Proceedings ACS (2019) 75-93

A Heuristic Strategy for Identifying Misclassified Data Using Classification Labels

Justin Li JUSTINNHLI@OXY.EDU
Occidental College, 1600 Campus Road, Los Angeles 90041

Abstract

Cognitive architectures have increasingly incorporated statistical classifiers as components for long-lived agents. As these machine learning algorithms receive more training data over the agent's lifetime, however, prior results stored in long-term memory may become incorrect, leading to false positives and negatives during memory retrieval. This paper proposes a heuristic strategy that can efficiently identify such misclassified data, by leveraging semantic information inherent in the classification labels. The key insight is that a classifier-independent distance function can be defined on the labels, that correlates with false positive classifications. We show that this strategy can be applied to color naming, to image recognition, and to text classification, in all cases outperforming baselines. Implemented in a cognitive architecture, the heuristic halves the number of re-classifications required to retrieve the desired knowledge, thus building on and complementing the existing memory mechanisms.

1. Introduction

Over the last decade, the cognitive systems community has advanced the role of long-term declarative memory in agent architectures. Distinctions such as semantic and episodic memory, and mechanisms such as partial matching and spontaneous retrieval, have been developed. These memory systems are used by long-lived agents to store knowledge acquired from its accumulated experience and from instruction from human collaborators. At the same time, complex domains such as cognitive robotics have pushed architectures to tightly integrate statistical machine learning techniques, which convert low level visual, audio, and textual data into symbolic structures for agent reasoning. As agents operate in the environment and acquire new percepts and faces new task demands, these classifiers and models must be fine-tuned to increase accuracy, or replaced entirely to produce more nuanced output. However, the question of how this learning process interacts with the existing memory mechanisms remains an open question, and the coupling of statistical and symbolic systems gives rise to unique challenges.

One such challenge is that of *knowledge obsolescence*, where previously learned knowledge is no longer accurate in light of new information. As classifiers a re r efined, it s pr evious ou tput that is stored in long-term memory may become outdated, incongruent with the output of the updated classifier. D ata t hat w as p reviously c lassified ma y, wi th mo re tr aining da ta, be di scovered to have been classified i n correctly; a n d y e t, i n l o ng-term s y mbolic m e mory, t h e p r evious l a bel persists. Misclassifications of this kind may lead to false positives for the old label and false negatives for

the new label, and may indirectly introduce additional error if used for future learning. At the same time, for long-lived agents with a wealth of perceptual data stored in memory, the number of misclassifications may be small, making it difficult and computationally expensive to identify.

In this paper, we propose an efficient strategy for identifying these misclassified data. The key insight is that the symbolic labels — such as the color names or the objects being recognized — contain semantic information that could be leveraged to predict how a datapoint may be misclassified. The agent can then use this heuristic to identify such data and reclassify them as necessary. This strategy is general in that it works with high-dimensional data in multiple modalities and with different classifiers, and can take advantage of heuristics unrelated to the classifier. Furthermore, this strategy allows the agent to dynamically tradeoff between accuracy (identifying all misclassified datapoints) and computational expense (the amount of data to search through). The strategy can therefore be embedded within a cognitive architecture as a generic procedure for memory search that builds on top of the architectural memory mechanisms.

The main contributions of this paper are:

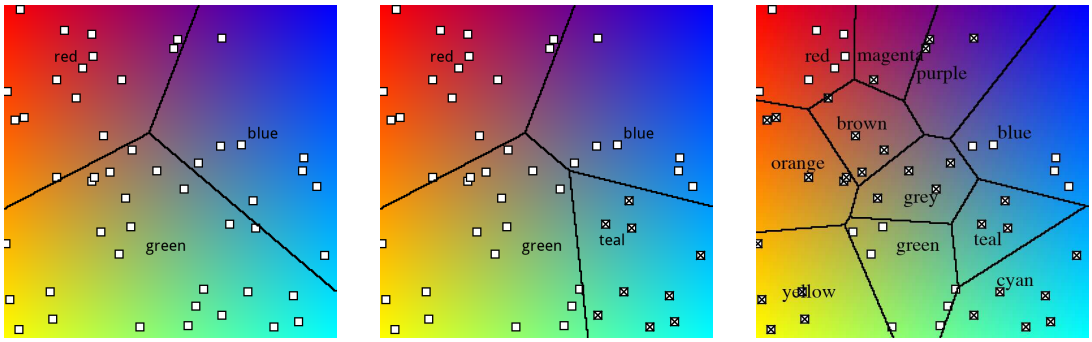
- Formalizing the problem of knowledge obsolescence due to incremental classifier improvement;
- Defining and evaluating a class of heuristic strategies for identifying misclassified datapoints; and
- Showing that such a strategy can be implemented in a cognitive architecture and out-performs the naive approach.

The rest of the paper is laid out as follows. Since we are specifically focusing on how incrementally trained machine learning techniques may lead to knowledge obsolescence, the next two sections describe our problem formulation and approach in machine learning terms. We then present three experiments in Section 4 to show that across visual and textual data, meaningful semantic information can be found in the labels to guide the identification of misclassified data. In Section 5, we return to the integration with architectural memory mechanisms by implementing our strategy in a Soar agent in an interactive-task-learning-like domain. We conclude with a discussion of related work and open questions.

2. Defining Knowledge Obsolescence

While many types of knowledge obsolescence exist, we focus on the interaction between statistical machine learning classifiers and the symbolic memory systems of a cognitive architecture. This paper is specifically interested in how the symbolic output of a classifier, stored in memory, may become incorrect as the classifier is updated. When the classifier has changed, the same input may now produce different output, which does not match the stored symbol, rendering that piece of knowledge obsolete.

Formally, a classifier can be defined as a function $C : X \rightarrow Y$ that takes data as the input and returns a symbolic label as the output. The domain of this classifier X could be data from the agent's perceptual system, such as raw color data from a camera, with the range Y being a color name. Note



(a) (b) (c)

Figure 1: Illustration of knowledge obsolescence as a classifier grows from 3, to 4, to 11 labels. Datapoints are in white, with misclassified datapoints marked with an X.

that this definition does not depend on the specific data or label, nor does it specify the algorithm used for the classifier; this makes it amendable to other problems such as object recognition and text categorization. Section 4 will show that the same framework applies to different classifiers on different types of data.

We define knowledge obsolescence as when learning occurs and the classifier is replaced by a new classifier $C_+ : X \rightarrow Y_+ \supset Y$, such that there exists datapoints $x \in X$ for which $C(x) \neq C_+(x)$. Since C and C_+ have the same domain but the range Y_+ is a superset of Y, there must necessarily be datapoints whose labels have changed between the two classifiers. If these datapoints and its old label are stored in the long-term memory of an agent, that stored knowledge will become inaccurate and obsolete, leading to difficulties for retrieval.

As a concrete running example, consider an interactive robot that obtains RGB information about an object from its camera, and must then associate a color name to identify the object to its human collaborators. During its operation, the agent encounters many objects, applies the classifier to determine its color, and stores that information in symbolic memory. While the agent may initially only start with basic colors such as BLUE and GREEN (Figure 1a), over time it might learn more nuanced labels such as TEAL, and update its classifier to output these new labels (Figure 1b). Especially in an interactive setting, human collaborators and instructors may use these more-specific labels in communication, and in turn expect the agent to use them in its responses. Objects that were encountered before this learning occurred, however, would still be labeled BLUE, when the updated classifier labels them as TEAL. The agent may therefore fail to identify the TEAL object that its human collaborators are referring to, while consistently mis-communicating on which object it may require assistance.

To constrain knowledge obsolescence problem, we consider three ways in which the classifier may be updated. First, the agent could accumulate additional data, which could be used to further train the classifier. Since this would shift the decision boundaries, datapoints that are near those boundaries could be misclassified. In the running example, the agent might be told that an RGB

value on the BLUE/GREEN boundary is in fact BLUE; this would cause nearby datapoints to also be labeled BLUE, even if they were previously labeled GREEN.

Second, the agent could learn additional labels that the classifier must output, as suggested in the running example with TEAL. Initially, the agent may only know of the labels RED, GREEN, BLUE (Figure 1a). The decision boundary is show in black, with the stored datapoints represented as white squares. The agent is then introduced to the label TEAL, leading to a new classifier with decision boundaries as depicted in Figure 1b. The stored datapoints are not reclassified, however, and as a result, a subset of the datapoints remain labeled as BLUE and GREEN, even though they should now be classified as TEAL; these datapoints are marked with an X. As the agent learns more new labels (Figure 1c), a larger portion of datapoints will be misclassified.

The third and final way in which agent learning could influence a classifier is the removal of labels. One can imagine a LIGHT-BLUE label no longer being relevant, and thus excised from the classifier. Although this is a potential consequence of agent learning, we consider it less likely than the first two scenarios. This paper focuses specifically on the second case, the monotonic increase in the range of labels, although our approach will also apply to the first case of shifting decision boundaries.

3. A Heuristic Approach

The problem of identifying misclassified data is challenging for several reasons. First, there may not be additional information about the data outside of the its existing label, providing little information to work with. Second, the number of datapoints may be large, especially if they are the accumulated percepts from a long-living agent. The naive approach of re-classifying all datapoints in the agent's memory is computationally expensive, especially as the majority of datapoints may not be misclassified. Finally, a task may not require the agent to identify all misclassified datapoints. Instead, the agent may choose to trade off label accuracy for reducing the computational resources needed. We are therefore interested in the problem of efficiently identifying these misclassified datapoints in a way that allows for this tradeoff. Specifically, given a new label $y_+ \in Y_+$, we would like to find as many misclassified datapoints by re-applying the classifier to as few datapoints as necessary.

Within the minimal framework we defined, the agent only has three pieces of information: the old labels that have been in use, the new labels that have been learned, and the stored datapoints and their potentially incorrect labels. Given no other knowledge, the only information that could inform which datapoints are misclassified are the labels themselves. That is, given a new label, the goal is to determine which old label is most likely to contain misclassifications, then apply the new classifier to datapoints with those labels. This process can then be repeated for the remaining old labels, in essence ranking the old labels by decreasing number of misclassified datapoints. This would ensure that the most misclassified datapoints are identified while only reclassifying the fewest datapoints overall.

Formally, the false negatives of a new label $y_+ \in Y_+$ are the datapoints which should have new label y_+ under the new classifier C_+, but are instead stored as a different label under the old classifier C. We can define a function $m_{y_+}(y)$ that gives the number datapoints that have old label

y but should have new label y_+:

$$m_{y_+}(y) = |\{x : C(x) = y \neq C_+(x) = y_+\}|$$

To identify all false negatives of y_+, the optimal order is to process datapoints by their old labels in decreasing order by m_{y_+}. We define this optimal ground-truth ranking as a function $\sigma_{y_+} : Y \to \mathbb{N}^+$ that takes an old label y and outputs its rank (a positive integer), where rank 1 is the old label with the most misclassified datapoints, and should therefore be reclassified first. A label should have a lower rank if and only if it has more misclassified points:

$$\sigma_{y_+}(y_i) < \sigma_{y_+}(y_j) \iff m_{y_+}(y_i) > m_{y_+}(y_j) \qquad \forall y_i, y_j \in Y$$

Since the ground-truth misclassifications are not known to the agent, it is not possible to directly calculate this ranking. However, this allows us to formulate the misclassification identification problem as one of *approximating* σ_{y_+} with $\hat{\sigma}_{y_+}$. Note that this is an approximation of the *rank* of each old label (with respect to a new label), and not the *number* of misclassified datapoints with that label. This not only reduces the space of approximations to permutations of Y, but also decouples the approximation from the details of the classifier. This avoids the challenging task of determining how classifier parameters might influence the misclassification of datapoints.

The key insight of this paper is that the symbolic labels often contain semantic information that can serve as a heuristic for the rank. In particular, we hypothesize that the number of y_+ datapoints misclassified as y is inversely proportional to some distance metric d on the labels y_+ and y. This distance metric then allows us to define $\hat{\sigma}_{y_+}(y) \propto 1/d(y_+, y)$.

As a concrete example, consider the new label TEAL in the color naming classifier (Figure 1b). Although we cannot immediately determine which datapoints would belong in this new label, we know that TEAL is more similar to BLUE (or GREEN) than it is to RED. Or, conversely, the distance between TEAL and RED is larger than the distance between TEAL and BLUE. An approximate ranking of the old labels would reclassify all BLUE datapoints first, and only process RED datapoints if warranted by the task and justified by the computational cost. The heuristic strategy therefore enables the agent to prioritize datapoints that are likely to be misclassified, allowing for more efficient and accurate retrieval from memory.

4. Modality Experiments

In this section, we examine whether such a distance function can be defined in three different domains: for color naming, for object recognition, and for text classification. Although these experiments are not within the context of an agent and its memory systems, these experiments are important to validate the broad applicability of our approach. Once the effectiveness of the heuristics are established, we show that it can be integrated into a cognitive architecture in Section 5.

Each experiment uses significantly different classifier algorithms on different data. The goal of these experiments is to demonstrate that in all three cases, an appropriate distance function can be defined and that it can be used to efficiently identify misclassified datapoints. Additionally, we are interested in the performance of this approach under two conditions. First, since the computational expense comes from potentially large amounts of data, we are interested in whether this heuristic

Heuristic Rank	Ground-truth Rank	Misclassified Points	Cum. % Found
1	1	180	87.80
2	3	2	88.78
3	3	2	89.76
3	4	1	90.24
4	2	20	100.00

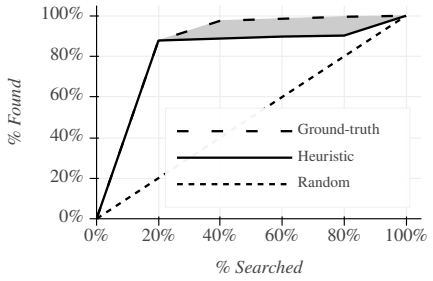


Figure 2: An example approach compared to the random and the optimal ranking. This heuristic ranking has a mean scaled regret of 0.07.

scales to large datasets. Similarly, as an agent gains experience, it will likely already have many labels on existing data; an ideal heuristic would also scale to large numbers of labels. Second, we are interested in whether the performance of this heuristic changes depending on the accuracy of the original classifier. Since less accurate classifiers have more misclassified datapoints, even before new labels are introduced, the inaccuracy may carry over to heuristic and decrease its accuracy in turn.

We use the same methodology in all three experiments. Given a labeled dataset in the domain, a subset of the labels is chosen and is used to train a classifier. The classifier is then presented with the complete dataset, including data whose labels were omitted from the training. For each of the unused labels, a heuristic is then used to rank the old labels. This method of training the initial classifier with only a subset of the labels ensures that some data will be misclassified, and that the number of new and old labels can be varied.

4.1 Evaluation

We are primarily interested in the efficiency of our approach in identifying misclassified datapoints, by finding as much obsolete data in as few re-classifications as possible. Figure 2 graphically depicts this objective with demonstration data. The x-axis shows the percentage of datapoints searched, while the y-axis shows the percentage of misclassified datapoints found. An efficient strategy would find a large proportion of misclassified datapoints while only searching a small proportion of datapoints, leading to a line towards the upper-left corner of the plot.

In addition to the performance of the heuristic strategy (solid line), Figure 2 also depicts two baselines. The top-most, dashed line is the ground-truth oracle, which corresponds to σ_{y+}, and is the best possible ranking of old labels. Note that contrary to expectation, the "best" result is generally not a step function that jumps immediately to 100%, where the agent only applies the classifier to the misclassified datapoints. Such a result would only be possible if all misclassified datapoints were in the same existing label, when in most cases they would be more distributed, such as how TEAL datapoints are incorrectly classified as both BLUE and GREEN. The bottom-most, dotted line represents the brute-force or random strategy, which uses an arbitrary permutation of the old labels as the ranking. On average, the random strategy will find misclassified points in proportion to the

percentage of points searched. Together, these two baselines provide the upper and lower bound for measuring our heuristics.

We quantify the performance of a heuristic ranking by measuring its *regret*. Geometrically, this regret is equivalent to the area between the heuristic strategy and the ground-truth, shown shaded in Figure 2. In the simple case, we can calculate regret between two total orderings using the generalized Kendall's tau metric with element weights (Kumar & Vassilvitskii, 2010). This regret is given by:

$$r(\hat{\sigma}_{y_+}) = \sum_{\sigma_{y_+}(y_i)<\sigma_{y_+}(y_j)} \begin{cases} m_{y_+}(y_i) - m_{y_+}(y_j), & \text{if } \hat{\sigma}_{y_+}(y_i) > \hat{\sigma}_{y_+}(y_j) \\ 0, & \text{otherwise} \end{cases}$$

Given the two orderings σ_{y_+} and $\hat{\sigma}_{y_+}$, Kendall's tau considers every pair of ranks in σ_{y_+}. If the other ordering has that pair in reverse, a penalty of the difference in misclassifications is added. Intuitively, this metric measures the number of inversions of the heuristic ranking compared to the ground-truth ranking, weighted by the difference in the number of misclassifications. This captures the intuition that the more mis-ranked a label is, the longer before its misclassified datapoints will be identified.

Kendall's tau only applies to totally-ordered rankings. In reality, however, both the ground-truth and heuristic rankings are *bucket orders* (Fagin et al., 2006), where each "bucket" contains items of the same rank, and there is a total ordering between buckets. Unlike general partial orderings, all items in a bucket order can be compared, with items in the same "bucket" having no preference for ordering. For the ground-truth ranking, a bucket indicates that two labels contain the same number of misclassified datapoints ($m_{y_+}(y_i) = m_{y_+}(y_j)$), and there is no difference as to which label should be reclassified first. For the heuristic ranking, a bucket indicates that two existing labels are equi-distant from a new label ($d(y_+, y_i) = d(y_+, y_j)$).

Figure 2 is therefore a simplification — as shown in the table, there are in fact two old labels with the same heuristic rank, and two other labels with the same number of misclassified datapoints (and thus the same ground-truth rank). An accurate Figure 2 should instead have two lines for the heuristic and two lines for the ground truth, which combinatorially leads to four different regret values that must be averaged. Although it's possible to calculate an average Kendall's tau for all total orderings of the ground-truth and heuristic rankings, this is computationally intractable, with a worst case of $n!$ total orderings. Instead, we take advantage of the bucket order to decompose this computation into the sum of inter- and intra-bucket components, both of which have n^2 complexity. Each component of the computation is given in Figure 3.

Since the performance of the ground-truth strategy depends on the classifier and the dataset, we scale regret to $[0, 1]$ to allow for comparison between experiments. The scaling factor used is the maximal regret possible for a classifier and dataset, which corresponds to the regret of the reverse ground-truth ranking. The symmetry of this definition means that the random strategy will always have a scaled regret of 0.5. Finally, because regret is calculated per new label, the regret of a heuristic is the average regret for all new labels, $r(\hat{\sigma}) = \frac{1}{|Y_+|} \sum_{y_+ \in (Y_+ - Y)} r(\hat{\sigma}_{y_+})$. All three experiments below use the mean scaled regret as the metric for performance.

procedure INTRABUCKETREGRET($bckt$)
  $regret \leftarrow 0$
  for y_i, y_j in $bckt$ **do**
    if $m_{y_+}(y_i) \neq m_{y_+}(y_j)$ **then**
      $addend \leftarrow abs(m_{y_+}(y_i) - m_{y_+}(y_j))$
      $regret \leftarrow regret + addend$
  return $regret/2$

procedure INTERBUCKETREGRET($bckt1$, $bckt2$)
  $regret \leftarrow 0$
  for y_i in $bckt1$ **do**
    for y_j in $bckt2$ **do**
      if $m_{y_+}(y_i) > m_{y_+}(y_j)$ **then**
        $addend \leftarrow m_{y_+}(y_i) - m_{y_+}(y_j)$
        $regret \leftarrow regret + addend$
  return $regret$

Figure 3: Regret between Bucket Orders

4.2 Color Domain

This experiment is an implementation of the running example of color naming. Color was chosen as the domain for several reasons. First, the classifier and the resulting boundaries are relatively simple, providing intuition for why the heuristic approach to identifying misclassification works. Second, it presents an example where the heuristic is a measurement directly used by the classifier, and thus is a best case scenario for our approach. Finally, due to the simplicity of the domain, it is easy to generate synthetic data and to scale the number of new and old labels used by the classifier. This allows us to explore the effects of data size and the number of labels on the performance of our approach.

In a color naming task, the goal is to convert an RGB value (three integers between 0 and 255, inclusive) into an English color name. For this experiment, we use a nearest-centroid classifier, with the centroids obtained from the results of an online survey (Munroe, 2010), where respondents were asked to name a particular RGB color. This ensures that the color labels accurately represent human perception, and is therefore representative of what might be used in cognitive robotics. We extracted the top 200 color names from the survey, each of which were used over 1,000 times by respondents; the centroid of the cluster corresponding to a color name is defined by the mean RGB values. The nearest centroid algorithm partitions the RGB space into Voronoi cells, such that for any RGB value, its label is the closest centroid by Euclidean distance. A 2D projection of this classifier is shown in Figure 1. We assume that the agent has access to the centroids, and simply use the Euclidean distance between centroids as the heuristic distance between labels. Since this directly corresponds to how the classifier assigns labels, we expect it to be accurate in identifying misclassified points.

Each trial in this experiment is parameterized by the number of datapoints, the number of old labels $|Y|$, and the number of new labels $|Y_+|-|Y|$; the last two parameters indirectly control the total number of labels. To reflect more common color names being learned first, the labels are chosen from the most popular color names. The lower-level data was uniformly randomly drawn from the RGB space. The results described below are the average of 25 runs for each trial. Note that this experiments adds multiple new labels at the same time, which may be unrealistic for real agents, where such labels may be acquired incrementally and added individually. Nonetheless, learning multiple new labels simultaneously through instruction is not implausible, and this experiment allows us to characterize its effects.

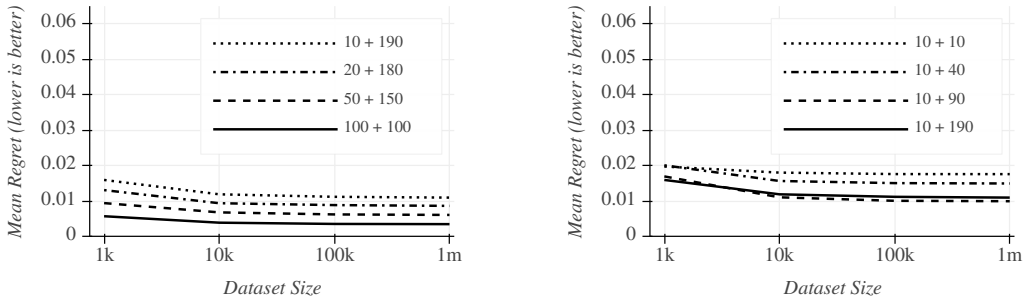


Figure 4: Heuristic regret as the number of (a) old and (b) new labels increase, denoted as "old + new". The leg The x-axis is in log scale.

4.3 Color Domain Results

As expected, because the heuristic ranking is closely related to the classifier, the regret is generally extremely low. The regret values were below 0.05 for all trials, significantly lower than the regret of 0.5 for a random strategy.

Figure 4 shows the regret of the algorithm for different number of precepts and different number of old labels (Figure 4a) and new labels (Figure 4b). The x-axis of both plots show the number of data points from 10^3 to 10^6, on a log scale; the y-axis shows the mean scaled regret. As the number of data points increase (along the x-axis), the regret decreases for all parameter settings. This is in part a result of the distribution of datapoints: as the number of datapoints increase, it is more likely that datapoints will exist in the space where labels have changed. The number of misclassifications would correspond with the size of the space, which is in turn inversely correlated with the distance between centroids. The number of datapoints therefore increases the accuracy with which the heuristic predicts the ground-truth ranking.

Focusing on the effects of increasing the number of old labels (Figure 4a), we see that as the number of old labels increase (indicated by different trend lines), the regret tends to decrease. Two separate phenomena contribute to this trend. First, as the number of existing labels increase, the changes in decisions boundaries caused by the addition of new labels are smaller. Compare the size of the TEAL in Figure 1b to that in Figure 1c: the smaller differences in decisions boundaries means that there will be fewer misclassified datapoints, which are distributed over fewer existing labels. Additionally, the other existing labels also occupy smaller regions. When these labels are ranked by the heuristic, they in effect become a "higher-resolution" features of the space, allowing the heuristic to more accurately specify where misclassified datapoints might be. Together, these two effects lead to decreased regret as the number of existing labels increase.

Conversely, as the number of new labels increase (Figure 4b), the regret tends to increase, although the effect is inconsistent. This is consistent with the intuition that the more the classifier is changed, the less information previous labels can provide. Specifically, as more labels are learned, the boundaries of the new labels interact with each other, causing the datapoints to no longer correspond to their previous labels. This effect can be seen with the BROWN datapoints in Figure 1c

— although the label is closest to RED, its region is also shaped by the ORANGE, MAGENTA, and PURPLE labels, causing it to include space that was originally labeled GREEN. These interactions between labels decrease the accuracy of the heuristic, leading to higher regret.

To summarize this experiment, using the Euclidean distance between centroids in RGB color space as a heuristic, we were able to accurately rank the old labels in which datapoints may be misclassified. The regret of our approach decreases with more datapoints and more existing labels, but increases with more new labels. Although these results are encouraging, the color domain is relatively simple. The choice of classifier, closest-centroid, is intimately related to the distance between centroids, making it obvious that the rank of labels will correspond to the rank of false negatives.

4.4 Image Domain

This second experiment explores the heuristic approach with more sophisticated machine learning methods, namely, with image classification on a neural network. Image recognition is not a task for which symbolic cognitive systems are traditionally well-suited; the integration with a neural network may therefore be necessary and beneficial for certain tasks, such as cognitive robotics. Neural networks present two additional challenges beyond that of the nearest centroid classifier. First, the decision boundaries of a neural network are opaque and less amendable to analysis, with no obvious distance metric between labels. Second, neural networks also introduce the problem of classifier accuracy. This experiment focuses on the exploration of these two issues.

For this experiment, we use the CIFAR-10 dataset, which contain 60,000 images split equally between 10 labels, including vehicles, mammals, and other animals (Krizhevsky, 2009). Each image is 32x32 pixels in full-color, meaning that it is represented as three 32x32 matrices for each of the red, green, and blue channels, for a total of 3,072 features. Since we are not striving for the highest possible accuracy, we use an off-the-shelf network architecture[1] from the Keras library, using the TensorFlow backend (Chollet et al., 2015; Abadi et al., 2015). The classifier is an 8-layer convolutional neural network, with four convolutional layers with rectified linear (ReLU) activation, two max-pooling layers, and two fully-connected layers, using softmax for this multi-class domain and dropout for regularization.

As with the previous experiment, we only train the network on a subset of labels. In this case, we took all combinations of three labels to create $\binom{10}{3} = 120$ trials. Each network is therefore trained on 15,000 images, 5,000 images from each label, in batches of 32 images, for 200 epochs; the data is augmented with random translations and horizontal reflections. An additional 1,000 images from each class is used as the validation set; the accuracy from this validation set is used below the characterize the performance of the heuristic. To simulate agent learning, the network is asked to classify 1,000 images from each of the 7 unused labels. Unlike the color domain, each label is used independently from the others; this is equivalent to an agent learning a fourth label after being trained on the first three. No additional training of the network network is done with this new label.

1. `https://github.com/fchollet/keras/blob/master/examples/cifar10_cnn.py`

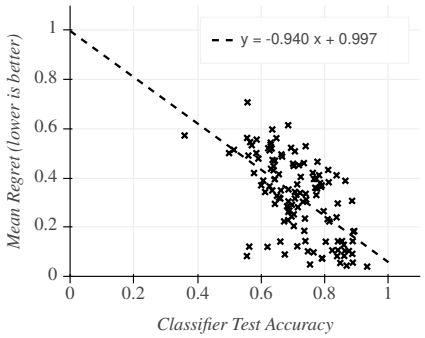


Figure 5: Regret vs. classifier accuracy for CIFAR-10 images. $r^2 = 0.362$

The complexity and opacity of neural networks makes it difficult to extract a useful distance metric. Instead, we used an external source: the UMBEL ontology, a subset of the OpenCyc knowledge base (Bergman & Giasson, 2016; Lenat, 1995). The ontology contains 34,000 concepts, out of which we identified the equivalent concepts for the ten labels. To create a distance metric, we extracted an undirected subgraph using the `subClassOf` relation, and used the shortest graph distance between pairs of labels as the heuristic. Although UMBEL includes some taxonomic relations such as species and genus, this is often not part of the shortest path between labels. For example, `umbel-rc:Dog` is not only a subclass of `umbel-rc:CanisGenus`, but also of `umbel-rc:DomesticatedAnimal`, a category that also includes `umbel-rc:Horse`. DOG and HORSE therefore have a distance of 2, which is shorter than their taxonomic distance. In general, the UMBEL distances do not correspond to evolutionary history, with similarly non-standard categories for vehicles.

4.5 Image Domain Results

Across all combinations of three old labels from CIFAR-10, the mean regret for all new labels is 0.188, with a standard deviation of 0.146. This is significantly lower than the regret of 0.5 for the random strategy, suggesting that the UMBEL graph distance heuristic is effective and efficient at identifying misclassified datapoints despite being completely unrelated to the data and the classifier.

We are additionally interested in the performance of the heuristic as a function of classifier accuracy. Figure 5 plots the regret for each of the 120 classifiers on the y-axis, against the validation accuracy for the three old labels for that classifier on the x-axis. The plot shows a clear relationship between the accuracy of the classifier and the regret of this heuristic: the more accurate the classifier, the lower the regret. This is not surprising, as an inaccurate classifier would lead to datapoints being misclassified in the first place, and therefore bear no relation to the new label. As a concrete example, consider the results from Figure 1 if GREEN was (incorrectly) determined to at be the top of the image, without affecting its ranking with respect to TEAL. The TEAL datapoints in the bottom-right would instead have been misclassified as BLUE, meaning that although GREEN remains ranked first by the heuristic, it no longer contains misclassified datapoints, and prioritizing GREEN would

lead to higher regret. The accuracy of the classifier therefore has a large effect on performance of a heuristic, even if the heuristic is otherwise accurate.

It is worth exploring why ontology graph distance would accurately predict misclassifications. Although the UMBEL ontology has no explicit relationship with the neural network classifier, there is an implicit relationship in the data. The ontology encodes evolutionary and functional characteristics of objects, such as the fact that both dogs and deers are mammals, or that both automobiles and trucks are land vehicles. Objects with similar characteristics are placed close to each other in the ontology, and would therefore have a short graph distance. At the same time, these functional characteristics are also indirectly expressed in images of objects, for example by having common backgrounds (e.g., roads) or common features (e.g., legs), making it more likely that the neural network will confuse dogs with deers than with birds. That is, images of objects with similar functional characteristics will tend to be more similar than images of objects that do not share such characteristics. As a result, the graph distance in an ontology can serve as a heuristic for how likely images will be misclassified.

In summary, using semantic heuristics for identifying misclassified points works even when the data is high dimensional and the classifier is an opaque neural network. However, the accuracy of the classifier plays a large role in the performance of the heuristic.

4.6 Text Domain

The final experiment in this section moves from visual to textual data.

For this experiment, we use the newsgroup dataset from scikit-learn (Pedregosa et al., 2011). The dataset contains 18,000 Usenet newsgroup posts on 20 different topics, which also serve as the labels. Given the text of a particular post, the goal of this task is to identify which group the text belongs to. As with the image experiment, we use an existing classification pipeline for this task.[2] The pipeline consists of three steps, which first converts each post into a vector of word counts, then performs TFIDF (Term Frequency / Inverse Document Frequency) to extract features, before finally applying a linear classifier. The parameters were determined from a grid search and evaluated on a 20-class classifier.

To obtain a heuristic for the newsgroup labels, we take advantage of the newsgroup hierarchy itself. Since labels such as `comp.sys.ibm.pc.hardware` represent paths, we can define a distance function using the tree distance between two labels. For example, even though `comp.sys.ibm.pc.hardware` and `comp.sys.mac.hardware` share the final component of their paths (`hardware`), they would still have a distance of 5 because of the tree-structured hierarchy.

4.7 Text Domain Results

Similar to the neural network experiment, we trained classifiers on all combinations of 3 labels, for a total of $\binom{20}{3} = 1140$ trials. Across these trials, the mean regret for all new labels was 0.26, with a standard deviation of 0.10; again, this is below the regret of 0.5 from a random strategy. The success

2. `https://scikit-learn.org/stable/auto_examples/model_selection/grid_search_text_feature_extraction.html`

of this heuristic is likely due to the name of each newsgroup having a strong correlation with the words that are used in the posts. Newsgroups under `comp.sys.*` will likely share common words about computers, unlike other groups such as `soc.religion.christian`. Using TFIDF as a feature ensures that the most unique words for each post are identified, and the linear classifier allows shared weights for words that are indicative of a label. The tree-distance on these labels therefore serve as an acceptable approximation of the misclassification rankings.

5. Memory Strategy Experiment

The previous section provided evidence that across different types of data, the symbolic labels often contain semantic information that can be used to mitigate the problem of knowledge obsolescence. All three experiments were done outside the agent-context, however, and so does not address how this strategy can be integrated into a cognitive architecture. This next experiment addresses the omission.

In this experiment, we implemented the heuristic strategy in the Soar cognitive architecture (Laird, 2012). Although the strategy should be viable in all architectures that implement the standard model of the mind (Laird et al., 2017), we chose to implement the strategy in Soar. We are interested in the question of how this heuristic strategy interacts with the architectural long-term memory mechanisms, especially give the space in which such mechanisms might differ. Soar is a compelling architecture for this experiment for this reason, as it contains separate semantic and episodic memory systems, which have different mechanisms for retrieval. As the results show, these differences alter the baseline performance of the agent.

To provide fidelity to this experiment, we applied the strategy to an existing interactive tabletop robotics task (Mohan et al., 2012). In this task, a Soar agent controls a robotic arm on a table, on top of which are foam blocks of different sizes, colors, and shapes. The agent is equipped with a Kinect camera, which provides both color and depth data. This low-level data is mediated by three classifiers that provides labels for the size, color, and shape of objects, such as a *large red triangle*; we focus only on the size and color labels in this experiment. Since objects can be occluded by each other, the agent must store the location and description of objects in long-term memory. The interactive nature of the task comes from the agent interacting with a human instructor, who can teach the agent about new colors and sizes. For example, the instructor might select a block and tell the agent "this is an orange rectangle"; this would then be used as a training datapoint for the color classifier. Additionally, the instructor can also ask the agent to identify and interact with blocks through their descriptions, with instructions such as "pick up the large red triangle."

5.1 Declarative Memories in Soar

The Soar cognitive architecture, on which this agent is built, contains two long-term declarative memories. In general, these architectural components are for the storage of factual information about the world and the agent. In Soar specifically, the *semantic* and *episodic* memories serve different roles and are accessed by the agent in different ways. Semantic memory contains facts about the world, and requires deliberation by the agent for knowledge to be stored. In the tabletop robotics task, semantic memory contains linguistic knowledge, for example, that the word "red" is a color.

To retrieve knowledge from semantic memory, the agent must create a *cue* describing the desired features of the piece of information. For example, if the agent was looking for the ID of a block, the agent would query semantic memory for objects have the color red and is triangularly shaped. Semantic memory would find such an object and return all its associated attributes. Importantly, semantic memory only returns objects that exactly matches all attributes cues, or fail if no such object exists.

In contrast, episodic memory contains the contextualized experiences of the agent, with the understanding that the agent may not be able to determine what information may be useful *in situ*. Episodic memory storage therefore occurs automatically and periodically, without deliberation by the agent, and captures the entire agent state. As with semantic memory, retrieval from episodic memory requires the agent to create a cue. Unlike semantic memory, however, if no complete matches exist, episodic memory will return episodes that only *partially* match the cue. For example, a query for "large red triangles" might return an episode where a large red *rectangle* was present instead. This partial matching is done at a symbolic level, with "large," "red," and "triangle" each being weighed equally. Episodic memory does not contain any additional heuristics about the similarity between "red" and "orange", so "large *blue* triangle," "*tiny* red triangle," and "large red *hexagon*" are all equally valid as partial matches to "large red triangle."

5.2 Task Description

To show how misclassification heuristics complement the existing memory mechanisms of Soar, we created an agent that stores the size and color of blocks, as indicated by the instructor. Since the robotics component of this task is outside the scope of this paper, the environment is entirely simulated. Each trial consists of a presentation phase, followed by the quiz phase. During the presentation phase, the instructor presents a new block to the agent at every time step, and teaches the agent the correct size and color labels for that block. After 20 blocks are presented to the agent, the instructor asks the agent to identify the block with a specific size, color, and shape. The agent will then propose different blocks as the answer, until the instructor indicates that the agent is correct.

In this task, knowledge obsolescence occurs because the both the size and color classifiers are being trained as the instructor presents blocks. These two classifiers change in different ways. As with the previous color experiment, colors are labeled with a nearest centroid classifier, and are drawn uniform randomly from RGB space. During the presentation phase, only five colors labels are used; however, the quiz phases uses two additional color labels, meaning that some objects may now be classified differently. The size labels, on the other hand, are scaled to the smallest and largest diameters of the blocks the agent has seen thus far, and are drawn from a Gaussian distribution. What was a "gigantic"' block at the beginning may therefore later be classified as merely "large" when other, bigger blocks are seen. Due to these incremental changes in the classifier, the description of the block asked for by the instructor may not match any block stored in memory.

5.3 Agent Description

The compare the different strategies for identifying misclassified datapoints, we implemented agents that used different strategies for the robotics task. No architectural modifications were necessary, as the strategies merely differ on how and when the long-term memory systems are used. To explore the interaction between memory strategy and memory mechanism, we parameterize the agents by both the memory system used and by the strategy for identifying the desired block. A total of three strategies were implemented:

- Attempting only a retrieval with the **exact** labels. Note that the results depend on the memory used; while semantic memory will attempt to find blocks with those labels and fail otherwise, episodic memory will employ partial matching to return other potential blocks.
- **Exhaustively** enumerating the contents of memory. For semantic memory, this enumeration is dependent on base-level activation and spreading; for episodic memory, this enumeration is in reverse chronological order.
- Using the **heuristic** to find nearby labels, then querying for objects that match those labels exactly. For both sizes and colors, the rank of each label for every other label is loaded into semantic memory. If the agent is asked to identify a "large red triangle", it will retrieve the labels semantically close to "large" and "red", and query for objects with those labels in turn. The sum of the ranks is used as the retrieval order, so the agent would query for "*huge* red triangle" and "large *orange* triangle" first before querying for "huge orange triangle". Since this heuristic contains more information than the partial matching in episodic memory, only complete matches from episodic memory are used.

The exact and exhaustive strategies serve as baselines for comparison. The exact strategy is representative of an agent that assumes no uncertainty or error in its memory systems and in the query by the instructor: either an object of that exact description is found, or there is no object that could be the desired result of the query. The exhaustive strategy, on the other hand, is representative of the naive approach memory search, namely, to try everything until it succeeds. This is equivalent to the brute-force/random strategy in Figure 2, with no additional knowledge of the domain. Note that both the exhaustive and heuristic strategies are guaranteed to find the correct block, since all objects are eventually enumerated.

For evaluation, we look at both the number of objects the agent correctly identifies, as well as the number of guesses needed to do so.

5.4 Results

The results from 1,000 agents with each memory and strategy are presented in Table 1, further separated by whether one or both of the size and color labels are incorrectly labeled.

In general, using label heuristics to search memory leads to fewer guesses than exhaustively enumerating all possibilities. The results for the heuristic strategy is the same for both semantic and episodic memory, since the agent does not take advantage of the partial matching mechanism. The different regrets using the exact match strategy for semantic and episodic memories illustrates how

Table 1: Results from implementing heuristic search in Soar.

Memory	**Strategy**	**One Label Differs** ($n = 778$)		**Both Labels Differ** ($n = 222$)	
		% Correct	Mean Guesses (std.)	% Correct	Mean Guesses (std.)
Semantic	Exact	0%	-	0%	-
	Exhaustive	100%	12.59 (5.92)	100%	15.38 (4.40)
	Heuristic	100%	4.69 (4.64)	100%	7.25 (5.36)
Episodic	Exact	48.8%	1 (0.00)	1.4%	1 (0.00)
	Exhaustive	100%	8.85 (8.32)	100%	16.27 (4.65)
	Heuristic	100%	4.69 (4.64)	100%	7.25 (5.36)

architectural mechanisms might interact with the results of machine learning algorithms. Although semantic memory fails in cases where one or both labels differ, episodic memory has some success with the single-label case, correctly identifying the block in a single attempt in 49% of cases. This is due to the partial matching mechanism — even if one label is obsolete, partial matching could return an object that matches the other label, coincidentally selecting the correct object. Much of this success is due to chance, however, and as more potentially-incorrect cues are added, the ability for episodic memory to cope with obsolete knowledge decreases.

In summary, this experiment showed that the heuristic strategy presented in this paper can be implemented in Soar without needing architectural modifications. The strategy requires fewer memory retrievals to identify the object requested by the instructor, but the amount of relative improvement depends on the degree to which the memory mechanisms allow for uncertainty in the query terms.

6. Related Work

In general, the problem of knowledge obsolescence and identifying misclassification has not been well-studied. Machine learning traditionally operates in a framework where all labels are available during training; reclassification would imply training a completely new classifier over the new dataset and labels. Where learning is incremental, the additional input to the algorithm is often in training data and not in new labels. The closest prior work on reclassification is Sierra & Corbacho (2000), where detecting misclassification is treated as a clustering problem. Standard clustering metrics can be used to detect previously unknown clusters that then serve as new labels, implicitly identifying those datapoints as misclassified. This approach does not leverage external knowledge, however, and since it requires clustering of the complete dataset, it does not have the property of allowing agents to trade off label accuracy with computational cost.

While the literature on reclassification is sparse, the use of symbolic taxonomies for classification is not new. For example, it has been noted that the confusion matrix for a large multi-class (10,000+) image classifier contains block-diagonal structures, that roughly align with whether the image is of an object or an animal (Deng et al., 2010). Further sub-matrices show correlation with the structure of WordNet (Miller, 1995), suggesting that it and other semantic networks may provide meaningful information for efficient classification (or reclassification). This is implied by the how ImageNet provides not only the label for its images, but also a taxonomy of those labels based on WordNet (Deng et al., 2009). Other taxonomies and distances metrics using information retrieval

techniques on Google and Flickr have also been defined and used for classification (Wu et al., 2008). To the best of our knowledge, neither taxonomies nor distance metrics have been applied to the problem of reclassification.

For the cognitive modeling and cognitive architecture communities, this work builds on prior results on memory mechanisms and memory strategies. Although it is well-known that human memory retrieval is strategic and involves multiple signals and processes (Burgess & Shallice, 1996), no such general set of strategies have been developed for cognitive architectures. Even where large knowledge bases are used, the approach has mostly remained at the level of translating a question into terms for an exact match, while allowing for mechanisms such as partial match and spreading (Salvucci, 2015). The possibility of knowledge obsolescence introduces uncertainty into the memory retrieval process, which existing memory systems may not be able to fully mitigate (Li et al., 2016); using other sources of knowledge as a heuristic may allow domain-specific efficiencies that an architectural mechanism cannot provide.

7. Discussion

In this paper, we proposed that agents can use additional semantic knowledge to identify misclassified datapoints, in order to mitigate knowledge obsolescence from incremental knowledge acquisition. We have shown that effective heuristics exist in three different domains, and have explored their performance with large datasets and with decreased classifier accuracy. The implementation of a heuristic retrieval strategy in Soar has also shown that it out-performs a brute-force strategy, without the need for new architectural memory mechanisms.

Several open questions remain, however, the foremost being how a heuristic might be found for any particular domain. We acknowledge that it is unlikely that suitable heuristics exist for all domains — while the ontology graph distance may correlate with misclassified images of dogs and cars, we doubt that they can be used to identify audio recordings from the same objects. That said, some general guidelines could be laid down. First, for certain classifiers such as nearest neighbors, the labels/centroids may directly serve as the heuristic. Although the first experiment used color as the domain, the heuristic did not use any external knowledge about color except for the RGB value of the centroids. This suggests that the same approach can be applied to other datasets, provided an appropriate distance function is used. We suspect the same can to applied to other exemplar-based methods such as Gaussian mixture models.

Less can be said with certainty regarding other datasets and classifiers. Both the image and text experiments used a tree distance, and we suspect that taxonomies and other semantic networks will serve as useful heuristics, provided a causal link between the classification domain and the method of organization. While this may appear to be a narrow claim, many domains of interest have such taxonomies available. The evolutionary phylogenetic taxonomy, for example, is likely to apply to many types of biological data, given the key role of evolution in biology. With the rise of machine-learned ontologies, such as hierarchical topic models (Griffiths et al., 2004), there is reason to believe that the ontologies can then be used to identify misclassified data in new classifiers. We concede, however, that such heuristics are also subject to idiosyncrasies. The newsgroup hierarchy, for example, has computer science (`comp.*`) as a top-level branch, making it equivalent to the

broad branch of `alt.*`; this would skew any distances between technology and (say) geographical topics. Ultimately, the effectiveness of a heuristic for any particular domain can only be determined through empirical experiments.

A separate set of open questions address efficient memory retrieval in a cognitive architecture. Architecture development in the cognitive systems community has focused on low-level memory mechanisms, but has thus far ignored domain-independent patterns of memory use. Although this paper has focused on identifying obsolete knowledge, the use of higher-level representations (such as the symbolic output of classifiers) can be applied to other memory queries. For example, a query for the closest visited location to a coordinate might use the county as a coarse filter. Domains such as geography and color may also allow for a hierarchy of representations, such as how a geographical coordinate is part of a city, a state, as well as a country, and "baby blue" may be described as both "light blue" and "blue". Other structures that exists in the semantics of the data could be exploited by agents for efficient memory retrieval. Future research may examine more complex memory behavior that incorporates problem solving and reasoning into knowledge search.

References

Abadi, M., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org/`. From `https://tensorflow.org/`.

Bergman, M. K., & Giasson, F. (2016). Upper mapping and binding exchange layer (UMBEL) specification. `http://techwiki.umbel.org/index.php/UMBEL_Specification`.

Burgess, P. W., & Shallice, T. (1996). Confabulation and the control of recollection. *Memory*, *4*, 359–412.

Chollet, F., et al. (2015). Keras: Deep learning for python. `https://keras.io`.

Deng, J., Berg, A. C., Li, K., & Fei-Fei, L. (2010). What does classifying more than 10,000 image categories tell us? *Proceedings of the 11th European Conference on Computer Vision: Part V* (pp. 71–84). Berlin, Heidelberg: Springer-Verlag.

Deng, J., Dong, W., Socher, R., Li, L., Kai Li, & Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255).

Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D., & Vee, E. (2006). Comparing partial rankings. *SIAM Journal on Discrete Mathematics*, *20*, 628–648.

Griffiths, T. L., Jordan, M. I., Tenenbaum, J. B., & Blei, D. M. (2004). Hierarchical topic models and the nested chinese restaurant process. *Advances in Neural Information Processing Systems 16 (NIPS)* (pp. 17–24). MIT Press.

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images*. Technical report, University of Toronto.

Kumar, R., & Vassilvitskii, S. (2010). Generalized distances between rankings. *Proceedings of the 19th International Conference on World Wide Web (WWW)* (pp. 571–580).

Laird, J. E. (2012). *The Soar cognitive architecture*. MIT Press.

Laird, J. E., Lebiere, C., & Rosenbloom, P. S. (2017). A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics. *AI Magazine*, *38*.

Lenat, D. B. (1995). Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, *38*, 33–38.

Li, J., Jones, S., Mohan, S., & Derbinsky, N. (2016). Architectural mechanisms for mitigating uncertainty during long-term declarative knowledge access. *Proceedings of the 4th Annual Conference on Advances in Cognitive Systems (ACS)*.

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, *38*, 39–41.

Mohan, S., Mininger, A., Kirk, J., & Laird, J. E. (2012). Acquiring grounded representations of words with situated interactive instruction. *Advances in Cognitive Systems*, *2*.

Munroe, R. (2010). Color survey results. `https://blog.xkcd.com/2010/05/03/color-survey-results/`, May 2010.

Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Salvucci, D. D. (2015). Endowing a cognitive architecture with world knowledge. *Proceedings of the 37th Annual Conference of the Cognitive Science Society (CogSci)*.

Sierra, A., & Corbacho, F. J. (2000). Reclassification as supervised clustering. *Neural Computation*, *12*, 2537–2546.

Wu, L., Hua, X.-S., Yu, N., Ma, W.-Y., & Li, S. (2008). Flickr distance. *Proceedings of the 16th ACM International Conference on Multimedia* (pp. 31–40). New York, NY, USA: ACM.