Joint Effort: A Strategy for Human Computation

Thomas B. Jones[a], Chayan Chakrabarti[b,*], George F. Luger[a], Matthew D. Turner[c], Jessica A. Turner[c]

[a]*Department of Computer Science, University of New Mexico, Albuquerque, NM*
[b]*General Electric Company, San Ramon, CA*
[c]*Department of Psychology, Georgia State University, Atlanta, GA*

Abstract

As machine learning processes have a greater impact on our lives, humans and machines must often work together on data classification tasks. A human-machine interaction system that is able to determine which items are most difficult to automatically annotate can then pass those items off to a human user. These *joint effort* strategies can increase performance while decreasing labor costs. In this paper, we first present a meta-learning strategy for joint effort. Next, we explore how a new measure derived from the F-score relates to annotation costs and benefits when humans are used to perform part of an annotation task. We then simulate the behavior of this score when used to measure joint effort system performance with differing competencies at both the base learner and meta-learner levels. Finally, we apply the strategy to a use case — annotating neuroimaging research literature — and explore its lessons for human-machine interaction systems. Our work shows that the strategy employed by the meta-learner that maximizes total profits is dependent on the *cost geometry* of the task — the costs and values of correct and incorrect label results — as well as the accuracy of the meta-learner in choosing between easy and hard generalization tasks.

Keywords: multi-label classification, meta-learning, cost geometry, joint effort

*Corresponding Author
Email addresses: ThomasBJones2@gmail.com (Thomas B. Jones), cchakrab@gmail.com (Chayan Chakrabarti), luger@cs.unm.edu (George F. Luger), mturner46@gsu.edu (Matthew D. Turner), jturner63@gsu.edu (Jessica A. Turner)

1. Introduction

Meta learning — learning about learning — has been an established field of inquiry in machine learning for more than two decades. Much of this work has focused on building *meta-knowledge* about a set of automatic machine learning algorithms and training data sets so as to choose the best available or constructable learning algorithm for the generalization set (Vilalta and Drissi, 2002; Smith-Miles, 2008; Stolfo et al., 1997; Michie et al., 1994).

This work presumes that every item in a generalization set — the set of items that the learner must label, classify, identify, hypothesize about, etc. — must be generalized by a set of automatically generated machine learners. This *fully automated machine learning* is a lofty goal, but it leaves us open to a lack of understanding about possible trade offs between human and machine learners.

Additionally, a large amount of work dealing with learning systems and cost focuses on the single-label case where a classifier is trying to choose one label among many for each item in the generalization set (Lomax and Vadera, 2013). Here we examine how the properties of the multi-label case can modify how costs are accounted for compared to the single-label case. Specifically, we note that there is an asymmetry between false positives and false negatives when considering many multi-label classification problems. Metrics derived from the F score can be used to balance tasks between human laborers and machines when positive label results are more important than negative label results.

We present a strategy for dividing tasks between human and machine learners. This strategy takes into account (1) the benefits accrued when tasks are successfully or unsuccessfully completed (2) the costs of completing the tasks using humans and (3) the ability of a machine to pass some tasks off to a human. Our method is one of many possible *joint effort* strategies — learning strategies that automatically balance generalization tasks between human laborers and a machine learner so that both perform some *significant and appropriate* proportion of the work.

The normative goal of many meta-learning systems lies in finding the best learning algorithm(s) for a particular learning task. This is achieved by meta-learning the performance of a set of base machine learning algorithms and then combining the outputs of those algorithms according a model produced by the meta-learner. Overviews of this approach to meta-learning are provided by Vilalta and Drissi (2002), Ruta and Gabrys (2000), and Kittler et al. (1998).

A great deal of research has gone into building automated meta-learning systems. For example, Kadlec and Gabrys (2008) present a gated neural network based meta-learner that weights local expert machine learners based on their performance. They show that the resulting combination learner outperforms a different learning algorithm combination technique on mean squared error. More recently, Hmida and Slimani (2010) present a distributed system for meta-learning built on the WEKA machine learning system (Witten and Frank, 2005). In their system a fully automated set of learning nodes attempt to learn models of a data set locally and then the best learners are learned by a set of meta-learning nodes.

Finally, turning to a meta-learning system for the multi-label case, Chekina et al. (2011) use data set meta-features — features such as data set length, mean, standard deviation, number of attributes, etc. — to aid the meta-learner in determining which underlying machine learner would be best on a number of multi-label classification tasks. They discovered that their meta-learning algorithm could predict the best performing algorithm two-thirds of the time.

In contrast, our work focuses on splitting a multi-label data set based on the difficulty of labeling items in that data set. We measure different ways of splitting such a set with a *joint effort classifier* — a meta-learner set up to split a generalization set into easy and hard categories for consumption by human and machine learners. Then we examine how our joint effort learner affects the costs of employing human labor to complete learning tasks.

In this paper we outline a strategy for joint effort. This strategy splits a generalization task into two groups — items that are *easy* to annotate and those that are *hard* to annotate, using a joint effort classifier.

We then present a method for determining when joint effort may be worth the effort using a metric derived from a common measure for machine learning performance, namely the F_β-score.

Next, we use a simulated joint effort classifier, a simulated machine learner, and simulated data to characterize the behavior of R on both the *easy* and *hard* items given differing learner and joint effort classifier competencies. Finally, we present the results of our cost-gap measurement on a gold standard data set built using CogPO (Turner and Laird, 2012).

Section 2 presents the joint effort strategy. Next, Section 3 describes the value cost ratio and provides its motivation, and Section 4 presents simulated data used to show how the metric varies with different learner and joint effort classifier accuracies. Section 5 shows the results of applying this metric to the BrainMap corpus using a multi-label naive Bayes learning algorithm and a naive Bayes joint effort classifier. Finally, Section 6 offers conclusions and discusses future work.

2. Joint Effort

Some foreseeable joint effort strategies include completing part of a generalization task with a machine and part of the same task with a human, using a machine learner to advise a human learner who makes a final call, or splitting tasks so that some tasks go to humans and some to machines. These strategies are driven by the idea that humans can interact with machine learning systems to reduce human labor and produce better results than might be produced by either the humans or the machine alone.

We are interested in joint effort since this strategy is generalizable to many domains. Consider a chatter-bot that accepts questions from a human user, classifies the questions using some machine learning algorithm, and then uses the classification to select from a number of pre-programmed responses (Chakrabarti, 2014). When the chatter-bot successfully identifies a human user's query, there is some value produced. Alternatively, when it is unsuccessful, a lesser value is achieved. In either case, the chatterbot alone has some performance on the task of classifying the needs of its human user.

When the chatter-bot is performing a task such as interfacing with an e-commerce site, the difference in value between a correct identification and an incorrect one can be large. A customer may choose a competitor's site or product if sufficiently frustrated by a bad chatter-bot experience. One alternative is providing humans instead of chatter-bots to aid human users. However, providing a human in every interaction is difficult to justify if a chatter-bot is cheaper than human labor and *mostly good-enough* — helping most users get close to achieving their goals. Using joint effort to seamlessly pull human experts into the loop when users become frustrated provides a higher total performance than a machine alone and lower costs than humans alone.

2.1. Human Computation

Traditionally, interaction between humans and machine learners has been limited to humans providing data to constrain a machine learning system and then stepping back while the machine learner does the rest of the work (Settles, 2010; Snow et al., 2008; Laws et al., 2011; Rashtchian et al., 2010). This human-as-machine-teacher model — where the human acts as an oracle to the machine — has been very useful, but it doesn't cover all the possible ways that machines might work with humans.

Human computation, on the other hand, uses humans to perform tasks that computers are incapable of performing (Von Ahn, 2009). In its original formulation, human computation involves a process of gamification, where humans are automatically presented with generalization tasks as a game. However, many of the strategies employed for joint effort involve placing the emphasis on allowing the humans to perform *all* of a generalization task. This is true even though the process of presenting those tasks to a human user may be highly automated.

Another strategy for human computation involves machines automatically delegating tasks between human learners. This gets much closer to a true joint effort task. For example, Nushi et al. (2014) build a system for automatically directing Q&A queries to those humans most likely to have the necessary expertise. Parameswaran et al. (2014) use machines to balance filtering loads between human laborers. They do not note it in their paper, but there are strong parallels between what they do, balancing filtering tasks between human laborers, and what a joint effort classifier does in balancing generalization tasks between machine learners.

2.2. A Strategy for Joint Effort Learning

An outline of our strategy for directing generalization tasks to machines or humans can be seen in Figure 1. In this figure we show how tasks can be balanced between humans and machine learners. To kick start the process, we assume that the human annotators will provide a gold standard data set for supervised learning. This data set will be used to both provide the underlying machine learner with bindings between feature evidence and output labels, as well as a method for scoring the learner's ability after learning. This strategy assumes that the data sets for learning and generalization are stationary — that future data will be drawn from a data set that is statistically similar to the gold standard data set.

3. Value-Cost Analysis

The cost of human labor is often very large while the costs of machine labor are often quite low. Properly balancing a joint effort task between human and machine learners, therefore, means having an appreciation for the costs incurred in the use of human labor. Additionally, an appreciation for the value produced by a particular task is also required. In previous work, these two different concepts - the cost of labeling an item and the value produced by the label, have often been considered jointly (Elkan, 2001). We separate them in an effort to show how common performance measures relate to labeling costs and values.

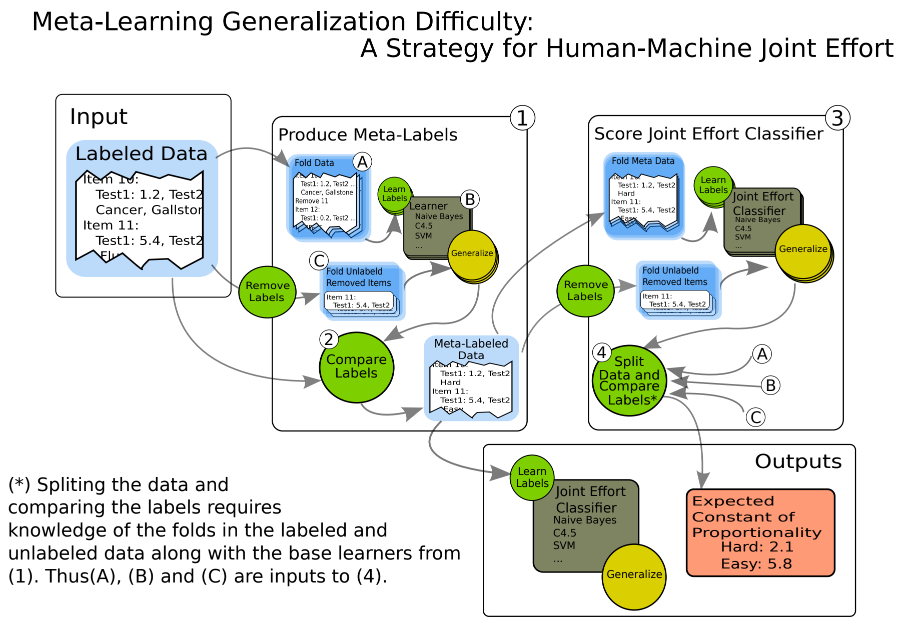


Figure 1: In this figure we present our strategy for human-machine joint effort. The strategy relies on (1) folding the labeled learning data to assign either an *easy* or a *hard* label to each item in the generalization task. (2) Each item's label is determined by the performance of the learner on that specific item with items that have mostly correct labels sent to the *easy* category, and those with mostly incorrect labels sent to the *hard* category. Then (3) the labeled data is also folded and (4) the constant of proportionality R for each fold is calculated and averaged. Finally a joint effort classifier produced with all of the meta-labeled data from (1) is returned along with the scores from (3). This strategy assumes that future data will be drawn from a data set that is statistically similar to the labeled data set.

3.1. Cost Dependent Learning Algorithms

Human-machine interaction systems often take the cost of *obtaining* information into account. This is especially true of decision tree algorithms that are able to ignore many available features on any given run. (Davis et al., 2006) design an implementation of the ID3 algorithm that places the most expensive features higher in the tree, while placing less expensive tests lower in the tree. A compelling argument for this paradigm of placing the most expensive tests first is the existence of problem sets, like forensics, where obtaining certain expensive features for every item earlier in the testing process is actually less expensive than obtaining these features for only a few items later on in the process.

While feature selection is an important part of determining classification costs, many authors also focus on misclassification costs — the cost of obtaining an erroneous output. Both Ling et al. (2006a) and Domingos (1999) focus on the cost of misclassification in decision trees, though the method presented by Domingos (1999) can be applied to many different learning algorithms.

A number of authors capture both misclassification and feature costs in a single algorithm (Núñez, 1991; Tan, 1993; Turney, 1995; Ling et al., 2006b). This matching of feature costs and misclassification costs can occur in a number of ways. Zhang (2010) presents a decision tree algorithm, *cost-time sensitive classification*, that uses different units for feature costs and misclassification costs. Sheng and Ling (2006) and Greiner et al. (2002) also use both feature and misclassification costs but with a focus on *active learning* — determining when to obtain missing feature values.

A recent survey of cost sensitive decision tree algorithms shows many decision tree algorithms that take costs into account (Lomax and Vadera, 2013). Similar to the previously cited papers (Núñez, 1991; Tan, 1993; Turney, 1995; Ling et al., 2006b), these algorithms redirect searchers so as to avoid the collection of high cost features and choose the least risky labels when the data can not overwhelmingly support more risky choices.

We focus partially on the costs of misclassification when using a Bayesian classifier. Elkan (2001) provides a theorem that re-weights data in the training set of a single learning algorithm to take costs into account. This procedure is applied to both decision trees and Bayesian classifiers.

Ibánez et al. (2011) use a cost sensitive naive Bayes classifier to predict h-index scores using a variety of features for different authors. Cost re-weighting is obtained by multiplying each label's output probability by its cost in a cost matrix.

Finally, Chai et al. (2004) present a method for evaluating feature costs in a Bayesian classifier. This work is unique in that it allows for *batch* feature collection — the collection of many features all at once. Decision tree algorithms often focus on *sequential* feature collection — the collection of a feature at the moment it is needed.

3.2. The Value-Cost Ratio

Previous work has looked at minimizing the cost of misclassification and data collection. This work is aimed at a slightly different problem. We will instead be examining the ratio between the average *value of a label* — the amount of resources that can be obtained after a specific label has been found — and the average *cost of labels* — the amount of resources that must be spent on human labor to obtain labels. Assuming that both costs and values are stated in the same units, when this ratio is below one then it is better to use an automated classifier. Otherwise, when this ratio is above one, human labor is justified.

There are many potential data structures available to represent the cost and value contexts of any given machine learning task. The simplest, and most complete, data structure for this task is the matrix. As pointed out by Elkan (2001), it is best to use a value or benefit matrix, instead of a cost matrix, when considering the values or costs of labels. Here, we instead reserve the cost matrix for the cost of using humans to perform classification tasks.

When the generalization task is a single-label task, the formulation of both cost and value matrices is straightforward. An expected value — in terms of dollars, time, resources saved, etc. — is assigned to each of the categories *true positive*, *false positive*, *true negative* and *false negative*. This gives a 2-by-2 matrix where the rows represent the 'truth' of a prediction while the columns represent the actual value of the prediction — either positive when the machine learner predicts a label should apply or negative when the machine learner predicts that it shouldn't apply. When dealing with multi-class or multi-label classification tasks, however, there are many possible non-equivalent cost matrices.

3.3. Value and Cost Matrices

Among the labeling tasks, there is a hierarchy constructed of single-label classification tasks, single-label/multi-class classification tasks, and multi-label classification tasks (Turner et al., 2013). The set of possible, meaningfully different single-label classification cost and value matrices is limited to 2-by-2 matrices as discussed in the previous section. On the other hand, more complicated classification tasks can have a larger set of potential meaningfully different cost matrices.

For example a single-label/multi-class value matrix with N labels must have up to N^2 cells, one for each combination of actual label and predicted label, to capture information about every combination of actual and predicted labels. An example value matrix for such a classification task to determine the proper diagnosis for a patient is given in table 1.

In this matrix, the available classes are *flu*, *cancer*, *gallstone*, or *hepatitis*. Now, in a single-label/multi-class classification task, the learner must return only a single classification. This means that a full description of the task will describe the value dependent on the *predicted* label and the *actual* label. The matrix in table 1 has 4 rows and 4 columns, one for each of the actual label-predicted label pairs. The actual values are recorded in the matrix element $V_{p,a}$ where p is the predicted label and a is the actual label.

When accounting for a multi-label classification task, however, there are many more degrees of freedom than in the single-label/multi-class classification task. Since more than one label may apply to a given generalization task, a cost matrix like table 1 is not sufficient to capture all the information available for label costs and values.

		Actual Label			
		Flu	Cancer	Gallstone	Hepatitis
Predicted Label	Flu	$V_{f,f}$	$V_{f,c}$	$V_{f,g}$	$V_{f,h}$
	Cancer	$V_{c,f}$	$V_{c,c}$	$V_{c,g}$	$V_{c,h}$
	Gallstone	$V_{g,f}$	$V_{g,c}$	$V_{g,g}$	$V_{g,h}$
	Hepatitis	$V_{h,f}$	$V_{h,c}$	$V_{h,g}$	$V_{h,h}$

Table 1: **Single-Label/Multi-Class Value Matrix** — In this value matrix we have four labels/diagnoses available: Flu, Cancer, Gallstones, and Hepatitis. Each of the cells contains the value of the predicted label *given* the actual label — so $V_{f,h}$ can be read the 'the value of a flu diagnosis given that a patient actually has hepatitis.'

One path forward is to build a matrix that contains a row for every combination of predicted labels and a column for every combination of actual labels. A matrix like this, for the same learning task as in table 1 is found in table 2. Since this matrix must account for every pair of combinations of predicted and actual labels it contains $(2^4)^2$ cells. Any such matrix for a learning task of N labels will contain $min((2^N)^2, |d|)$ cells.

This method is similar to the *label powerset* strategy for transforming multi-label classification problems into single-label classification problems (Tsoumakas and Vlahavas, 2007; Read et al., 2011). While this growth rate may be avoided by more closely examining the domain of the generalization task, limiting the output of the learner to only 'sane' label combinations, this may not always possible. As a result, we come upon a case where there is a trade-off between a full description of the cost of the problem and the amount of computational complexity required to implement that description.

One alternative to the matrix presented in table 2 follows the *binary relevance* strategy for transforming multi-label classification problems into single-label classification problems (Read et al., 2011). This strategy trains N single-label classifiers for a multi-label task with N labels. Each classifier is blind to the labels of the other classifiers and only works on determining if an item should be labeled with the label it has been assigned. In a similar way, the

		Actual Label			
		Flu	..	Flu, Cancer, Gallstone	All Labels
Predicted Label	Flu	$V_{f,f}$	..	$V_{f,(f,c,g)}$	$V_{f,A}$
	:	:	$\ddots$	:	:
	Flu, Cancer, Gallstone	$V_{(f,c,g),f}$	..	$V_{(f,c,g),(f,c,g)}$	$V_{(f,c,g),A}$
	All Labels	$V_{A,f}$	..	$V_{A,(f,c,g)}$	$V_{A,A}$

Table 2: **Multi-Label/Multi-Class *Combination* Value Matrix** — Here we show part of a combination value matrix for a multi-label problem with four labels: Flu, Cancer, Gallstone, and Hepatitis. In this matrix we use a very similar representation for value as those in table 1. We represent a combination of labels with a tuple and we use 'A' to stand in for all labels. In this matrix $V_{(f,g,c),f}$ can be read as the value of labeling an item with 'Flu', 'Cancer', and 'Gallstone' when the actual label is just 'Flu'. We present only part of this combination matrix since even a labeling task with only 4 labels requires 256 cells for a complete description of the values of its possible combinations.

matrix set in table 3 accounts for label costs separately, combining the 2-by-2 strategy of accounting for costs of the single-label classifier with an admission that all possible label combinations must be somehow accounted for. Each label receives its own cost matrix for when the machine learning algorithm produces a *true positive*, *false positive true negative* and *false negative* on that label.

The value scheme in table 3 is built for the case where any particular combination of labels may be likely and where the value of a mislabeling on a particular label is independent or mostly independent from the value of a mislabeling on a different label. This cost scheme presented in table 3 has the benefit of only requiring $4N$ cells compared to the $(2^N)^2$ cells for a matrix based on label combinations.

Determining the ratio of labor costs to the value of correctly labeling items also means determining the cost of human labor. Many different decision criteria can be used to account for the cost of human labor in labeling tasks. These include, but are not limited to: looking at the cost per generalization task, the cost per byte of data examined, and the cost per label. We focus on the cost of

Flu		
	Positive	Negative
True	V^f_{TP}	V^f_{TN}
False	V^f_{FP}	V^f_{FN}

Cancer		
	Positive	Negative
True	V^c_{TP}	V^c_{TN}
False	V^c_{FP}	V^c_{FN}

Gallstone		
	Positive	Negative
True	V^g_{TP}	V^g_{TN}
False	V^g_{FP}	V^g_{FN}

Hepatitis		
	Positive	Negative
True	V^h_{TP}	V^h_{TN}
False	V^h_{FP}	V^h_{FN}

Table 3: **Multi-Label/Multi-Class *Per Label* Value Matrix Set** — In this value matrix set we have four labels/diagnoses available: Flu, Cancer, Gallstones, and Hepatitis. Each of the matrices tracks the predicted and actual labeling values for a single-label. The cells contain the values for the four categories — true positive (TP), true negative (TN), false positive (FP), and false negative (FN) — on each of the labels separately. Therefore, V^g_{TP} is the value of a true positive on a gallstone. The final value of a classification task involves summing over the observed cell in each of the matrices in the matrix set.

correcting faults in a machine learning task on a per label basis. This is useful since it allows us to build a cost model that accounts for costs when human laborers are given the outcome from the machine learning process as a starting point for their own evaluation. In this model we are focused on the cost, to human laborers, of 'correcting' the machine learner.

Once a decision criteria is chosen to account for human labor costs, we must also assign values to the different possible outcomes that human laborers will have to deal with. For the sake of symmetry, we chose to handle human labor costs in the same way as values. In this paper, matrices like those in tables 1, 2, and 3 are assumed for both label values and labor costs. So, for example, if we are talking about the problem in table 3, a parallel set of cost matrices are assumed where C^g_{TP} is assumed to be the cost of 'correcting' a true positive gallstone, g, label.

We will assume a binary-relevance cost and value matrix set like that found in table 3. We represent the set of all possible labels with L. Next, we represent: (1) the set of true positives on any label, $\ell \in L$ as TP_ℓ, (2) the set of false positives on that label as FP_ℓ, (3) the set of true negatives on that label as TN_ℓ, and (4) the set of false negatives on that label as FN_ℓ. We will say that V is the set of matrices containing label values and C is the set of matrices containing human labor costs. For any given label ℓ, V^ℓ is the value matrix of that label and C^ℓ is the human labor cost matrix of ℓ. Additionally, V^ℓ_{TP}, V^ℓ_{FP}, V^ℓ_{TN} and V^ℓ_{FN} are the value of a true positive, false positive, true negative, and false negative on label ℓ respectively. Finally, we will represent the set of matrices for human labor costs with similar notation.

It is important to consider the profit of any generalization task. Given that a value and cost matrix are already provided, then the total profit of a generalization task is

$$\begin{aligned}\mathrm{G} = &\sum_{\ell \in L}(V^\ell_{TP}|TP_\ell| + V^\ell_{FP}|FP_\ell| + V^\ell_{TN}|TN_\ell| + V^\ell_{FN}|FN_\ell|) - \\ &\sum_{\ell \in L}(C^\ell_{TP}|TP_\ell| + C^\ell_{FP}|FP_\ell| + C^\ell_{TN}|TN_\ell| + C^\ell_{FN}|FN_\ell|)\end{aligned} \tag{1}$$

In this equation the *generalization task profit*, G, is equal to the sum of the values of the labels *minus* the costs of obtaining those labels. For any set S, $|S|$ is the size of that set. Therefore $|TP_\ell|$ is equal to the number of items in the generalization task that are labeled with ℓ and that *should* be labeled with ℓ — in other words that have a true positive labeling of ℓ.

We assume that (1) the labor costs of the automatic learner is 0 and that (2) the cost of human labor is positive [1]. Using these assumptions the total profit of the labels produced by an *automatic machine learning system*, A, is

[1]We chose these assumptions because we believe that the costs of human labor are often much greater than those of a machine learner. However, if this isn't the case, the framework can be extended by treating the costs in equation 3 as a difference of costs and always placing the method of labeling with lowest costs in the place of the machine learner in the framework. In this way, the framework becomes a comparison between a relatively inexpensive method of labeling and some more expensive method.

$$\mathrm{A} = \sum_{\ell \in L} (V^{\ell}_{TP}|TP_{\ell}| + V^{\ell}_{FP}|FP_{\ell}| + V^{\ell}_{TN}|TN_{\ell}| + V^{\ell}_{FN}|FN_{\ell}|) \tag{2}$$

Finally, we assume that expert human laborers will always return the correct labels. We also assume that we can capture the difference in costs between machine laborers and human laborers in a cost matrix whose every entry is positive. Therefore we can determine that the profit of *expert human labor*, H, is

$$\begin{aligned} \mathrm{H} = &\sum_{\ell \in L} (V^{\ell}_{TP}(|TP_{\ell}| + |FN_{\ell}|) + V^{\ell}_{TN}(|TN_{\ell}| + |FP_{\ell}|)) - \\ &\sum_{\ell \in L} (C^{\ell}_{TP}|TP_{\ell}| + C^{\ell}_{FP}|FP_{\ell}| + C^{\ell}_{TN}|TN_{\ell}| + C^{\ell}_{FN}|FN_{\ell}|) \end{aligned} \tag{3}$$

In equation 3, all the false positives and false negatives found by the machine learner are joined with the true positives and true negatives respectively. This accounts for our assumption that the humans will produce correct output or that, at the very least, the output they produce is the best possible output. We subtract the costs of employing the humans to correct the machine learner in the second half of the equation.

Using these values, determining when it is better to use human annotators or automatic annotators consists simply of determining if $A \leq H$. After some algebra on A and H we can see that justifying human labor means that inequality 4 must hold

$$1 \leq \Delta V_{avg}/\Delta C_{avg} = \frac{\sum_{\ell \in L} (V^{\ell}_{TP} - V^{\ell}_{FN})|FN_{\ell}| + (V^{\ell}_{TN} - V^{\ell}_{FP})|FP_{\ell}|}{\sum_{\ell \in L} (C^{\ell}_{TP}|TP_{\ell}| + C^{\ell}_{FP}|FP_{\ell}| + C^{\ell}_{TN}|TN_{\ell}| + C^{\ell}_{FN}|FN_{\ell}|)} \tag{4}$$

This ratio, $\Delta V_{avg}/\Delta C_{avg}$ is the ratio of the value of the task to the cost of the human labor necessary to complete the task. To justify scoring by a human scorer this value must be greater than 1.

3.4. The Confusion Score

The previous discussion raises an interesting point: there is an inherent asymmetry in many multi-label classification tasks. It lies in the fact that for many of these tasks, the number of negative, non present, labels greatly outnumbers the number of positive, present, labels. That is, most labels don't apply to most items most of the time. This means that the default assumption on any label is that it doesn't apply to any given item in the generalization task.

The number of labels on each item in a generalization task is called the item's *cardinality*. In many, if not most, multi-label tasks the average cardinality of items slated for generalization is much lower than the number of available labels. When this is the case, measures like the micro-F-score, which is the harmonic mean of precision and recall over all labels and task items, are much better than accuracy at measuring machine learning performance (Pillai et al., 2014). The micro F-score often makes use of a tuning value, β that determines the relative importance of precision and recall in the final F-score. A formula for the micro F_β is:

$$\mathrm{F}_\beta = \frac{(1+\beta^2)TP}{(1+\beta^2)TP + \beta^2 FN + FP} \tag{5}$$

Bringing this back to our discussion of costs, one reason the micro-F-score is much better than accuracy for multi-label classification problem lies in the way that it treats true negatives. They are effectively ignored by the F-score. This correlates well with an understanding of true negatives as being essentially valueless and costless due to a true negative being a correct example of the null hypotheses. Looking again at inequality 4, notice that if we set all $V_{TP} - V_{FP} = R(\beta^2)$, all $V_{FN} = R$, all $V_{TN} = 0$, all $C_{TP} = (1+\beta^2)$, all $C_{FP} = \beta^2$, all $C_{FN} = 1$ and all $C_{TN} = 0$ then we obtain:

$$\Delta V_{avg}/\Delta C_{avg} = R(1 - F_\beta) \tag{6}$$

This means that under the proper cost constraints, $1 - F_\beta$, the *confusion* of the machine learner, is proportional to the ratio $\Delta V_{avg}/\Delta C_{avg}$. In this paper we define the confusion score such that $1 - F_\beta = C_\beta$.

Additionally, we see that there is a constant of proportionality R in equation 6. Given that the confusion score is a property of a machine learner and its data set and that $\Delta V_{avg}/\Delta C_{avg}$ must be greater than 1 to justify human labor, we can determine the constant of proportionality that will ensure human labor is profitable on a given labeling task using just the confusion score:

$$\mathrm{R} \geq \frac{1}{C_\beta} \tag{7}$$

In the rest of this paper, we call $R_\beta = \frac{1}{C_\beta}$ a machine learner's R_β-Score and we concern ourselves only with the R_1-score, a score that equally weights learner precision and recall.

4. Simulation Results

We simulated the operation of a human-machine joint effort generalization like that outlined in Figure 1 on 100 generalization tasks, each containing 1000 items labeled with between one and five labels out of a total of twenty possible labels.

The simulation used a Monte-Carlo process to simulate both a learner and a joint effort classifier. The learner assigned between one and five labels to each item in the generalization task, using the Monte-Carlo probability to determine if the labels were correctly predicted or if they were alternatively incorrectly predicted (either false positive or false negative). We used the same process for the joint effort classifier, except that the joint effort classifier only needed to sort items into *easy* or *hard*. An item was determined to be correctly labeled *easy* if the C_1-score on just its labels was less than 0.5 and *hard* otherwise.

After simulating the labeling process and the splitting process, we then measured the R_1-score on the items labeled *easy* (Figure 4 A), the items labeled *hard* (Figure 4 B), and the difference in R_1-scores between *easy* and *hard* items (Figure 4 C).

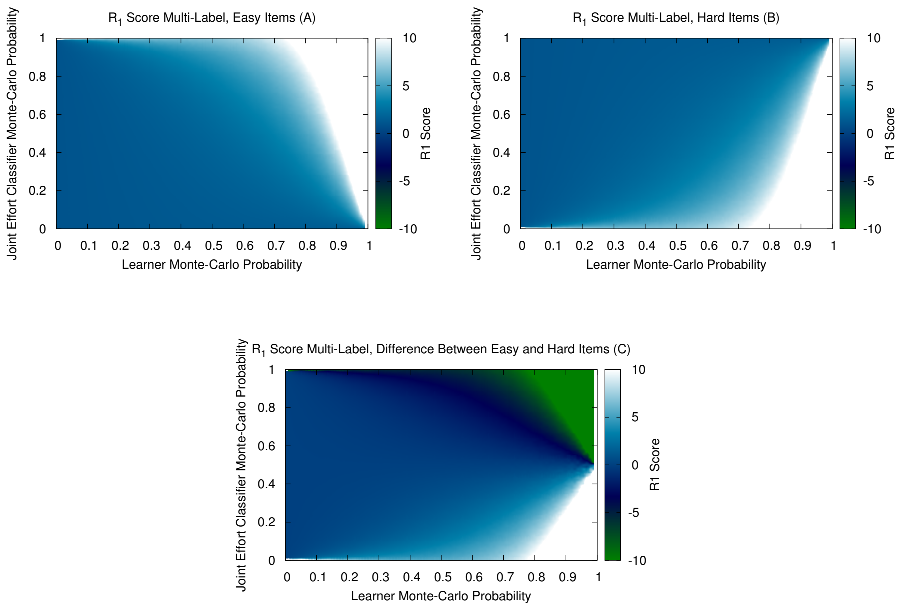


Figure 2: The heatmaps for the R_1-score on a set of (A) *easy* and (B) *hard* items as determined by a simulated joint effort classifier in terms of the probability of successful labeling in both the machine learner and the joint effort classifier. The third graph (C) displays the difference between the first and second graphs. Since the R_1-score varies greatly at the edges of the graph all values are clipped between -10 (green) and 10 (white), in order to show the behavior of the R_1-score over the center of the graph. Notice that there is a large difference between the R_1-score of the (A) *hard* and (B) *easy* data sets when both the joint effort classifier and learner are very accurate. It is in this operating region that a joint effort strategy is most likely to be viable.

As can be seen in Figure 4 A, the value of R_1 of the *easy* data set *increases* as both the joint effort classifier's Monte-Carlo success rate *increases* and as the machine learner's Monte-Carlo success rate *increases*. This occurs because the *easy* data set is benefiting from both the high base rate of correctly labeling items by the learner, and the high likelihood that all poorly labeled items will be removed from the *easy* data set by the joint effort classifier. Thus, a competent machine learner and joint effort classifier means that the confusion must be low and therefore R_1 must be high.

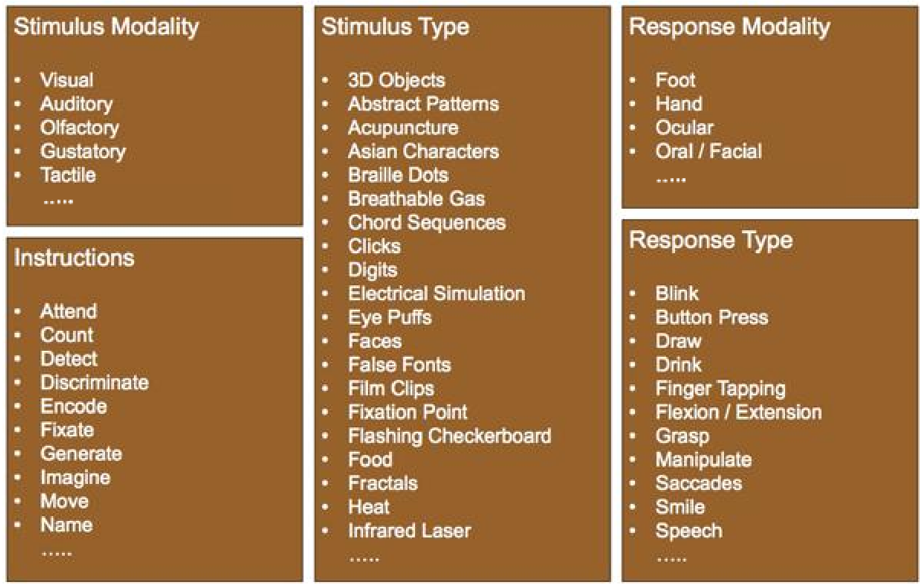
Looking at graph B we can see that the value of R_1 *increases* for the *hard* items as the joint effort classifier's success rate *decreases* while the learner's Monte-Carlo labeling probability *increases*. This is due to the fact that the joint-effort classifier must work harder to extract poorly labeled items from the items in the generalization task as the underlying learning process becomes more competent, but if the joint effort classifier is sufficiently capable, the *hard* data set will often still have a large number of poorly labeled items leading to a large confusion score. With a large confusion score comes a smaller R_1 value necessary to justify labor by human experts.

Finally, graph C in Figure 4 shows that there is a large difference between the R_1-scores of the *hard* and *easy* sets when both the Monte-Carlo labeling rate of the meta learner and the learner are high. When this is the case, both the *easy* and *hard* data sets contain a large number of correctly labeled items, however the performance of the learner on the *easy* data set is very close to perfect, while it is less perfect for the *hard* data set. Even this small difference in performance is enough to create a large difference in R_1-scores since the R_1-score is the *reciprocal* of the confusion score.

5. The Confusion Score on the BrainMap Corpus

We present the results of our experiments using that score on abstracts annotated with labels drawn from the CogPO ontology. The Cognitive Paradigm Ontology is a framework for the explicit representation of the cognitive paradigm, that is, the behavioral elements of neuroimaging experiments involving humans (Turner and Laird, 2012). It provides a controlled vocabulary for the annotation of the published neuroimaging literature with labels uniformly identifying aspects of experimental design. For instance, it links named behavioral paradigms (specific) with their functional descriptions (general).

The five categories we used in this study were: *Stimulus Type* with 46 labels and an average cardinality of 1.26, *Response Type* with 18 labels and an average cardinality of 1.56, *Stimulus Modality* with 6 labels and an average cardinality of 1.20, *Response Modality* with 10 labels and an average cardinality of 1.25, and *Instructions* with 20 labels and an average cardinality of 1.43. A rough overview can be seen in Figure 5 and more details are given by Turner and Laird (2012), Chakrabarti et al. (2014), Chakrabarti et al. (2013), and at `www.cogpo.org`.



Stimulus Modality

- Visual
- Auditory
- Olfactory
- Gustatory
- Tactile

.....

Instructions

- Attend
- Count
- Detect
- Discriminate
- Encode
- Fixate
- Generate
- Imagine
- Move
- Name

.....

Stimulus Type

- 3D Objects
- Abstract Patterns
- Acupuncture
- Asian Characters
- Braille Dots
- Breathable Gas
- Chord Sequences
- Clicks
- Digits
- Electrical Simulation
- Eye Puffs
- Faces
- False Fonts
- Film Clips
- Fixation Point
- Flashing Checkerboard
- Food
- Fractals
- Heat
- Infrared Laser

.....

Response Modality

- Foot
- Hand
- Ocular
- Oral / Facial

.....

Response Type

- Blink
- Button Press
- Draw
- Drink
- Finger Tapping
- Flexion / Extension
- Grasp
- Manipulate
- Saccades
- Smile
- Speech

.....

Figure 3: A rough overview of five categories in CogPO. There is a large diversity in the number of available labels over each of these categories.

The gold standard data set used here was obtained by expert human annotation of the full-text of the relevant scientific papers, primarily by one of the authors (ARL). For our analysis, these labels were paired with the text of the abstracts for the papers. It is worth noting this feature: the machine learning system was not given the same complete information used by the expert human annotators.

We used a java machine learning package called MALLET McCallum (2002) that provides the user with many classification objects among which is a single-label naive Bayes classifier transformed into a multi-label naive Bayes classifier using *binary relevance*. This method creates separate naive Bayes classifiers for each potential label in our five categories. Thus, the category Stimulus Modality would have six different binary naive Bayes classifiers for each of its six labels.

We used these classifiers in a two step classification process. In step (1) the 2251 abstracts were split into 100 folds and each abstract was classified using only abstracts from the other 99 folds with the classification results recorded in each category. Next, if the C_1-score of the abstract in a particular category was less than the *joint effort classifier discrimination rate* then the abstract was classified as *easy* and otherwise it was classified as *hard*. In step (2), a naive Bayes joint effort classifier was built for each of the 100 folds and each label category, with each joint effort classifier learning which items should be labeled *easy* or *hard*. Then, the the R_1-scores of the generalization task performed in this step were recorded separately for both the *easy* data set and the *hard* data set as determined by the joint effort classifier. This follows the joint effort strategy outlined in Figure 1. Throughout this process, we ensured that the folding was done properly across both the learner and the joint effort classifier, with the joint effort classifier being given access only to the labeling data from the 99 folds it was attempting to generalize.

This process was performed on all 2251 abstracts using all 100 folds. We performed this process with joint effort classifier discrimination rates — the confusion score at which items were moved from the *easy* category to the *hard* category — that varied from 0 to 1 in increments of 0.1. Finally, for each discrimination rate, we measured the R_1-score for each of the categories for both *hard* and *easy* data sets as well as a baseline rate for both sets together over all 100 folds. In Figure 5 we present the average R_1-score along with its standard deviation for each of the five categories at various joint effort classifier discrimination rates.

In Figure 5 we show that the items identified as *hard* have an R_1 Score that

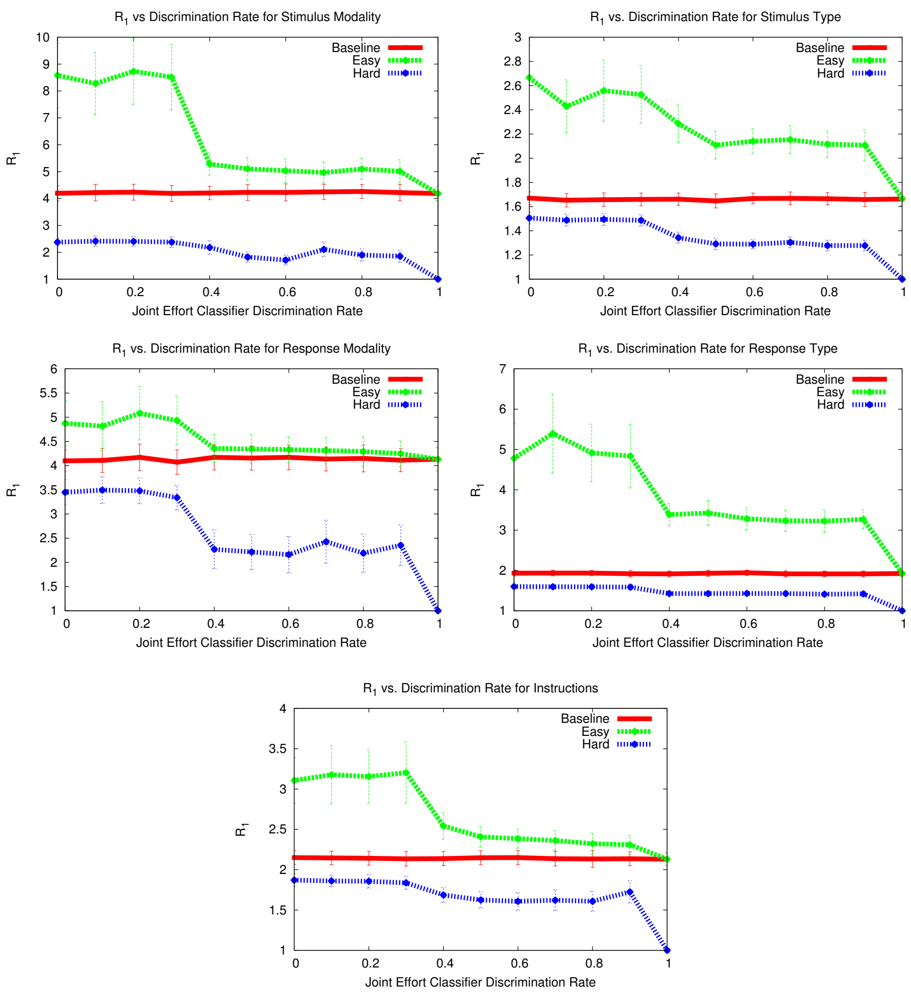


Figure 4: We show the R_1 rate for items labeled *easy* and *hard* by a naive Bayes joint effort classifier as well as a baseline that shows the confusion for both groups together on all five categories from CogPO. The independent variable is the joint effort classifier discrimination rate. As the discrimination rate is turned down, the constant of proportionality for both the *easy* and *hard* categories rise. The Y-axis is re-scaled for each panel. See text for explanation.

sits below the baseline score while those identified as *easy* have a constant of proportionality that sits above the baseline score (Remember, a *low* constant of proportionality means a *high* confusion score and vice versa). The naive Bayes joint effort classifier is able to discriminate between *easy* and *hard* items even though it relies on the same basic data as the underlying learners and has the same inductive biases.

There are two hypothesis for why this is the case. First, the joint effort classifier is discovering *ambiguous* terms in the BrainMap abstracts that indicate that an item may be *hard*. An example of such a term is "oddball" which can indicate either a *visual* or *auditory* Stimulus Modality. The use of the same term for items that differ in stimulus modality may make the classification task harder.

Our second hypothesis relies on the fact that for each category, most items have one of a few different labels. In the Stimulus Modality category, for instance, items are labeled "Visual" 73.5% of the time. The joint effort classifier may be acting as a 'second vote' in favor of items in the majority label, corralling the items that truly have that label into *easy* set and sending items that come from the more difficult minority labels to the *hard* set.

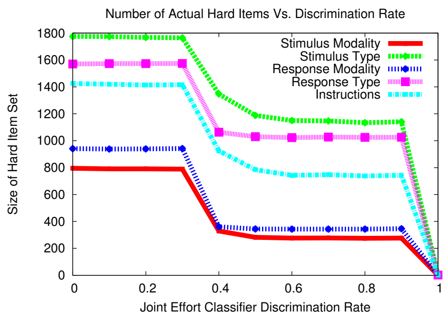
Additionally, we see a transition point over all five data sets at a discrimination rate of 0.3 where the R_1-score falls for both the *easy* and *hard* sets. An explanation for this is straightforward. As the confusion discrimination rate rises, the joint effort classifier switches its bias from sending all items to the *easy* set unless they're *obviously hard*, to sending all items to the *hard* set unless they're *obviously easy*. This means that a cohort of items with confusion scores near the baseline switch from the *hard* to the *easy* data set at a confusion rate of about 0.3. Since this cohort has a confusion that is lower than the *hard* data set, but higher than the *easy* data set, the confusion rate of both data sets falls, leading to a rise in both data sets R_1-score.

Considering the $\Delta V_{avg}/\Delta C_{avg}$ ratio presented in equation 6 for these five experiments, we might ask how much does the ratio of proportionality, R, vary and leave us with a useful strategy for joint effort. If R_β^h is the R_β-score of the *hard* data set and R^e_β is the R_β-score of the *easy* data set, then notice that each of the discrimination rates in Figure 5 can vary between R^e_β and R^h_β and that the joint effort classifier will still be able to usefully differentiate between a set that should be sent to humans and one that should be learned by machines. This means that for a single joint effort classifier/base-learner pair on a single data set and task, R can be anywhere in the interval between (R^h_β, R^e_β) obtained at each of the discrimination rates measured. Additionally, when all R^e_β are less than all R^h_β, which is the case for the items in our experiment, R can range between $(Min(R^h_\beta), Max(R^e_\beta))$.

Since each R^h_1 for each item is 1 when the discrimination rate is 1 all items with a confusion score less than 1 are sent to the *easy* set while items with confusion scores of exactly 1 are sent to the *hard* set. This means that in some cases the *hard* data set has either no items, or only a very small number of items. Therefore, we only allow a confusion score to be in the set of R^h_1 if its size is greater than 50 items.

Next, consider the maximum variance in R. Using the number of *hard* items in Figure 5 to decide if a score should be included in the set R^h_1, we see that the minimum R^h_1 cannot be drawn from a discrimination rate of 1. Taking this into account, the range of R for the five categories is: Stimulus Modality - (1.85, 8.62) , Stimulus Type - (1.30, 2.66), Response Modality - (2.35 , 5.0), Response Type - (1.42, 5.31), and Instructions - (1.72 , 3.39).

Finally, we also report the average accuracy (not F-score) of the meta learner in Figure 5. We find it interesting that the joint effort classifier accuracy behaved differently for two of the categories as the discrimination rate was adjusted when compared to the other three categories. For Stimulus Modality and Response Modality, the learner's accuracy went up as fewer items were included in the *hard* category. On the other hand, for Stimulus Type, Response Type, and Instructions, the opposite was the case. There were differences between these two categories and every other category, including the fact that they always had a smaller number of items chosen as *hard* at every discrimination rate and that there are fewer labels in both categories compared to their average cardinality.



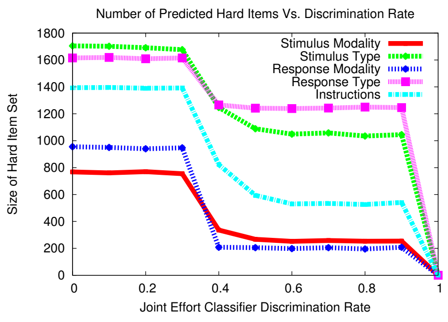


Figure 5: The size of the set of predicted *hard* items and the size of the set of actual *hard* items for each of the five categories in the CogPO ontology at diverse discrimination rates

All of our categories were dominated by items that had only a single correct label.

We believe that joint effort classifier accuracy increased for the two modality categories as the discrimination rate went up because both of these categories had a large number of *easy* items, even when the discrimination rate was low. This, indirectly, was due to the ratio of the number of labels in these two classes compared to their average cardinality. At a discrimination rate of about $0.3-0.4$ both of these categories went from identifying roughly half the items as *easy* or *hard* to identifying most as *easy*. Additionally, for the other three categories the opposite was the case. They went from needing to identify most items as *hard* at low discrimination rates, to identifying roughly half the items as *hard* and half as *easy* at higher discrimination rates. This seems to indicate that the joint effort classifier works best when there is a large asymmetry between the number of *easy* items and the number of *hard* items.

6. Conclusions and Future Work

We explored a strategy for human-machine joint effort — a generalization task where some labels might be labeled by a machine, while others are labeled by humans. Our strategy involved the use of a joint effort classifier to remove

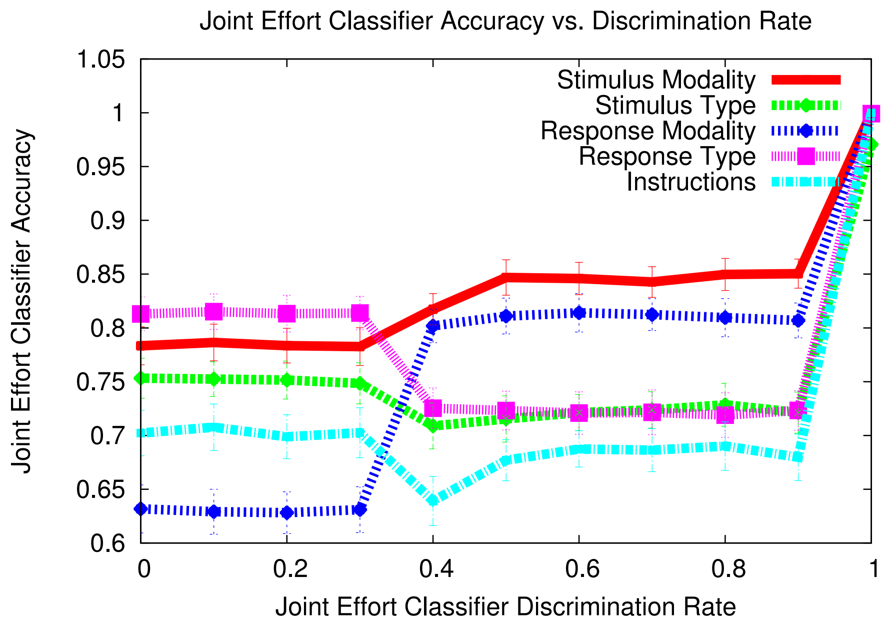


Figure 6: The accuracy of the joint effort classifier for the five categories in CogPO over diverse discrimination rates.

items that were *hard* for an underlying learner to label and sending those items to human labelers.

We developed a measure called the R_β-score that can determine when a joint effort strategy should be used. This score is the reciprocal of the C_β-score which is equal to $1 - F_\beta$. We demonstrated that the F_β-score ties into the value of correct and incorrect labels as well as the costs of correcting labels. We simulated the behavior of the R_1-score on a generalization task where the learner and joint effort classifier had varying competencies. We found that the joint effort strategy was most likely to be viable when both the underlying learner and joint effort classifier are accurate.

Finally, we applied our strategy to the BrainMap corpus with labels drawn from the CogPO ontology and found that the *easy* set determined by the meta learner had R_1-scores that were roughly 2 to 4.5 times higher than the *hard* items in the generalization task. We also posed two hypotheses for why the joint effort classifier was able to do this even though it had the same inductive bias as the underlying learner.

An area of further exploration could involve using a joint effort classifier with different inductive biases than the underlying learner to determine if items should be sent to the *easy* and *hard* categories. Additionally, our framework examines the question of balancing items between a costless automatic learner and expert human annotators and how that question relates to the confusion score and the constant of proportionality or R_β-score. Another line of exploration might look at the relationship between a costless automatic learner and a 'mid-skill' learner. Such a learner may be one like the humans presented by Snow et al. (2008) who are not able to provide a gold standard data set. Alternatively the learner may also include machine learners that consume a considerable amount of resources and provide good results without providing perfect results. Another interesting consideration involves re-training the underlying learner on only *easy* items to see if the confusion score on that learner can be reduced even further.

References

Chai, X., Deng, L., Yang, Q., Ling, C.X., 2004. Test-cost sensitive naive bayes classification, in: Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on, IEEE. pp. 51–58.

Chakrabarti, C., 2014. Artificial Conversations for Chatter Bots Using Knowledge Representation, Learning, and Pragmatics. Ph.D. thesis. University of New Mexico.

Chakrabarti, C., Jones, T.B., Luger, G.F., Xu, J.F., Turner, M.D., Laird, A.R., Turner, J.A., 2014. Statistical algorithms for ontology-based annotation of scientific literature. Journal of biomedical semantics 5, S2.

Chakrabarti, C., Jones, T.B., Xu, J.F., Luger, G.F., Laird, A.R., Turner, M.D., Turner, J.A., 2013. A probabilistic framework for ontology-based annotation in neuroimaging literature., in: Intelligent Systems in Molecular Biology Bio Ontologies Special Interest Group, Berlin, Germany.

Chekina, L., Rokach, L., Shapira, B., 2011. Meta-learning for selecting a multi-label classification algorithm, in: Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, IEEE. pp. 220–227.

Davis, J.V., Ha, J., Rossbach, C.J., Ramadan, H.E., Witchel, E., 2006. Cost-sensitive decision tree learning for forensic classification, in: Machine Learning: ECML 2006. Springer, pp. 622–629.

Domingos, P., 1999. Metacost: A general method for making classifiers cost-sensitive, in: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM. pp. 155–164.

Elkan, C., 2001. The foundations of cost-sensitive learning, in: International joint conference on artificial intelligence, Citeseer. pp. 973–978.

Fox, P.T., Lancaster, J.L., 2002. Mapping context and content: the brainmap model. Nature Reviews Neuroscience 3, 319–321.

Greiner, R., Grove, A.J., Roth, D., 2002. Learning cost-sensitive active classifiers. Artificial Intelligence 139, 137–174.

Hmida, M.B.H., Slimani, Y., 2010. Meta-learning in grid-based data mining systems. International journal of communication networks and distributed systems 5, 214–228.

Ibánez, A., Larranaga, P., Bielza, C., 2011. Predicting the h-index with cost-sensitive naive bayes, in: Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on, IEEE. pp. 599–604.

Kadlec, P., Gabrys, B., 2008. Learnt topology gating artificial neural networks, in: Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on, IEEE. pp. 2604–2611.

Kittler, J., Hatef, M., Duin, R.P., Matas, J., 1998. On combining classifiers. Pattern Analysis and Machine Intelligence, IEEE Transactions on 20, 226–239.

Laws, F., Scheible, C., Schütze, H., 2011. Active learning with amazon mechanical turk, in: Proceedings of the conference on empirical methods in natural language processing, Association for Computational Linguistics. pp. 1546–1556.

Ling, C.X., Sheng, V.S., Bruckhaus, T., Madhavji, N.H., 2006a. Maximum profit mining and its application in software development, in: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM. pp. 929–934.

Ling, C.X., Sheng, V.S., Yang, Q., 2006b. Test strategies for cost-sensitive decision trees. Knowledge and Data Engineering, IEEE Transactions on 18, 1055–1067.

Lomax, S., Vadera, S., 2013. A survey of cost-sensitive decision tree induction algorithms. ACM Computing Surveys (CSUR) 45, 16.

McCallum, A.K., 2002. Mallet: A machine learning for language toolkit. Http://mallet.cs.umass.edu.

Michie, D., Spiegelhalter, D.J., Taylor, C.C., 1994. Machine learning, neural and statistical classification. Ellis Horwood.

Núñez, M., 1991. The use of background knowledge in decision tree induction. Machine learning 6, 231–250.

Nushi, B., Alonso, O., Hentschel, M., Kandylas, V., 2014. Crowdstar: A social task routing framework for online communities. arXiv preprint arXiv:1407.6714 .

Parameswaran, A., Boyd, S., Garcia-Molina, H., Gupta, A., Polyzotis, N., Widom, J., 2014. Optimal crowd-powered rating and filtering algorithms. Proceedings Very Large Data Bases (VLDB) .

Pillai, I., Fumera, G., Roli, F., 2014. Learning of multilabel classifiers, in: Pattern Recognition (ICPR), 2014 22nd International Conference on, IEEE. pp. 3452–3456.

Rashtchian, C., Young, P., Hodosh, M., Hockenmaier, J., 2010. Collecting image annotations using amazon's mechanical turk, in: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, Association for Computational Linguistics. pp. 139–147.

Read, J., Pfahringer, B., Holmes, G., Frank, E., 2011. Classifier chains for multi-label classification. Machine learning 85, 333–359.

Ruta, D., Gabrys, B., 2000. An overview of classifier fusion methods. Computing and Information systems 7, 1–10.

Settles, B., 2010. Active learning literature survey. University of Wisconsin, Madison 52, 55–66.

Sheng, V.S., Ling, C.X., 2006. Feature value acquisition in testing: a sequential batch test algorithm, in: Proceedings of the 23rd international conference on Machine learning, ACM. pp. 809–816.

Smith-Miles, K.A., 2008. Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys (CSUR) 41, 6.

Snow, R., O'Connor, B., Jurafsky, D., Ng, A.Y., 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks, in: Proceedings of the conference on empirical methods in natural language processing, Association for Computational Linguistics. pp. 254–263.

Stolfo, S.J., Prodromidis, A.L., Tselepis, S., Lee, W., Fan, D.W., Chan, P.K., 1997. Jam: Java agents for meta-learning over distributed databases., in: KDD, pp. 74–81.

Tan, M., 1993. Cost-sensitive learning of classification knowledge and its applications in robotics. Machine Learning 13, 7–33.

Tsoumakas, G., Vlahavas, I., 2007. Random k-labelsets: An ensemble method for multilabel classification, in: Machine learning: ECML 2007. Springer, pp. 406–417.

Turner, J.A., Laird, A.R., 2012. The cognitive paradigm ontology: design and application. Neuroinformatics 10, 57–66.

Turner, M.D., Chakrabarti, C., Jones, T.B., Xu, J.F., Fox, P.T., Luger, G.F., Laird, A.R., Turner, J.A., 2013. Automated annotation of functional imaging experiments via multi-label classification. Frontiers in neuroscience 7.

Turney, P., 1995. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. Journal of Artificial Intelligence Research (JAIR) 2.

Vilalta, R., Drissi, Y., 2002. A perspective view and survey of meta-learning. Artificial Intelligence Review 18, 77–95.

Von Ahn, L., 2009. Human computation, in: Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE, IEEE. pp. 418–419.

Witten, I.H., Frank, E., 2005. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

Zhang, S., 2010. Cost-sensitive classification with respect to waiting cost. Knowledge-Based Systems 23, 369–378.