructing all the feature dimensions for every unlabeled example, we only evaluate the blocking dimension and check if it is equal to 0. We assume that the blocking dimension has the highest predictive power among all the feature dimensions and if it is 0, then all other feature dimensions evaluate to 0. The weights of all the feature dimensions are readily available in the weight vector W of the linear classifier; a possible blocking dimension is the one with the highest absolute weight.

Since a single blocking dimension may not be predictive enough in determining the values of all remaining feature dimensions, we pick multiple feature dimensions with top-K absolute weights as the blocking dimensions (see Figure 6). As we want to prune away high-confidence examples from both the matching and non-matching classes, the top-K (=3 in the figure) blocking dimensions have the largest magnitude in the weight vector disregarding the sign. As per the figure, all the blocking dimensions evaluate to 0 for the third example. Hence, we skip it and compute full feature vectors and dot products for all other examples and select those with the least absolute dot products (margin) for labeling.

5.2 Active Ensemble of Linear Classifiers

In contrast to learning an ensemble using supervised algorithms, the active ensemble is learned incrementally over several active learning iterations. In the context of entity matching, active learning for an ensemble of several high precision rules, rather than for a single rule, has been shown to significantly enhance recall [4, 25]. Along the same lines, we learn an active ensemble of linear classifiers.

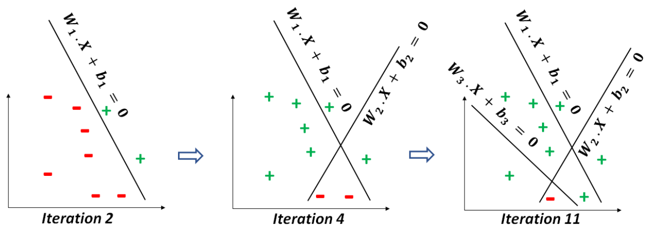


Figure 7: Active ensemble of linear classifiers

Figure 7 illustrates the active ensemble into which three linear classifiers are accepted by the time active learning terminates. We ensure that distinct classifiers are learned into the ensemble by eliminating the positive label predictions (denoted by + in the figure) or pairs which are predicted to be matching by the accepted classifiers in the ensemble from unlabeled and labeled data before attempting to learn a new classifier. The next model is thus learned from the remaining pool of uncovered examples in the subsequent iterations. Eventually, the union of the positive predictions made by all the accepted linear classifiers in the ensemble are labeled as belonging to the positive class. This can lead to high recall at the expense of losing precision if the accepted classifiers do not exceed a preset precision threshold τ. The precision is computed on the selected examples in each active learning iteration whose labels are provided by the Oracle. If the precision computed on the matches predicted by a candidate linear classifier is $\geq \tau$, it is accepted into the ensemble and its covered examples are removed from the labeled and unlabeled example sets. We set τ to 0.85 uniformly on all the EM datasets. Ensemble is a general enhancement and can also be applied to QBC but the prohibitively high committee creation times of QBC confined our implementation of ensemble to margin-based strategies.

Similar time and quality enhancement techniques can also be tried for non-convex non-linear models though we have not explored those in this paper. A possible blocking solution for non-linear classifiers would be to include the largest weights for each exponent - X^n, X^{n-1},..., X^2, X^1 as the blocking dimensions. Blocking during example selection for rule-based or tree-based models is trivial as the blocking predicate (similarity function and threshold evaluation) can be executed on all the unlabeled examples to prune away non-qualifying examples. Active ensemble for neural networks can be applied as discussed in the current section without much of a modification. Learning active ensembles for rules already exists in Qian et al. [25].

6 EXPERIMENTS

We use a cluster with 24 Intel Xeon 2.4GHz CPUs each containing 6 CPU cores and 99 GB main memory, but a limited Java heap space of 4 GB. We use Weka [3] for the implementation of SVM and random forests while we use Apache SystemML [2] for neural networks. As we have mentioned in Section 1, we answer the following questions:

- Among the example selection strategies applicable to each classifier, which is the best performing approach w.r.t. both EM prediction quality and latency?
- Can active learning methods achieve comparable quality metrics as supervised learning? If so which is the best combination of learner and example selector?
- How many labels are required by each active learning method on a dataset to reach a convergent F1-score?
- How does rule-based learning [25] compare to tree-based learners on quality and interpretability?

Our experiments can be classified into two broad categories where we assume the presence of either "perfect" or "imperfect (noisy)" Oracle. We use perfect Oracles without labeling error for our experiments on Abt-Buy, Amazon-Google Products from the Products category and DBLP-ACM, DBLP-Scholar and Cora from the Publication domain. For experiments using noisy Oracles, we choose Abt-Buy, Walmart-Amazon (Products), Amazon-BestBuy (Electronics), BeerAdvocate-RateBeer (Beer) and BuyBuyBaby-BabiesRUs (Baby Products) upon which earlier works like Magellan [20] and DeepMatcher [23] achieve an F1-score of 0.6 - 0.7.

Each dataset contains left and right tables that produce a Cartesian product of record pairs, whose size is denoted by "#Total Pairs" in Table 1. To reduce the size of candidate pairs to be matched, during the feature extraction phase (see Section 3), we prune away the obvious non-matches using Jaccard similarity function with a numerical threshold in an offline blocking step on the tokenized attributes from each pair. We set the threshold to 0.1875 to roughly retain the same number of post-blocking pairs as Mozafari et al. [22], Wang et al. [34] on Abt-Buy, DBLP-ACM and DBLP-Scholar. We use conservative similarity thresholds of 0.12 on Amazon-GoogleProducts and 0.16 on Cora and Walmart-Amazon to avoid pruning too many non-matching pairs. Due to the unavailability of the entire ground truth for the Amazon-BestBuy, Beer and Baby Products datasets, we use Labeled Data L from Das et al. [9] as the set of post-blocking pairs.

Train-Test Splits and Termination Criteria: We start active learning with a seed of 30 labeled examples. In each active learning iteration, we query the Oracle (which happens to be the available ground truth on these datasets) for the labels of a batch of 10 examples chosen from the unlabeled set, upon which the learned model is refined and evaluated on the test set. We use the following settings for train-test splits.

- We evaluate active learning methods on the test set created from all the post-blocking pairs, while progressively querying the Oracle for a sample of them to be added to the training set in each labeling iteration. While an earlier crowd-sourcing work Vesdapunt et al. [33] defines progressive recall, we analogously define *progressive* $F1$ as the test F1-score obtained on post-blocking pairs.
- For active vs. supervised learning experiments, we create the conventional train-test splits (with the same class skew as post-blocking pairs) used in supervised learning scenarios. 80% of the post-blocking pairs form an unlabeled set, out of which examples are selected in each learning iteration, while the remaining 20% form a held-out test set upon which the learned models are evaluated. We use this only for the experiments in Fig. 16 and 17.

The termination criteria differ between perfect and imperfect Oracles. In the case of perfect Oracles, once an active learning method achieves a convergent F1-score, there is little change to it with the addition of more examples. In contrast, in the case of imperfect Oracles, the addition of more examples leads to deteriorating F1-scores because of an added amount of noisy labels. Therefore, we terminate active learning with perfect Oracles in Fig. 8 to 13 as soon as either one of the approaches achieves a near-perfect (close to 1.0) F1-score or all the examples are labeled. In the experiments on noisy Oracles from Fig. 14 to 17, the termination criterion is the exhaustion of all unlabeled examples. Rule-based learners terminate as soon as no likely false positives (LFPs) or likely false negatives (LFNs) are found among the selected examples. This results in no new rules being discovered, that leads to early termination. We present the results on perfect Oracles in Section 6.1 while Section 6.2 contains those on noisy Oracles. Section 6.3 covers interpretability.

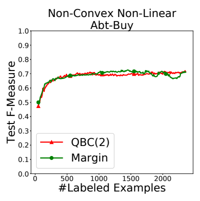
6.1 Comparison of Selectors & Classifiers

In this section, we assume the presence of a perfect Oracle with no labeling error. We first compare various example

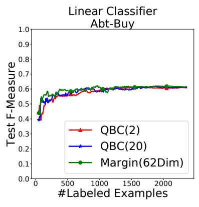
Table 1: Details of the Public EM Datasets.

Dataset	Matched Columns	#Total Pairs	#Post-Blocking Pairs	Class skew
Abt-Buy	{name, description, price}	1.18 M	8682	0.12
Amazon-GoogleProducts	{name, description, manufacturer, price}	4.39 M	14294	0.09
DBLP-ACM	{title, authors, venue, year}	6 M	11194	0.198
DBLP-Scholar	{title, authors, venue, year}	168 M	49042	0.109
Cora	{author, title, venue, address, publisher, editor, date, vol, pgs}	0.97 M	114525	0.124
Walmart-Amazon	{brand, modelno, title, price, dimensions, shipweight, orig_longdescr, shortdescr, longdescr, groupname}	56.37 M	13843	0.083
Amazon-BestBuy	{brand, title, price, features}	21.29 M	395	0.147
BeerAdvocate-RateBeer	{beer_name, brew_factory_name, style, ABV}	13.03 M	450	0.151
BuyBuyBaby-BabiesRUs	{title, price, is_discounted, category, company_struct, company_free, brand, weight, length, width, height, fabrics, colors, materials}	54.5 M	400	0.27

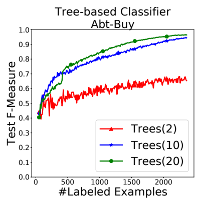
selectors applicable to each classifier. Subsequently we compare the best strategies from each family of classifiers in order to understand the combination of classifier and example selector that works best on a majority of the datasets.



(a) Non-Convex Non-Linear

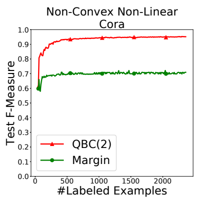


(b) Linear

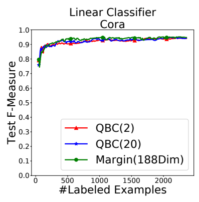


(c) Tree-based

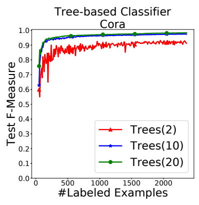
Figure 8: QBC vs. Margin *(Progressive F1, Abt-Buy)*



(a) Non-Convex Non-Linear



(b) Linear



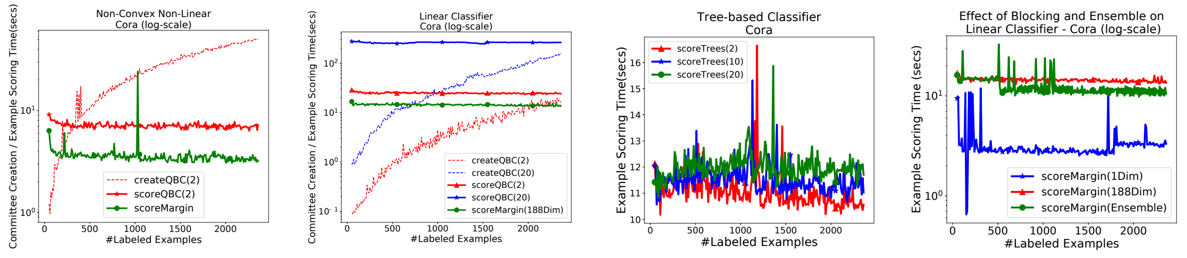
(c) Tree-based

Figure 9: QBC vs. Margin *(Progressive F1, Cora)*

Comparison of Example Selectors: Figs. 8 and 9 show the progressive F1-scores of various example selectors applied to non-convex non-linear (neural networks), linear (SVMs) and tree-based (random forests) classifiers respectively. Upon neural networks and SVMs, we compare the learner-agnostic strategy of QBC against margin as the learner-aware strategy. Since neural networks take a long time to train, larger committees are prohibitively expensive in terms of example selection latency. Therefore, we implemented a smaller committee of size 2 for QBC on neural nets, whereas for SVMs (see Fig. 8b, 9b), we implemented both QBC(2) and QBC(20) with 2 and 20 learners in the committee. In the case of random forests, margin is inapplicable and the trees in the forest are equivalent to the classifier committee in QBC but created in a learner-aware manner. The F1-scores are plotted for 233 active learning iterations until 2360 labeled examples are consumed (including the initial training set of 30 examples) because this is the maximum #labels needed among all the approaches for the best convergent progressive F1 score.

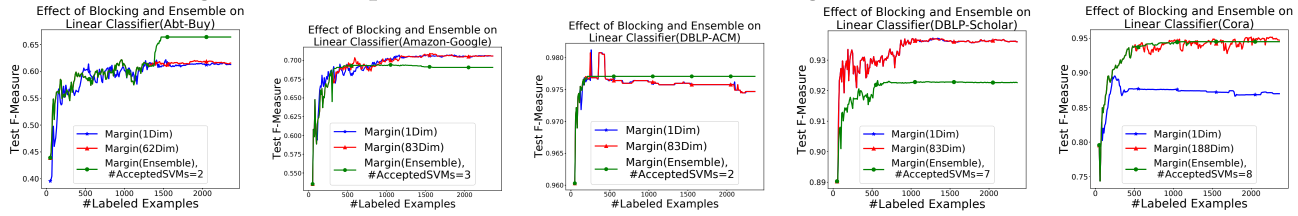
As we can see from Figs. 8 and 9, margin-based selection achieves similar F1-scores as QBC in conjunction with all learners on both Abt-Buy and Cora. The observations are similar on Amazon-GoogleProducts, DBLP-ACM and DBLP-Scholar although we do not plot them for space reasons. The only exception is in the case of neural networks on the Cora dataset (Fig. 9a) where QBC(2) outperforms margin. Likewise, we plot example selection times on Cora because it has the highest number of pairs post-blocking (114K) incurring the longest example selection time among all the datasets. The latency trends we observe in Fig. 10 hold on the remaining datasets as well. Figs. 10a and 10b present the QBC selection time broken down into committee creation (dashed lines in the figures) and example scoring times (solid lines) (see latency metric in Section 3 for definitions).

While the committee creation times increase with more active learning iterations and labeled examples, example scoring times decline with more labels as the unlabeled set shrinks gradually. Thus, margin-based strategy consumes lesser example scoring times than QBC on both neural networks and SVM. Adding the committee creation time to the scoring time of QBC leads to margin outperforming QBC by 10-100x on aggregate selection times. In contrast to neural networks and SVMs for which we explicitly learn the committees from resampled training data, in the case of random forests, the committee is learned during the training phase and example selection only involves scoring the unlabeled examples based on committee labeling variance. Therefore the difference in example selection time (Fig. 10c) among the forests of different #trees (2, 10 and 20) is not too high either. The training times are also not too different because of optimized learner-aware ensemble learning. Ensembles



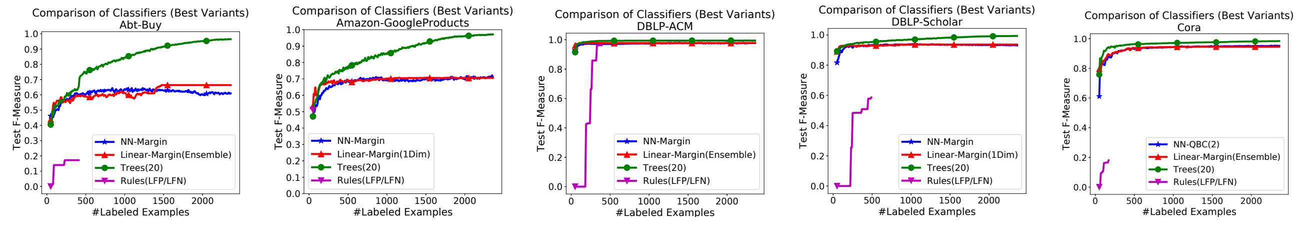
(a) Non-Convex Non-Linear **(b) Linear** **(c) Tree-based** **(d) Linear (Enhancements)**

Figure 10: Example Selection Times of various Strategies on each Classifier *(Cora)*



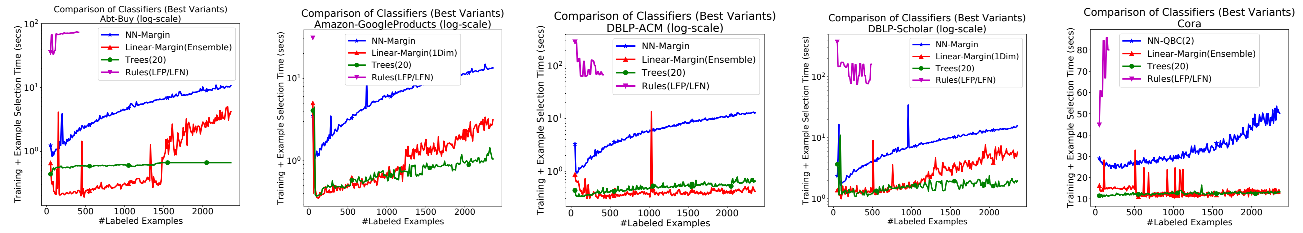
(a) Abt-Buy **(b) Amazon-Google** **(c) DBLP-ACM** **(d) DBLP-Scholar** **(e) Cora**

Figure 11: Effect of Blocking and Active Ensemble on Linear Classifiers *(Progressive F1-Scores, Perfect Oracle)*



(a) Abt-Buy **(b) Amazon-Google** **(c) DBLP-ACM** **(d) DBLP-Scholar** **(e) Cora**

Figure 12: Comparison of Classifiers with Best Selection Strategies *(Progressive F1-Scores, Perfect Oracle)*



(a) Abt-Buy **(b) Amazon-Google** **(c) DBLP-ACM** **(d) DBLP-Scholar** **(e) Cora**

Figure 13: Comparison of Classifiers with Best Selection Strategies *(User Wait Time)*

with 20 trees, Trees(20), achieve a near-perfect progressive F1 on all the datasets as compared to smaller tree ensembles.

Effect of Blocking and Ensembles on SVM: As we have discussed in Section 5.1, blocking dimensions are used to prune the ambiguous example space during margin computation. Thus, if we assume that all the dimensions in the weight vector are blocking dimensions, the margin has to be computed for every example as none of the examples gets pruned away. This turns out to be equivalent to not using blocking in the first place. We compare the margin baseline that uses all dimensions for blocking against using a single blocking dimension. As we can notice from Fig. 10d, using a single blocking dimension denoted by margin(1Dim), brings savings in example selection time without sacrificing quality (see Fig. 11). With the exception of the Cora dataset (Fig. 11e) where margin(1Dim) performs worse than the baseline margin(188 Dim), blocking performs same as vanilla margin flavors - margin(62Dim) from Abt-Buy and margin(83Dim) from Amazon-GoogleProducts, DBLP-ACM and DBLP-Scholar on all the other datasets (Fig. 11a to 11d).

As we have described in Section 5.2, active ensembles learned incrementally over several active learning iterations prune predicted matching pairs from both labeled and unlabeled examples. This results in the example selection time using active ensembles decreasing aggressively in the later active learning iterations as shown in Fig. 10d. While active

Approach	Abt-Buy	Amazon-GoogleProducts	DBLP-ACM	DBLP-Scholar	Cora
Trees(20)	**0.963 (2360 labels)**	**0.971 (2360 labels)**	**0.99 (260 labels)**	**0.99 (1770 labels)**	**0.98 (1700 labels)**
Linear-Margin(Ensemble)	0.663 (1470)	0.69 (330)	0.977 (210)	0.922 (560)	0.945 (1220)
Linear-Margin(Blocking)	0.61 (640)	0.7 (930)	0.975 (170)	0.936 (920)	0.89 (220)
Linear-QBC(2)	0.61 (1420)	0.7 (1550)	0.976 (170)	0.935 (1090)	0.941 (2190)
Linear-QBC(20)	0.61 (1620)	0.7 (1260)	0.976 (180)	0.936 (1600)	0.95 (2130)
Non-Convex Non-Linear-Margin	0.63 (670)	0.72 (2360)	0.978 (1100)	0.938 (970)	0.709 (410)
Non-Convex Non-Linear-QBC(2)	0.63 (970)	0.725 (1350)	0.97 (90)	0.949 (740)	0.95 (1640)
Rules(LFP/LFN)	0.17 (230)	0.51 (50)	0.962 (350)	0.586 (490)	0.18 (170)
Mudgal et al. [23]	0.628	0.693	0.984	**0.947**	N/A
Singh et al. [29]	N/A	**0.694**	N/A	0.9436	0.9718
Kopcke et al. [21]	0.713	0.622	0.976	0.894	N/A
Kasai et al. [18]	N/A	N/A	**0.985**	0.929	0.987
Mozafari et al. [22]	0.56	N/A	N/A	N/A	N/A
Corleone [14]	N/A	N/A	N/A	0.921	N/A
Waldo [32]	**0.8 (2200)**	N/A	N/A	N/A	**1.0 (1600)**
Whang et al. [37]	N/A	N/A	N/A	N/A	0.9 (1000)
Chai et al. [7]	N/A	N/A	0.9 (150)	N/A	0.92 (354)

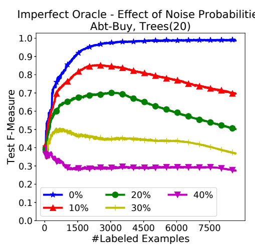
Table 2: Best Progressive F1-Scores from our Benchmark using Perfect Oracles vs. State-of-the-art Approaches

ensembles of SVM achieve slightly higher progressive F1 on Abt-Buy and DBLP-ACM (see Fig. 11), they perform slightly worse than baseline margin on Amazon-GoogleProducts and DBLP-Scholar and show no effect on Cora. This is because of a uniform precision threshold of 0.85 we use across all the datasets. This threshold is conservative enough for Abt-Buy and DBLP-ACM, which can be inferred from two high precision SVMs accepted into the ensemble by the termination of active learning. However, this may not be a suitable threshold for DBLP-Scholar and Amazon-GoogleProducts on which there is no significant boost in the progressive F1-score despite accepting 7 (or 3) SVMs into the ensemble.

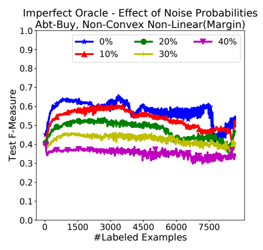
Comparison of Classifiers: The results we have reported so far fix the classifier and vary the example selector. In Figs. 12 and 13, we compare the best performing example selectors from each of the classifiers (margin for neural nets, margin with ensemble or blocking for linear SVMs, learner aware QBC(20) for random forests and LFP/LFN for rule learners) against each other w.r.t. progressive F1-score and user wait time (which is the sum of train time and example selection time, see Section 3 for definition). Having compared these best example selectors from each classifier, we observe from Fig. 12 that random forest with QBC(20) labeled as Trees(20) outperforms all other learners upon progressive F1-scores on all the datasets. We find that rules lead to the largest user wait time, least progressive F1-scores and early termination. However, they perform much better on interpretability and produce an ensemble of concise, high precision DNF rules that can be easily understood and debugged by the end user. We delve into the details of rule-based results in the section on interpretability. Neural networks incur the second largest latency because of the long training they undergo, while random forests require the user to wait for the shortest time despite training an ensemble of 20 trees. This emphasizes the importance of learner-aware training rather than learner-agnostic training used in QBC. SVMs with blocking and ensembles incur the least user wait times in the beginning but with the arrival of more labels, the training time increases thus increasing the wait time between iterations.

#Labels: Table 2 presents the best progressive F1-scores from the active learning approaches implemented in our benchmark (highlighted in green) and the best F1-scores reported by earlier related works (highlighted in red). For our benchmark results, we also present within parentheses, the minimum # labels required by the approaches from a perfect Oracle, to converge to the corresponding F1-scores. For results on noisy Oracles, please refer to Section 6.2. Learner-aware committees (size 20) of tree-based learners achieve the best results close to 1.0 on all the datasets but also consume the largest # labels. However, we can also notice from Fig. 13 that these approaches achieve them with least user wait time. Among the active learning methods for linear classifiers, margin-based optimizations of blocking and active ensemble achieve comparable progressive F1 as QBC while requiring fewer labels and lesser user wait time on almost all the datasets. Although QBC(2) of non-convex non-linear classifiers consumes fewer labels than its margin counterpart on 3 out of 5 datasets and achieves similar F1 scores, training committees of neural nets incurs huge training times. Rule learning using LFP/LFN terminates as soon as no LFPs or LFNs are found on the learned ensemble of high precision rules. This keeps its #labels low and because of the limited number of similarity functions supported by the heuristic, it achieves low progressive F1-scores.

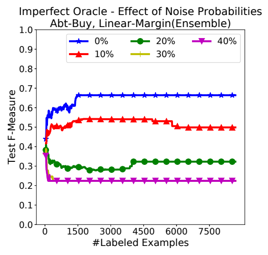
Results from Related Work: Among the related work, we found supervised, transfer and crowd-sourced learning-based approaches which have reported results on our experimental datasets. Note that each of these works use their own set



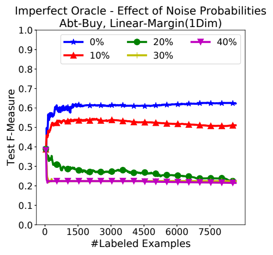
(a) Tree-based



(b) Non-Convex Non-Linear

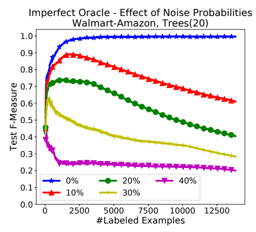


(c) Linear (Ensembles)

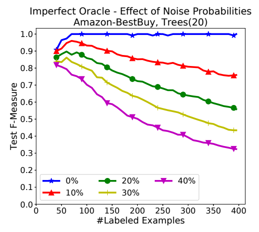


(d) Linear (Blocking)

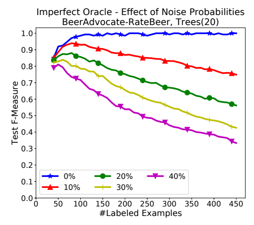
Figure 14: Active Learning using a Probabilistically Noisy Oracle *(Abt-Buy, Progressive F1-Scores)*



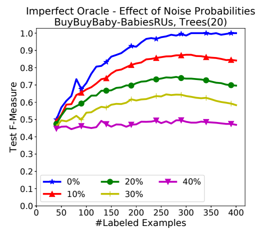
(a) Walmart-Amazon



(b) Amazon-BestBuy

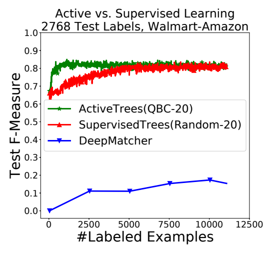


(c) BeerAdvocate-RateBeer

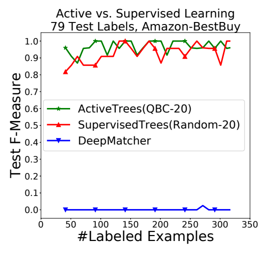


(d) BuyBuyBaby-BabiesRUs

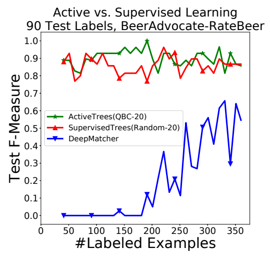
Figure 15: Tree Ensembles on Magellan/DeepMatcher Datasets *(Noisy Oracles, Progressive F1)*



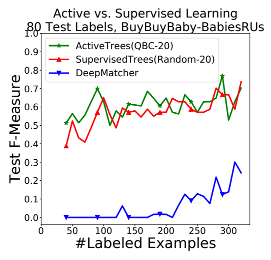
(a) Walmart-Amazon



(b) Amazon-BestBuy



(c) BeerAdvocate-RateBeer



(d) BuyBuyBaby-BabiesRUs

Figure 16: Active vs. Supervised Learning on Magellan/DeepMatcher Datasets *(Perfect Oracles, 20% Test Labels)*

of train and test splits which may vary from our test set that happens to be the entire set of post-blocking EM pairs comprising both labeled and unlabeled examples. The purpose of Table 2 is thus to contrast the best F1 results from the related work with those from our benchmark. For an experimental comparison of active learning with the state-of-the-art supervised learning, see Fig. 16. Among the supervised learning approaches in Table 2, we mention results from DeepMatcher (Mudgal et al. [23]), Singh et al. [29] and Köpcke et al. [21]. While Mudgal et al. [23] use deep learning, the latter two compare rule-based models with machine learning models. However, the results we report here are not necessarily from those approaches but also from their contenders who achieved the best results on these datasets. For instance, the best result from Singh et al. [29] is from supervised random forests. Likewise, some of the best results in Kasai et al. [18] are from their implementation of supervised deep learning models that they try to surpass through transfer learning. Mozafari et al. [22], Corleone [14], Waldo [32], Whang et al. [37] and Chai et al. [7] use crowdsourcing approaches which are slightly different from active learning as they rely on the crowd for labels unlike active learning which assumes the presence of experts for labeling.

6.2 Experiments with Noisy Oracles

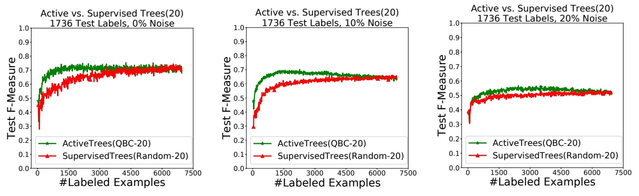
In order to model crowd-sourced scenarios, we use an imperfect Oracle which perturbs the original label with a fixed probability whenever it is asked to label an example. We vary the noise from 10% to 40% following the DeepMatcher [23] settings. It should be noted that we always perturb the original label whenever the imperfect Oracle generates a random probability that falls within the noise percentage threshold. This is a harsher criterion than real-world crowdsourced settings which regulate the noisy labels using techniques such as majority voting and label inference. Each F1-score observed with a noisy Oracle is averaged over 5 random runs using distinct random seeds to account for the randomness and ensure experimental reproducibility.

Our results on the Abt-Buy dataset in Fig. 14 show that tree ensembles produce a near-perfect F1-score using a perfect Oracle, and their performance degrades gracefully with increasing noise percentages (Fig. 14a). Tree ensembles have a relative advantage until 20% noise beyond which their F1 equalizes to that of other classifiers. In contrast, the other classifiers do not even come close to near-perfect F1-scores on perfect Oracles. While linear SVMs show a quick drop beyond 10% noise (30% and 40% overlap in Figs. 14c & 14d), neural networks do not suffer a steep decline in F1-scores at higher noise percentages because of regularization techniques such as drop-out and batch normalization. We do not present results of rule-based classifiers as they produce low progressive F1-scores even with perfect Oracles.

On similar lines, we notice from our experiments on the Magellan/DeepMatcher datasets in Fig. 15 that using the 0% noisy (perfect) Oracle, tree ensembles of size 20 produce high progressive F1-scores close to 1.0 from early on with as few as 100 labels on Amazon-BestBuy and Beer datasets (Fig. 15b, 15c). However, the convergence on Walmart-Amazon and BabyProducts (Fig. 15a, 15d) happens only after 2500 and 300 labels respectively indicating that these are indeed challenging datasets. Upon higher noise percentages, the gradual drop in the F1-scores with an increase in labeled examples is a testimony to the fact that crowd-sourcing in practical scenarios warrant a much earlier termination and error correction techniques such as majority voting. However, the sweet spot in terms of when to terminate active learning in such scenarios may differ across datasets. For instance, while the progressive F1-scores show a monotonically declining F1-score curve on Walmart-Amazon, Amazon-BestBuy and Beer datasets, they show monotonically increasing F1-scores on the Baby Products dataset in Fig. 15d.

Comparison with Supervised Learning: We conduct these experiments following the conventional train-test splits of supervised learning where example selection is done out of a training set containing 80% of post-blocking examples and the evaluation is on a held-out test set of 20% of the tuple pairs which never participate in example selection. In Fig. 17, we compare active learning against supervised learning on Abt-Buy using ensembles of 20 trees upon various imperfection levels of an Oracle. In each iteration, while active learning uses learner-aware QBC to label examples that lead to highest labeling disagreement (entropy) among the 20 decision trees, supervised learning picks random examples in each iteration. The results show that the former outperforms the latter within the first few iterations while achieving test F1-scores comparable to those that supervised learning achieves after training on the entire set of 80% training examples. This difference between supervised and active learning is insignificant at 20% noisy Oracle (see Fig 17c).

Comparison with DeepMatcher [23]: Fig. 16 compares tree ensembles against a state-of-the-art supervised learning approach, DeepMatcher, upon perfect Oracles. We ran DeepMatcher with the same settings in [23] by dividing the labeled examples into 3:1 (train to validation set size ratio)

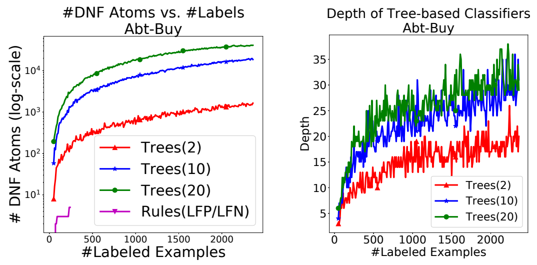


(a) 0% Noisy Oracle **(b) 10% Noisy Oracle** **(c) 20% Noisy Oracle**

Figure 17 Active vs. Supervised Trees *(Abt-Buy, 20% Test Labels)*

and evaluating it on the same 20% test set as tree ensembles. Similar to supervised tree ensembles, DeepMatcher picks a random set of unlabeled examples to label in each iteration. While we did not notice a significant variation among the test F1-scores from different executions of tree ensembles, we noticed a standard deviation of 0.002 (Amazon-BestBuy), 0.016 (Walmart-Amazon), 0.06 (Baby Products) and 0.125 (Beer) across 5 runs of DeepMatcher over all the learning iterations. Hence we report an average F1-score across these 5 runs in Fig. 16. The results confirm that active learning using random forests requires fewer labels to achieve significantly higher test F1-scores than supervised learning. While supervised and active tree ensembles perform similarly on the smaller datasets, DeepMatcher, requires the entire 80% training labels to achieve its best test F1-scores.

6.3 Interpretability: Rules vs. Trees



(a) #Atoms (Trees vs. Rules) **(b) Tree Ensemble Depth**

Figure 18: Interpretability Experiments

While model interpretability has been established to be crucial for supervised learning-based EM by earlier works such as Singh et al. [29], it is also important for active learning. In the case of supervised learning, interpretable models are used for explainability purposes in order to understand why a particular model produces higher quality of matches than a different model and also for debugging purposes to reduce false positives and false negatives, and thereby enhance precision and recall. While all these benefits also exist for active learning, a direct usage of interpretable models is to decide whether or not to accept a model into the active ensemble in a learning iteration and when to terminate active learning under the absence of ground truth. In this section, we contrast the #atoms in rule DNFs learned by

LFP/LFN with those of the DNF formulae obtained using random forests. We convert the trees learned by random forests into DNF formulae by traversing the path from the root of the tree until all the leaf nodes whose predicted label is 1 or *matching*. The path turns into a conjunction of rule-based predicates, and the disjunction or union of all such formulae leads to a DNF. We do not further optimize the DNFs into concise Boolean formulae unlike Singh et al. [29] as the latter may seem concise but need to be mentally unrolled into DNFs by a human. This is because, DNFs are more intuitive to a human. It is therefore possible that there are overlapping atoms across different conjunctive predicates in a DNF and they are counted with repetition to compute #atoms for both rules and random forests. As mentioned in Section 3, an atom can be defined [29] as a DNF predicate or a similarity function evaluated on a pair of attributes from two records and compared against a numerical threshold. We can observe from Figs. 18a & 18b that # DNF atoms in the learned trees as well as their depths increase with more active learning iterations, since larger tree ensembles contain more atoms than the smaller ones. The depth of a tree ensemble is the maximum among the depths of all the trees in the random forest. We can notice from Fig. 18a that rules have significantly fewer atoms than random forests on all the datasets and are thus easily interpretable by a human.

Following is the ensemble of rules learned by LFP/LFN active learning heuristic for the Abt-Buy dataset. Each of these rules has a test precision $\geq$ 0.88 and is accepted into the ensemble at a distinct iteration. Similar concise DNF rule ensembles were obtained on other datasets as well. We do not present the DNF rules for trees as they are prohibitively large.

Abt-Buy (# DNF Atoms = 5):

***Rule 1*: Abt.price = Buy.price**

$\wedge$ JaccardSim(Abt.name, Buy.name) $\geq$ 0.4

$\vee$

***Rule 2*: JaccardSim(Abt.name, Buy.name) $\geq$ 0.7**

$\vee$

***Rule 3*: JaccardSim(Abt.name, Buy.name) $\geq$ 0.6**

$\wedge$ JaccardSim(Abt.description, Buy.description) $\geq$ 0.1

6.3.1 LFP/LFN vs. QBC (Rules) on Social Media Dataset. We use an EM dataset from Qian et al. [25], where the goal is to match 467,761 employee records from a large enterprise to a set of 50M user profiles from a social media platform. The attributes comprise name, location, email address, occupation, gender and a URL to the personal homepage for each user profile. We use this dataset to compare the learner-agnostic QBC with committee sizes ranging from 2 to 20, against the learner-aware heuristic of LFP/LFN on rule-based classifiers. In the absence of ground truth for this real-world dataset, we evaluate example selection strategies indirectly based on the actual rules they produce. Each learned rule is interpreted by a human expert and manually validated by labeling adversarial examples (i.e., LFPs). Once accepted by the human expert, it can be reasonably assumed that they reached high precision. Thus, the quality of a selection strategy is determined by the number of manually accepted (validated) rules along with their aggregate number of predicted matches (called coverage) on the dataset. Since active learning on this dataset requires human validation of the model at each step, it is essential that the model used is interpretable. Therefore, we do not conduct this experiment with the remaining learners. From Fig. 19, we note that LFP/LFN performs comparably to larger bootstrap committees of sizes 10 and 20 on coverage and # valid rules, respectively, while being 1.9x and 4.7x faster w.r.t. the total user wait time (across all iterations) which includes rule learning, rule execution and example selection times. QBC of committee size 2 is faster than LFP/LFN but produces fewer valid rules with less coverage. Fig. 19 also plots average user wait time taken to learn a valid rule, average user wait time per iteration and # iterations.

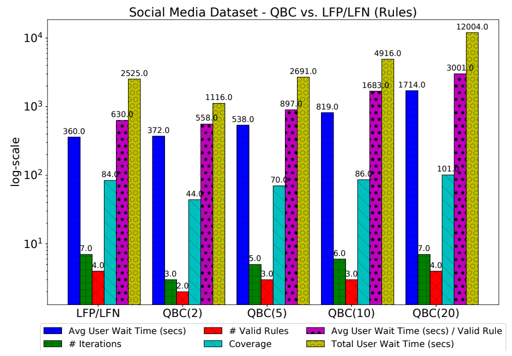


Figure 19: QBC vs. Heuristic Strategies (Rules)

7 CONCLUSION

In this paper, we proposed a unified active learning benchmark framework that can mix-and-match several learners with multiple example selectors for entity matching (EM). Using the framework, we found that active learning upon learner-aware ensembles of tree-based models achieves close to perfect progressive F1-scores on all the public EM datasets we experimented with. Our best active learning methods require fewer #labels for a convergent F1-score than their supervised learning counterparts up until 10% labeling noise and also surpass a state-of-the-art supervised learning algorithm on perfect Oracles. We also found that tree-based learners achieve high quality at the expense of interpretability and applications where concise, highly precise EM rules are required may still resort to rule-based learning. Our experiments on a real-world social media dataset lacking ground truth emphasize the need for interpretable models which are manually validated in each active learning iteration.

REFERENCES

[1] [n.d.]. Simmetrics Java Library. https://github.com/mpkorstanje/simmetrics.git.

[2] [n.d.]. SystemML. https://systemml.apache.org/.

[3] [n.d.]. Weka. https://cs.waikato.ac.nz/ml/weka/.

[4] A. Arasu, M. Götz, and R. Kaushik. 2010. On Active Learning of Record Matching Packages. In *SIGMOD*. 783–794.

[5] K. Bellare, S. Iyengar, A. Parameswaran, and V. Rastogi. 2012. Active Sampling for Entity Matching. In *KDD*. 1131–1139.

[6] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. 2009. Importance Weighted Active Learning. In *ICML*. 49–56.

[7] Chengliang Chai, Guoliang Li, Jian Li, Dong Deng, and Jianhua Feng. 2016. Cost-Effective Crowdsourced Entity Resolution: A Partial-Order Approach. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16)*. ACM, New York, NY, USA, 969–984. https://doi.org/10.1145/2882903.2915252

[8] David Cohn, Les Atlas, and Richard Ladner. 1994. Improving Generalization with Active Learning. *Machine Learning* (1994), 201–221.

[9] Sanjib Das, AnHai Doan, Paul Suganthan G. C., Chaitanya Gokhale, Pradap Konda, Yash Govind, and Derek Paulsen. [n.d.]. The Magellan Data Repository. https://sites.google.com/site/anhaidgroup/projects/data.

[10] Sanjib Das, Paul Suganthan G.C., AnHai Doan, Jeffrey F. Naughton, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, Vijay Raghavendra, and Youngchoon Park. 2017. Falcon: Scaling Up Hands-Off Crowdsourced Entity Matching to Build Cloud Services. In *Proceedings of the 2017 ACM International Conference on Management of Data* (Chicago, Illinois, USA) *(SIGMOD '17)*. ACM, New York, NY, USA, 1431–1446. https://doi.org/10.1145/3035918.3035960

[11] Sanjoy Dasgupta. 2011. Two Faces of Active Learning. *Theor. Comput. Sci.* 412, 19 (2011), 1767–1781.

[12] B. Efron and R. Tibshirani. 1993. An Introduction to the Bootstrap.

[13] Y. Freund, H. Seung, E. Shamir, and N. Tishby. 1997. Selective Sampling Using the Query by Committee Algorithm. *Machine Learning* 28, 2-3 (1997), 133–168.

[14] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F. Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojin Zhu. 2014. Corleone: Hands-off Crowdsourcing for Entity Matching. In *SIGMOD*. 601–612.

[15] Alon Gonen, Sivan Sabato, and Shai Shalev-Shwartz. 2013. Efficient Active Learning of Halfspaces: An Aggressive Approach. *JMLR* 14, 1 (2013), 2583–2615.

[16] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*. 448–456.

[17] P. Jain, S. Vijayanarasimhan, and K. Grauman. 2010. Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning. In *NIPS*. 928–936.

[18] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao li, and Lucian Popa. 2019. Low-resource Deep Entity Resolution with Transfer and Active Learning. In *ACL*.

[19] Asif R. Khan and H. Garcia-Molina. 2016. Attribute-based Crowd Entity Resolution. In *CIKM*.

[20] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeffrey F. Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: Toward Building Entity Matching Management Systems. *PVLDB* 9, 12 (2016), 1197–1208. https://doi.org/10.14778/2994509.2994535

[21] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of Entity Resolution Approaches on Real-world Match Problems. *PVLDB* 3, 1 (2010), 484–493.

[22] Barzan Mozafari, Purna Sarkar, Michael Franklin, Michael Jordan, and Samuel Madden. 2014. Scaling Up Crowd-sourcing to Very Large Datasets: A Case for Active Learning. *PVLDB* 8, 2 (2014), 125–136.

[23] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. ACM, New York, NY, USA, 19–34. https://doi.org/10.1145/3183713.3196926

[24] Tan T. Nguyen and Scott Sanner. 2013. Algorithms for Direct 0-1 Loss Optimization in Binary Classification. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* (Atlanta, GA, USA) *(ICML'13)*. JMLR.org, III–1085–III–1093. http://dl.acm.org/citation.cfm?id=3042817.3043058

[25] Kun Qian, Lucian Popa, and Prithviraj Sen. 2017. Active Learning for Large-Scale Entity Resolution. In *CIKM*. 1379–1388.

[26] Sunita Sarawagi and Anuradha Bhamidipaty. 2002. Interactive Deduplication Using Active Learning. In *KDD*. 269–278.

[27] Burr Settles. 2009. *Active Learning Literature Survey.* Technical Report. University of Wisconsin-Madison.

[28] H. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Workshop on COLT*. 287–294.

[29] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Synthesizing Entity Matching Rules by Examples. *PVLDB* 11, 2 (2017), 189–202. https://doi.org/10.14778/3149193.3149199

[30] S. Tejada, C. Knoblock, and S. Minton. 2001. Learning Object Identification Rules for Information Integration. *Inf. Syst.* 26, 8 (2001), 607–633.

[31] Simon Tong and Daphne Koller. 2001. Support Vector Machine Active Learning with Applications to Text Classification. *JMLR* 2 (2001), 45–66.

[32] V. Verroios, H. Garcia-Molina, and Y. Papakonstantinou. 2017. Waldo: An adaptive human interface for crowd entity resolution. In *SIGMOD*.

[33] N. Vesdapunt, K. Bellare, and N. Dalvi. 2014. Crowdsourcing algorithms for entity resolution. In *PVLDB*.

[34] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. *PVLDB* 5, 11 (2012), 1483–1494.

[35] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. 2013. Leveraging transitive relations for crowdsourced joins. In *SIGMOD*.

[36] S. Wang, X. Xiao, and C. Lee. 2015. Crowd-Based Deduplication: An Adaptive Approach. In *SIGMOD*.

[37] S. Whang, P. Lofgren, and H. Garcia-Molina. 2013. Question selection for crowd entity resolution. In *VLDB*.