Probabilistic Graphical Modeling of Data-Dependent Annotator Accuracy for Active Learning

Nathan Hatch, Erik Wijmans – CS 8803 Class Project

April 26, 2018

Abstract

Collecting labels for active learning is often a noisy process, but the noise can be reduced with careful selection of annotators. Previous approaches to estimating annotator accuracy do not account for the possibility that accuracy depends on the latent class of the example being annotated. We propose using a partially observed Bayesian network to model class-dependent annotator accuracy, and we propose an algorithm that will (1) learn this model, and (2) apply it to select annotators for active learning queries. Experiments on Rotten Tomatoes movie review sentiment analysis indicate that this method accurately recovers the true annotator accuracies with very little training data, and that it significantly outperforms naive annotator selection in terms of sample efficiency.

1 Introduction

Active learning is a promising technique to address the problem of sample efficiency in machine learning applications. It has been shown that, by selectively choosing particular examples to be labeled, one can achieve good generalization performance with relatively few labeled training examples [Settles, 2011]. However, one major challenge to active learning in practice is that such labels are often noisy [Settles, 2011]. For example, researchers who collect a dataset using workers from Amazon Mechanical Turk cannot be sure of the accuracy of the labels given by these workers. In some settings, there is nothing to be done about this. However, if the setting allows some control over who annotates each particular example, it is interesting to consider annotator selection algorithms that can improve label accuracy.

As detailed in Section 3, prior work has approached this problem by estimating the overall accuracy of each individual annotator (e.g. Donmez et al. [2009]). Such approaches help, but they do not account for the possibility that annotator accuracy depends on the latent class of the example being annotated. For example, perhaps annotator A can reliably label examples from class X, but annotator B is more reliable for examples from class Y.

This kind of case-dependent probabilistic reasoning is a perfect application for probabilistic graphical models. In fact, Rzhetsky et al. [2009] propose two partially observed Bayesian networks for modeling the accuracy of sentence annotation in biological scientific writing. Such models could be useful for active learning.

In this work, we propose a two-step algorithm to improve label fidelity in noisy active learning settings, detailed in Section 4. In the first step, using "Model B" from Rzhetsky et al. [2009], we estimate the accuracy of each annotator, dependent on the latent class of the example under consideration. In the second step, we use this model during query selection to choose the annotator with the highest expected accuracy for each query example. In this way, we take advantage of the particular skills of each annotator.

For experimental evaluation, we apply this algorithm to the Kaggle challenge "Sentiment Analysis on Movie Reviews". This is a fully supervised dataset. As detailed in Section 5, we convert this to a noisy active learning problem by hiding the labels and introducing synthetic noisy annotators with controlled, class-dependent accuracies. Under this simplified setting, we are able to analyze the extent to which our graphical model recovers the true class-dependent accuracy of each annotator, as well as how the accuracy of this model affects overall label fidelity and classification performance. Our results, detailed in Section 6, indicate that our algorithm accurately recovers the true annotator accuracies with very little training data. Furthermore, when applied to active learning, this model significantly improves label fidelity and sample efficiency when compared to random selection of annotators. In fact, in situations where the annotators are very skewed (i.e. highly accurate for a few classes, but highly inaccurate for the rest) our method also outperforms naive, non-data-dependent approaches to estimating annotator accuracy.

In the next section, we give detailed background information about active learning.

2 Background: Active Learning

Sample efficiency is a major concern when training machine learning models. Complex models with many trainable parameters can be very powerful, but they require correspondingly large amounts of training data in order to avoid overfitting. For example, training neural networks with supervised statistical machine learning techniques requires datasets like ImageNet [Deng et al., 2009], which has over 14 million labeled examples. Collecting this many labeled examples is time consuming and expensive, so machine learning researchers are interested in algorithms that can achieve good performance with less annotation effort.

Active learning [Settles, 2012] is a promising technique to address this problem. In active learning, it is assumed that the machine learning model (a.k.a. "agent") can *interact* with the teacher by requesting labels for a particular example (or batch of examples). The examples for which it requests labels are called the *query*. The precise mechanism for querying has been formulated in a few different ways, but for the purposes of this paper we consider *pool-based active learning*.[1] Here, the agent has access to a pool of unlabeled examples from which it can choose a query. Once the query is chosen, labels for the query are obtained, and the newly labeled examples are used to update a machine learning model. By choosing queries in a clever way, perhaps an effective model can be trained after labeling only a small fraction of the pool of examples.

Much research in active learning is devoted to the question of how to choose queries. One straightforward approach is *uncertainty sampling*: choose from the pool the examples that the model is currently most uncertain about. Because many machine learning models make predictions based on an internal estimate of probability distributions, this is easy to do. For example, the output of a logistic regression model can be interpreted as a probability distribution over possible classes. Under uncertainty sampling, for each example in the pool, one would calculate the entropy of this distribution, then query the examples with highest entropy.[2]

Early active learning researchers assumed perfect annotators. That is, it was assumed

[1] One alternative formulation is *query synthesis*, wherein the agent asks for labels for examples that it creates from its own imagination.

[2] Uncertainty sampling has a few drawbacks, including a tendency to select uninformative outliers. Other querying methods have been proposed, but for the sake of simplicity, this paper focuses on uncertainty sampling.

that the labels given for query examples were the true labels. However, one of the biggest challenges of active learning in practice is that these labels are often noisy [Settles, 2011]. In the next section, we review recent work which has attempted to address this challenge.

3 Related Work

In this section, we review recent work which has attempted to address the challenge of noisy annotators, as well as other ways in which active learning has been combined with probabilistic graphical models.

Probabilistic graphical models (PGMs) can be combined with active learning in two main ways:

1. **Solving active learning problems with PGMs:** Given a problem (e.g. classification or regression) which we hope to solve in a sample-efficient way using queries, employ graphical models to help choose which queries will be most helpful to the learning algorithm.

2. **Solving PGM problems with active learning:** Given a distribution from which we can sample using (conditional) queries, efficiently learn the parameters or structure of a graphical model for that distribution.

This work falls into category (1). However, for completeness, we will first give a summary of previous work in category (2). Tong and Koller [2001a] consider using active learning to estimate a data distribution by learning the parameters of a Bayes net. In contrast to more recent, pool-based approaches to active learning, they consider the *selective active learning* model, wherein the learning agent can specify arbitrary assignments to a limited subset of random variables and obtain a sample of the remaining variables from the appropriate conditional distribution. Mathematically, they show that this can improve sample efficiency by allowing the learner to obtain many samples from relatively rare events. In a separate paper, the authors consider a related problem where the structure of the Bayes net must also be learned [Tong and Koller, 2001b].

Anderson and Moore [2005] consider a special case where the graphical model under consideration is a Hidden Markov Model (HMM). In this setting, we are given values of the observed nodes of the HMM, and we are allowed to query specific time steps to receive a (possibly noisy) label for the hidden state at that time step. Here, queries reveal previously hidden parts of a single sample, in contrast to Tong and Koller [2001a], where queries reveal entire samples taken from a conditional distribution. The authors identify three potential goals of interest:

- State estimation: Correctly identify as many hidden states as possible.
- Path estimation: Identify the single most likely chain of hidden states.
- Model estimation: Find the parameters of the HMM most likely to have generated this instance.

For each problem, they describe an efficient algorithm for choosing optimal time steps to query. The problem of model estimation is actually very similar to the general problem treated in Tong and Koller [2001a]. In both cases, the goal is to learn a model of a distribution using some kind of intelligent querying algorithm which takes advantage of the structure of the graphical model.

While such techniques for distribution estimation are an interesting area of research for PGMs, our approach falls into the other category identified above, **solving active learning problems with PGMs**. We are interested in addressing noisy active learning by modeling annotator accuracy.

Although recent work has addressed the challenge of noisy annotators, such works do not explicitly use graphical models. For instance, Sheng et al. [2008] adopt a model where each annotator returns a noisy label. Here, the parameters of the noise are known and identical for each annotator, and the annotators do not make correlated errors. The authors explore a set of heuristics for choosing which data points should be re-annotated and explore the difference in performance between re-annotating existing data versus annotating unlabeled data. They conclude that when annotators are very noisy, it is better to re-annotate existing data.

Donmez et al. [2009] extend this analysis to situations where label accuracy may vary across annotators. Their algorithm, IEThresh, decides the accuracy of a particular annotation based on whether it agrees with the majority of annotations for that example. For each new query, it selects the set of annotators whose average accuracy has an upper confidence bound above a threshold value. In this way, they are able to learn online the most accurate annotators together with the parameters of a classifier. In a later paper, Donmez et al. [2010] further relax their assumptions to allow annotator accuracy to vary over time (for example, due to fatigue). However, in both papers, the authors assume that there will be no correlation between the errors of different annotators, and they assume that the accuracy of an annotator does not depend on the latent class of the example being labeled. In contrast, our approach accounts for data-dependent accuracy by modeling each example with a Bayesian network that includes the latent class as an unobserved variable.

Our Bayesian network is taken from Rzhetsky et al. [2009]. In this paper, the authors introduce a dataset of sentences from biological scientific writing, each of which has been annotated by a number of expert scientists. Because these annotations often disagree, the authors devise two Bayesian networks to model the distribution from which the examples are drawn. Each network has predefined structure, and the parameters are learned based on the dataset of expert annotations. By taking the maximum likelihood assignment of the latent class, they can resolve disagreements among expert annotators without completely throwing away the example. For our work, we chose to use what they called "Model B" (see Fig. 1).

Although Rzhetsky et al. [2009] make explicit use of graphical models, they do not use the labeled examples to train a classifier. Instead, they evaluate their model based purely on whether its MLE for the latent class agrees with the majority vote of a held-out set of annotations. Hence, they are not doing active learning. Our contribution is to use their model in a true active learning setting. With Model B, we can make nuanced predictions of class-dependent annotator accuracy, which in turn allow us to collect a training dataset of high-fidelity labels in a sample-efficient way. In the next section, we describe the details of our approach.

4 Methodology

While Donmez et al. [2009] learn annotator accuracies while simultaneously training their classifier, our approach separates this into two distinct steps.

1. **Modeling annotators.** Train Model B on a densely annotated "warmup set" of examples. Use the majority vote from these examples to train an initial classifier.

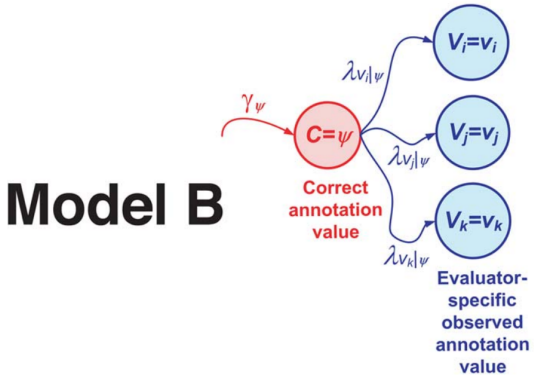

Figure 1: Our graphical model. The correct annotation is a latent node, and its children are the observed annotations from each annotator. Image from Rzhetsky et al. [2009].

2. **Active learning with annotator models.** Choose queries with uncertainty sampling, and use Model B to decide which annotator should label each query example.

The following subsections go into more detail on each of these two steps.

4.1 Modeling Annotators

We model annotators using Model B from Rzhetsky et al. [2009] (see Figure 1). This is a Bayesian network with predefined structure. Its parameters are a prior distribution over N possible classes, and for each of the R annotators, an $N \times N$ conditional probability distribution (CPD) for their annotation given the latent class. Given a set of partially observed samples from the network, we can learn its parameters using expectation maximization (EM).

To collect training samples, we choose a random set of M examples from our active learning pool. For each of these examples, we collect labels from each of the R annotators. Hence, this "warmup" set comes at a cost of $M \cdot R$ queries. M is a hyperparameter that must be chosen carefully; see Section 6.2 for experimental details. In principle, for each training example, one could collect annotations from only two or three annotators, relying on EM to account for the possible annotations that might have been given by the others. However, for simplicity, we decided to use dense annotation.

Because EM is sensitive to initial parameter settings, we chose an initialization that assumes each annotator is reasonably accurate. That is, our initial setting for each $N \times N$ CPD was $(0.8 - \frac{0.2}{N}) \cdot I + \frac{0.2}{N} \cdot \mathbf{1}$, where $\mathbf{1}$ denotes the $N \times N$ matrix of all ones.

4.2 Active Learning with Annotator Models

Once EM is complete, we can use the trained parameters to select annotators for active learning queries. For this section, we assume that the results of these queries are used to train a machine learning model which can give a probability distribution over labels for each example from the unlabeled pool.[3] We use uncertainty sampling to select queries, which means that we select the examples whose distributions according to our (partially trained) model have the highest entropy. Then we select which annotator to use for each query example, as follows.

[3]This assumption is quite reasonable. Many ML models, including Naive Bayes and deep networks, use such distributions in their decision rule.

We treat the classifier's prediction as a prior distribution $p_{\text{classifier}}$, then use the annotators' CPDs from p_{modelB} to choose the annotator with the highest posterior probability of providing an accurate label. More precisely, for each annotator i, we calculate their *expected accuracy* as a weighted sum of the diagonal entries of their CPD, where the weights are given by $p_{\text{classifier}}$:

$$\text{expected accuracy}_i = p(V_i = C) = \sum_{\text{classes } c} p_{\text{classifier}}(C = c) p_{\text{modelB}}(V_i = c | C = c)$$

Then we choose the annotator i with the highest expected accuracy.

One alternative approach would have been to guess that the latent class is the argmax over the classifier's prior distribution, then choose the annotator with the highest accuracy for that class:

$$\text{best annotator} = \arg\max_i p_{\text{modelB}}(V_i = \hat{c} | C = \hat{c})$$

where $\hat{c} = \arg\max_{\text{classes } c} p_{\text{classifier}}(C = c)$. However, this approach seems more fragile and less mathematically justified. This is true especially in the early stages of learning, when the classifier's predictions are not very accurate.

In both of the methods above, note the interaction with uncertainty sampling. We are sampling data points to label based on what the classifier is most uncertain about, but we are then using those predictions to determine which annotator is expected to have the best accuracy on that piece of data.

5 Experimental Setup

In Section 5.1, we introduce our task and dataset. Section 5.2 describes the annotators used during our experiments. Finally, Section 5.3 describes the Naive Bayes classifier used for most of the active learning experiments.

5.1 Rotten Tomatoes Sentiment Analysis

To examine our method, we use a sentiment analysis dataset comprising movie reviews from Rotten Tomatoes, which was originally introduced in Socher et al. [2013]. In this dataset, the goal is to predict the sentiment of phrases from movie reviews. Sentiments range from highly positive to highly negative. See Figure 2 for examples from the dataset.

We use the training data from the Kaggle "Sentiment Analysis on Movie Reviews" competition and create a 70/30 train-val split for our experiments.

5.2 Noisy Annotators

Our dataset is fully supervised. We convert this to a setting with noisy labels by annotating the dataset with synthetic annotators. The annotators have known error rates and make class-specific errors. To do this, we introduce four groups of five noisy annotators each. Each annotator is particularly good at predicting one of the five sentiments. Let $p(V_i = v | C = c)$ be the probability that annotator i emits an annotation of v given that the correct label is c. For each annotator, we choose the diagonal entries $p(V_i = c | C = c)$ for all sentiments c according to the schemes specified in Table 4. The remaining entries of each row of the CPD are uniformly random: $p(V_i = c' | C = c) = \frac{1 - p(V_i = c | C = c)}{4}$ for all $c' \neq c$. See Table 5 for an example of a complete CPD.

Phrase	Sentiment
Its generic villains lack any intrigue -LRB- other than their funny accents -RRB- and the action scenes are poorly delivered	Highly Negative
There 's very little sense to what 's going on here , but the makers serve up the cliches with considerable dash	Negative
Cattaneo should have followed the runaway success of his first film , The Full Monty , with something different.	Neutral
Fresnadillo has something serious to say about the ways in which extravagant chance can distort our perspective and throw us off the path of good sense	Positive
Over the years , Hollywood has crafted a solid formula for successful animated movies , and Ice Age only improves on it , with terrific computer graphics , inventive action sequences and a droll sense of humor .	Highly Positive

Figure 2: Examples from Sentiment Analysis dataset

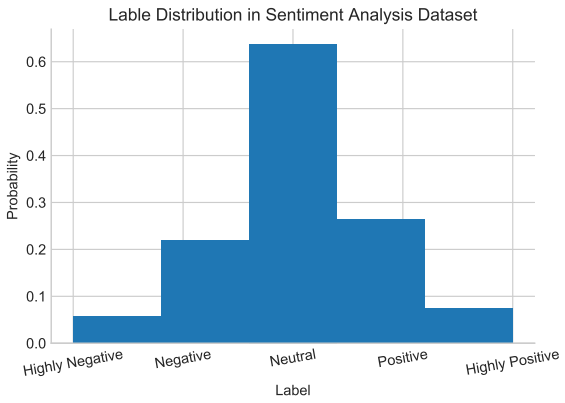

Figure 3: Density plot of the label distribution in the Sentiment Analysis dataset. This shows the dataset is heavily skewed in favor of neutral sentiments.

Annotator Group	$p(1\|1)$	$p(2\|2)$	$p(3\|3)$	$p(4\|4)$	$p(5\|5)$
Good annotators	0.95	0.9	0.85	0.8	0.75
Mediocre annotators	0.9	0.8	0.7	0.6	0.5
Bad annotators	0.6	0.5	0.4	0.3	0.2
Skewed annotators	0.95	0.2	0.2	0.2	0.2

Figure 4: Complete specification of the diagonal entries of the CPDs used for Annotator 1 in each of the four annotator groups considered in our experiments. Diagonal entries for other annotators are rotations of the diagonal entries for Annotator 1. Annotator 2 is best at predicting sentiment number 2, Annotator 3 is best at predicting sentiment 3, and so on.

$V_1 \backslash C$	0	1	2	3	4
0	0.95	0.0125	0.0125	0.0125	0.0125
1	0.025	0.9	0.025	0.025	0.025
2	0.0375	0.0375	0.85	0.0375	0.0375
3	0.05	0.05	0.05	0.8	0.05
4	0.0625	0.0625	0.0625	0.0625	0.75

Figure 5: Example full cpd, $p(V_1|C)$, for annotator 1 in the group of good annotators.

5.3 Naive Bayes Classifier

For the sentiment analysis experiments, we used a very simple Naive Bayes model for document classification. Under the Naive Bayes model, we treat each phrase as an independent set of identically distributed words. The distribution from which we draw each word is assumed to have the form of a simple Bayesian graphical model with a single latent class (representing the sentiment of the phrase) and one child node (representing the observed word).

In contrast to the model for the annotators, this classifier is trained with fully observed data. During the warm-up period, we use the label given by the majority vote of the dense annotations. During the active learning period, we use the label given by our chosen annotator. In either case, because the model is fully observed, we can calculate the maximum likelihood estimate (MLE) for its parameters according to the usual sufficient statistics for discrete Bayesian networks.

One challenge with using MLE for Naive Bayes is when, for a certain sentiment, we encounter a word that has never been seen before. In this case, the model will predict that the probability of that sentiment for the current phrase is zero, regardless of the evidence provided by other words in the phrase. To counteract this, we actually use Bayesian estimation for this network. Since the conjugate prior for discrete Bayesian networks is the Dirichlet prior, we use a Dirichlet prior with concentration parameters $\alpha_{c,w} = 1$ for all sentiments c and all words w. This ensures that for every sentiment, there is at least some probability of encountering every word in the vocabulary.

The rationale for using such a simple classifier was to ensure that we could focus on the effects of our annotator selection algorithm. In other words, we are not interested in finding the best possible classifier for sentiment analysis. However, as discussed in Section 6.2, perhaps in retrospect this was not a good idea. Our annotator selection algorithm depends on having a good prior distribution over the classes for an unlabeled example, for which we depend on having a good classifier. Hence, in section 6.3, we perform additional experiments on a different dataset, with a classifier better suited to the dataset.

6 Results

6.1 Modeling Annotators

Here we will investigate the ability of Model B to model our annotators. Specifically, we will investigate two questions:

- How much data do we need to model the annotators?
- How do the annotators themselves affect our ability to model them?

We will use two metrics to answer these questions:

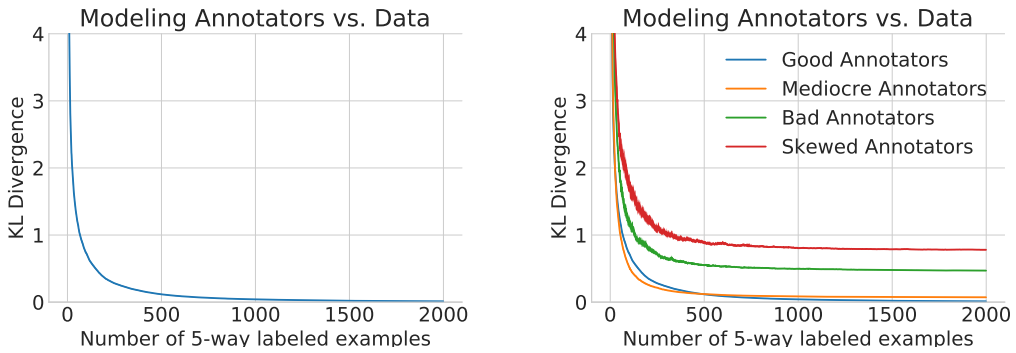

Figure 6: This figure shows the KL divergence vs. the number of 5-way annotated examples. **Left:** Our model is able to very quickly pull out the true probability distributions for each annotator. It takes 539 ± 190 5-way annotated examples for the model to find the best annotator for each class on average. **Right:** For the mediocre and good annotators, the model is able to learn the true probability distributions. However, the KL divergence does not converge to 0 for the bad and skewed annotators. It takes 539 ± 190, 339 ± 107, 771 ± 8, and 155 ± 40 5-way annotations on average for the model to find the best annotator for each class for the good, mediocre, bad, and skewed annotators, respectively.

1. The average KL divergence between the predicted probability distributions and the ground truth probability distributions

2. The number of data-points needed on average to find the best annotator for each class

Figure 6 shows the results. For the group of good annotators, the model is able to very quickly learn the true probability distributions for each annotator and rapidly drives the KL divergence to zero.

Now we will examine how we well we can model our 4 different groups of annotators. The striking part of these results is that the group of good annotators takes the significantly more data than the group of skewed annotators to be sufficiently modeled, despiting being the group that has the KL divergence converge to 0 the fastest. This is most likely due to the difference in accuracy between the best and second best class for each annotator being 5% for the good annotators, while being 10% for the other groups (and 75% for the skewed annotators). Smaller differences in accuracy are probably more difficult to learn. The other striking result is that the model is unable to fully model the groups of bad and skewed annotators. For the group of skewed annotators, the model is able to very quickly figure out which annotator is best for any given class, but is never able to fully model them. This is most likely due to the skewed annotators being no better than random at their worst classes. The model does quite bad overall on the group of bad annotators. It takes a large number of annotations to find the best annotator for each class, and it never finds the true accuracy distribution for each annotator. This is most likely due to the bad annotators being too noisy for the model to be able to separate signal from noise.

6.2 Active Learning with Annotator Models

Given the high accuracy of our annotator models in the previous section, we next wish to see whether these models can help select annotators during active learning. We compare the following approaches:

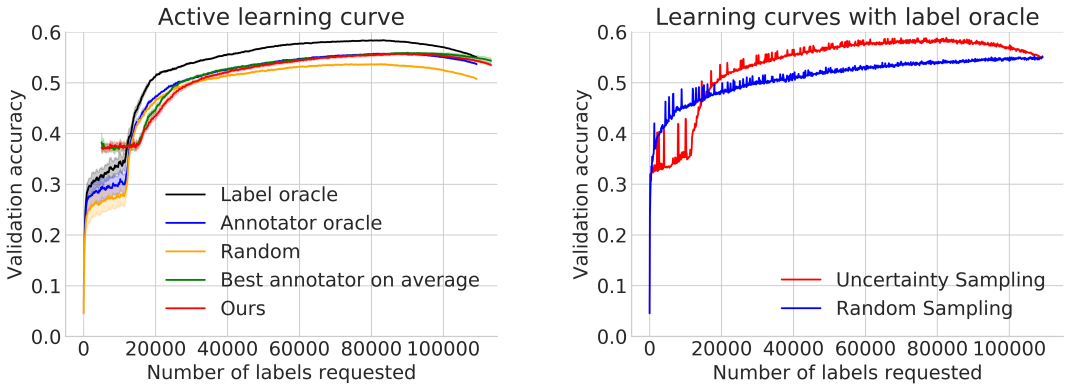

Figure 7: **Left:** Performance of active learning as a function of number of requested labels. Warm-up size is 1000, annotators are "mediocre", and we use batched queries of 100 examples. We report mean and 95% confidence interval over 13 trials. **Right:** Random sampling vs. uncertainty sampling for a single trial. Note the spikes and the strange shape of the uncertainty sampling curve.

- **Label oracle.** Suppose we can obtain the ground-truth labels.
- **Annotator oracle.** This method is the same as our approach, except that the CPDs used to model each annotator are the ground-truth CPDs.
- **Random.** For each query example, select one annotator at random and use the label they provide.
- **Best annotator on average.** Based on the warm-up set, find the annotator who agrees most often with the majority vote. Use that annotator for every query example. This is similar to the approach of Donmez et al. [2009] in that it does not model data-dependent annotator accuracy.
- **Ours.** For each query example, select the annotator with the highest expected accuracy given our annotator model and our classifier's current estimate of the probability distribution for the example. See Section 4 for details.

Only the last two approaches require a warm-up set. Hence, they suffer a "warm-up penalty" required to densely label the warm-up set. Results for a warm-up size of 1000 examples and "mediocre" annotators (see Section 5.2) are shown in Figure 7 (left).

All approaches show a learning curve of unusual shape, including a plateau at the beginning and degrading performance at the end. We believe this is an artifact of the combination of uncertainty sampling with the sentiment analysis dataset. At the beginning, the maximum-entropy examples tend to be single-word phrases containing rare words: e.g. "cascade", "first-rate", and "stillborn". Labeling these phrases does not help much with the rest of the dataset. Once these examples have all been labeled, uncertainty sampling moves on to more informative, multi-word phrases: e.g. "veiling tension", "drastic iconography", and "aimed mainly". This causes a period of steep improvement that tapers off gradually. Finally, towards the end, uncertainty sampling selects long, complicated sentences with unusual grammatical structures:

- "quietly lyrical tale probes the ambiguous welcome extended by Iran to the Afghani refugees who streamed across its borders , desperate for work and food ."

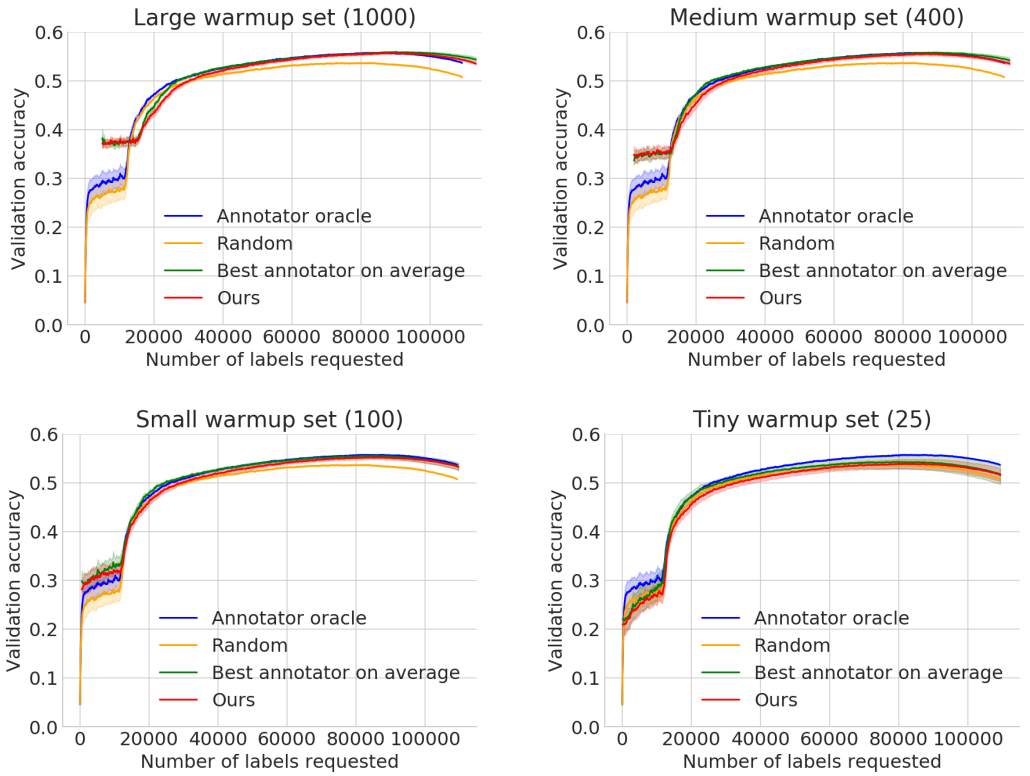

Figure 8: Warmup size vs. active learning performance (mediocre annotators).

- "Releasing a film with the word ‘ dog ’ in its title in January lends itself to easy jokes and insults , and"
- "wheedling reluctant witnesses and pointing his camera through the smeared windshield of his rental car"

We theorize that the Naive Bayes classifier has trouble modeling these structures, so when it tries to incorporate these examples into its predictions, its performance degrades. Figure 7 (right) compares uncertainty sampling against random sampling, which shows a more typical pattern of exponential decay. Note that, after labeling approximately 30% of the training data, uncertainty sampling actually *surpasses* the accuracy that we achieve after obtaining labels for the entire dataset, which is an unexpected and intriguing argument in favor of active learning.

Returning to Figure 7 (left), we note that the warm-up penalty means that our method leaves the "high-entropy plateau" later than the random and annotator oracle baselines. Thus, for a short period, our method performs worse than both. However, after around 30,000 labels, it surpasses the random baseline and begins to approach the accuracy of the oracle.

Choosing a good warm-up size is an important part of our approach, which balances the accuracy of the annotator model against the cost of the dense labels that train it. In Figure 8, we compare the empirical performance of various warm-up sizes. For mediocre annotators, it appears that a warmup size of more than 100 imposes a severe warm-up penalty, while a size less than 100 is not enough to improve the annotator model beyond the random baseline. Choosing a warm-up size of 100 strikes a good balance.

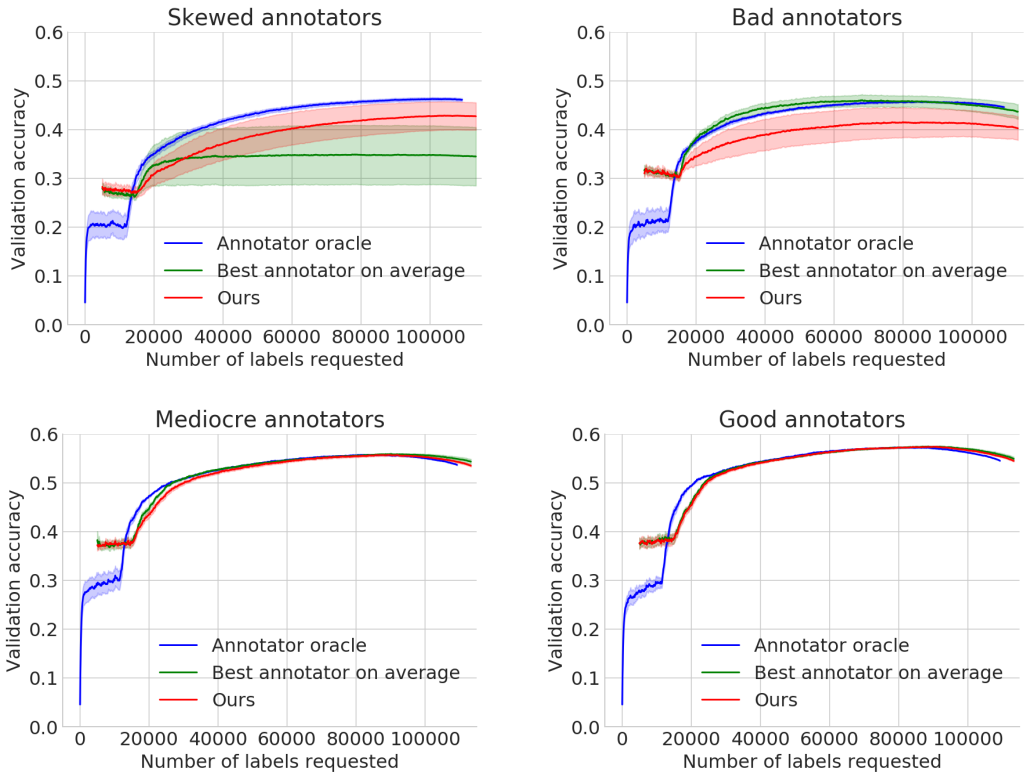

Figure 9: Annotator accuracy vs. active learning performance (warmup size 1000).

In all of the previous experiments, which were performed using mediocre annotators, our approach is outperformed by the "best annotator on average" baseline, which does not model data-dependent annotator accuracy. We theorize that this is due to the skewed nature of the sentiment analysis dataset. The best annotator on average will be the annotator that performs best on "neutral" sentiments. For the set of mediocre annotators, this annotator will also perform reasonably well on the other classes, given how rare they are. To test this, we also compared these active learning approaches for the other three sets of annotators discussed in Section 5.2.

Results are shown in Figure 9. We outperform the "best annotator on average" when the annotators are very skewed (very high accuracy for one class, but very low for all of the others). For reasonably good annotators, both methods perform well.

6.3 Additional Dataset – MNIST

As discussed above, our choice of toy dataset may have not been the wisest. We therefore redo our experiments on the MNIST handwritten digit dataset. We use a simple model consisting of PCA reduction to 40 dimensions and then a random forest of 120 trees. This model is able to achieve around a 3% validation error. We will only examine the group of good annotators on this dataset. In order to make the experiment similar, we add 5 new annotators to the group, so there are 10 annotators in total. For this group of annotators, $p(1|1) = 0.95$ and $p(10|10) = 0.5$, and all other classes are spaced evenly between these.

Figure 10 shows our results on MNIST. Our method is able to outperform the best annotator on average in terms of both sample efficiency and final accuracy. This is due to two main differences. First, the learner is significantly more accurate, even with only a

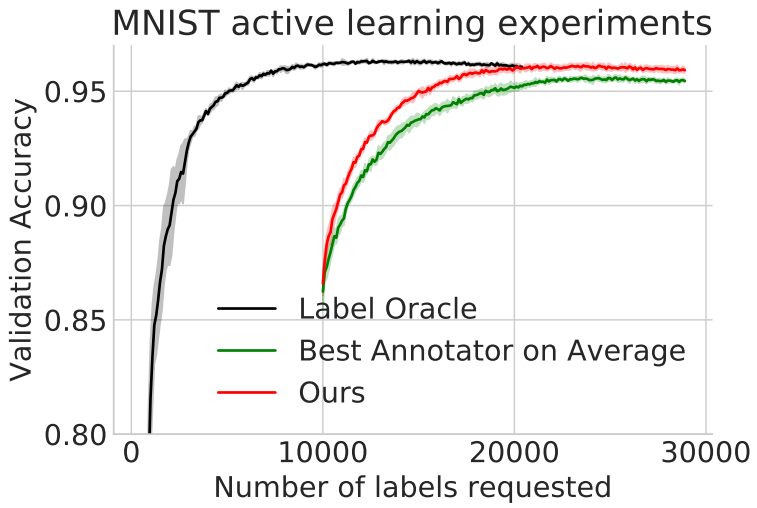

Figure 10: Results of active learning experiments on the MNIST dataset. Our method (red line) is able to outperform the strong base-line of best annotator on average in terms of both sample efficiency and final validation accuracy.

few training examples (86% with 1000 unique instances). This means that our assumption that the classifier has a reasonable idea of the class for the unlabeled piece of data is true here. (That assumption is significantly more suspect for the sentiment analysis dataset.) Second, the worst class for a given annotator is much worse, and learning from those labels (as the best-annotator-on-average approach must do) is therefore much harder.

7 Conclusion and Future Work

In this work, we first explored modeling annotator accuracies with a probabilistic graphical model by reimplementing the work of Rzhetsky et al. [2009] and then using a fully supervised dataset along with synthetic annotators to validate their model and results (something that was not properly done in the original paper). Next, we explored using this model of the annotators for active learning, building on the work of Donmez et al. [2009].

We found that the graphical model proposed by Rzhetsky et al. [2009], Model B, is indeed able to very accurately model the annotators with a very small amount of data when the annotators are quite good. The model is not able to fully model the annotators when they are quite poor; however, it can still still model them well enough to find the best annotator for each class.

We then applied this model to active learning by using it to select the best annotator for a piece of unlabeled data. For the sentiment analysis dataset, we found that for certain groups of annotators, our method outperforms the baseline of selecting the annotator in a non-data-dependent way (i.e. "best annotator on average"). In particular, when annotators are very skewed (highly accurate for a few classes, but highly inaccurate for the others), having a data-dependent model of annotator accuracy leads to a significant improvement in validation accuracy and sample efficiency.

However, when the difference in performance of the annotators is more subtle, our approach is outperformed by the approach that simply selects the best annotator on average. We theorize that this null result was because of two issues:

1. The sentiment analysis dataset is very class-imbalanced, so the annotator that does best at predicting neutral sentiments (class 2), does very well on average.

2. Sentiment analysis is a difficult task. During early stages of active learning, our naive Bayes model wasn't able to provide an accurate guess of the true class of the unlabeled piece of data.

We then choose to explore our method on a different dataset that eliminates these two issues. Specifically, we applied our method on the MNIST handwritten digit dataset using a different classifier: a Random Forest applied to a 40-dimensional PCA reduction of the data. This classifier does a much better job on MNIST than Naive Bayes does on sentiment analysis, suggesting that its predicted class distributions might be a better prior when selecting annotators during active learning. Indeed, on this problem, our method outperforms the best annotator on average.

In future work, one could explore on-line learning of Model B so our model is able to adapt to potentially time-varying accuracies of the annotators. On-line learning could also ameliorate the warm-up cost of collecting densely labeled samples, similar to Donmez et al. [2009]. One could also imagine a hybrid approach of best-annotator-on-average and our approach: using the best annotator on average when the classifier is still poor, but switching to our method once the classifier starts doing well.

In general, it will be important to analyze which combinations of classifiers and datasets lead to good performance. Certain datasets, like Rotten Tomatoes sentiment analysis, exhibit strange interactions with entropy-based query selection algorithms. For example, uncertainty sampling tends to focus on outliers. In future work, we will control for these effects by investigating alternatives to uncertainty sampling that behave in a more consistent way.

A Appendix: Implementation Details

A.1 Annotator Modeling

In order to learn the parameters of the graphical model of the annotator's accuracies (Model B), we used Expectation Maximization. We implemented both a general case of Expectation Maximization on calibrated clique trees built on top of PGMPY[4] and a faster version, specific to the structure of Model B.[5] We choose to build on top of PGMPY in order to produce code that is potentially useful outside this project and can be contributed to that open-source project. We also implemented a significantly faster version of EM, specific to Model B, in order to do more ablation studies.

We implemented the synthetic annotators, all the infrastructure to have them label data, and all the various methods for selecting which annotator is best ourselves. This includes calculating the best annotator on average according to agreement with the majority vote on the warmup set, and calculating the annotator with the highest expected accuracy for a given training sample.

A.2 Active Learning

All code related to our active learning experiments on the sentiment analysis dataset, i.e. Naive Bayes, annotator strategies, unlabeled pool (query) selection strategies, was implemented by us. We implemented several query selection strategies, including entropy-based uncertainty sampling, argmax uncertainty sampling, and random sampling.

We used Sci-kit Learn's implementation of PCA and Random Forests for our additional MNIST experiment.

We did implement an LSTM for sentiment analysis, however, it's still relatively poor performance (maximum of around 68% on the validation set), and relatively massive training time lead us go down the MNIST/PCA/Random forest route instead.

We repeated each of these experiments a minimum of twelve times in order to calculate a 95% confidence interval for each of our learning curves.

A.3 Datasets

We implemented a dataloader for the biological sentence annotation dataset from Rzhetsky et al. [2009], which was not a simple task given their bizarre HTML-table format for the dataset. However, the design of that dataset made it difficult to use with active learning, so we switched to Rotten Tomatoes sentiment analysis and wrote another dataloader.

[4] https://github.com/pgmpy/pgmpy

[5] We would like to point out that we could have used our own implementation of calibrated clique trees as we both choose to do message passing on calibrated clique trees for Homework 2.

References

Brigham Anderson and Andrew Moore. Active learning for hidden markov models: Objective functions and algorithms. In *Proceedings of the 22nd international conference on Machine learning*, pages 9–16. ACM, 2005. Available online.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

Pinar Donmez, Jaime G Carbonell, and Jeff Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 259–268. ACM, 2009. Available online.

Pinar Donmez, Jaime Carbonell, and Jeff Schneider. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 826–837. SIAM, 2010. Available online.

Andrey Rzhetsky, Hagit Shatkay, and W John Wilbur. How to get the most out of your curation effort. *PLoS computational biology*, 5(5):e1000391, 2009. Available online.

Burr Settles. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 1–18, 2011. Available online.

Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012. Available online.

Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622. ACM, 2008. Available online.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

Simon Tong and Daphne Koller. Active learning for parameter estimation in bayesian networks. In *Advances in neural information processing systems*, pages 647–653, 2001a. Available online.

Simon Tong and Daphne Koller. Active learning for structure in bayesian networks. volume 17. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001b. Available online.