Leveraging In-Batch Annotation Bias for Crowdsourced Active Learning

Honglei Zhuang
LinkedIn Corporation
University of Illinois at Urbana-Champaign
hzhuang3@illinois.edu

Joel Young
LinkedIn Corporation
joyoung@linkedin.com

ABSTRACT

Data annotation bias is found in many situations. Often it can be ignored as just another component of the noise floor. However, it is especially prevalent in crowdsourcing tasks and must be actively managed. Annotation bias on single data items has been studied with regard to data difficulty, annotator bias, etc., while annotation bias on batches of multiple data items simultaneously presented to annotators has not been studied. In this paper, we verify the existence of "in-batch annotation bias" between data items in the same batch. We propose a factor graph based batch annotation model to quantitatively capture the in-batch annotation bias, and measure the bias during a crowdsourcing annotation process of inappropriate comments in LinkedIn. We discover that annotators tend to make polarized annotations for the entire batch of data items in our task. We further leverage the batch annotation model to propose a novel batch active learning algorithm. We test the algorithm on a real crowdsourcing platform and find that it outperforms in-batch bias naïve algorithms.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; J.4 [**Social and Behavioral Sciences**]: Psychology

Keywords

Crowdsourcing; active learning; annotation bias

1. INTRODUCTION

Online crowdsourcing platforms leverage Internet users across the world to provide a scalable method for annotating data sets for various machine learning tasks. There are several systems providing online crowdsourcing services, *e.g.* Amazon Mechanical Turk [1] and CrowdFlower [2]. Although far cheaper than employing and training expert annotators, crowdsourcing can still be expensive as building high-performance classifiers often requires large sets of annotated data with multiple annotations for each data item. To minimize these costs, we turn to active learning, a technique for selecting particular unlabeled data instances for labeling to (hopefully) best improve classifier performance. Unfortunately, traditional active learning methods assume reliable annotators—this assumption is not feasible with crowdsourcing. A number of studies [12, 24, 26, 29] have discussed the topic of overcoming the annotation errors of crowds on single data items in active learning.

In addition to each individual annotator's annotation bias, we find interference between data items simultaneously presented for annotation through crowdsourcing. There are many situations in which batches of multiple data items are judged by crowds at the same time. A typical example is when evaluating results of a search engine given a certain query, the retrieved web pages are judged by crowds (either by explicit labeling or implicit click through rate) in batches. Other examples include object recognition [22] and clustering [8]. Batch active learning is particularly vulnerable as multiple data items are submitted simultaneously for annotation both to reduce annotation costs and to minimize classifier retraining cycles.

Although the annotation bias of individual annotators on single data items is well explored [15, 16, 28], there is little published research on annotation bias introduced by presenting data items in batches to crowd annotators. Scholer *et al.* [18] explored factors such as time intervals between two judgments that may affect the quality of annotation in a TREC data set, but without explicitly modeling the annotation bias.

In this paper, we study in-batch annotation bias; that is, the annotation interference between items in small sets of data items. As a motivating example, consider Table 1 showing annotation results from a crowdsourced task to identify inappropriate comments on LinkedIn posts. Batch 1 (left column) and Batch 2 (right column) capture two batches submitted separately for crowd annotation as to which comments are inappropriate. Both batches are sampled from comments on the same post. The *Label* columns show the crowdsourced annotation results. Although the final comment in each batch is identical, they were annotated differently. In Batch 1, it is labeled as an acceptable comment, probably because it is presented to the crowds along with all the other comments containing URLs. However, in Batch 2, it is labeled as inappropriate, perhaps as it is the only com-

[1] `https://www.mturk.com/`

[2] `http://www.crowdflower.com/`

WSDM'15, February 2–6, 2015, Shanghai, China.
ACM 978-1-4503-3317-7/15/02.
http://dx.doi.org/10.1145/2684822.2685301.

Table 1: Motivating example. The left column and the right column are two batches of data items sent to crowdsourcing for annotation of inappropriate comments. All data items are from comments on the same post. A "Yes" in the *Label* column indicates the comment is labeled as inappropriate by the crowds, while a "No" means the comment is acceptable. The last comments of both batches are the same comment yet are assigned different labels by the crowds. Annotations were generated by majority voting by annotators with greater than 70 percent accuracy on co-mingled quiz batches.

Batch 1		Batch 2	
Label	Comment content	*Label*	Comment content
Yes	`www.youtube.com/watch?v=BKorP55Aqvg`	No	Even after doing all this, sometimes it still doesn't work I might add. It is part of the job.
No	something related: `https://www.youtube.com/watch?v=BKorP55Aqvg`	No	Chernobyl nuclear power plant was probably engineered using "quality (sic !), speed and cost"...
No	`https://www.youtube.com/watch?v=BKorP55Aqvg` Its tough to be an engineer!	No	I think a culture where folks can air grievances can be very productive as long as they can be accumulated to identify themes and trends. Otherwise, ... *[Omitted]*
Yes	`http://youtu.be/BKorP55Aqvg`	No	Brian, my favorite saying as an engineer is: "I told you so 100 times." You should put that at the top of your list :)
No	Now from the perspective of the engineer: `https://www.youtube.com/watch?v=BKorP55Aqvg`	**Yes**	Now from the perspective of the engineer: `https://www.youtube.com/watch?v=BKorP55Aqvg`

ment in this batch containing a URL. Although both batches are given to the crowds with exactly the same instructions, the crowds can still make incoherent judgments. We focus on the case of small batch (*e.g.* size $k \leq 5$), as annotators are unlikely to see a large batch of data items at the same time. We leave the case of larger batches to future study.

There are three major questions to address: Do annotations of data items in the same batch interfere with each other? If we find in-batch annotation bias, can we quantitatively measure the bias? Regarding the impact of in-batch annotation bias, how can we design a batch active learning algorithm that is still effective? We present the following contributions exploring these questions:

1. *Verifying and modeling in-batch annotation bias.* We conduct preliminary studies exploring the existence of in-batch annotation bias. We then propose a factor graph model to capture the annotating behaviors of crowd annotators of data items in batches.

2. *Proposing active learning algorithms with in-batch annotation bias.* Based on the proposed annotating model, we propose a novel batch active learning algorithm, leveraging the in-batch annotation bias to help improve the classifier performance, and prevent it from being hurt by biased annotation.

3. *Real-world experiments on a crowdsourcing platform.* We conduct experiments on an online crowdsourcing platform to actively build a classifier for inappropriate comments identification on LinkedIn data set.

The rest of this paper is organized as follows: Section 2 briefly introduces the background and the data set used, defines the research problem, and presents preliminary experiments on verifying in-batch annotation bias; Section 3 proposes a factor graph based annotation model; Section 4 proposes a novel batch active learning algorithm with in-batch annotation bias; Section 5 shows the experimental results; Section 6 introduces related work and Section 7 concludes.

2. PRELIMINARIES

In this section, we briefly introduce the problem space, the data set used, and refine the particular active learning problem we are trying to solve. We then refine our definition of "in-batch annotation bias" referring to the interference between annotations of different data items presented together to the annotators. From this foundation we present a series of studies to verify the existence of in-batch annotation bias, and how the data items in a batch may interfere with the annotation of each other.

2.1 Background and Data Set

LinkedIn is the world's largest professional network service operator, with more than 300 million members in over 200 countries and territories [3]. LinkedIn allows users to comment on posts published by both companies and LinkedIn influencers [4]. Unfortunately, a certain portion of the comments might be spams, offensive, or otherwise inappropriate to display. To maintain a high-quality member experience, it is critical to identify and prevent these comments from being displayed.

We employ a supervised classifier (*e.g.* logistic regression) trained offline, which requires a sufficiently large set of annotated data. In order to collect training data, we use CrowdFlower for data annotation. Annotators are provided an annotation codebook with definitions of inappropriate comments, as well as several examples of both inappropriate and acceptable comments. In this task, we define inappropriate comments as promotional, profane, blatant soliciting, random greeting comments, as well as comments with *only* web links and contact information. Comments on the same post are grouped together and divided into batches, where each batch consists of up to five comments. All comments in a batch are displayed simultaneously to the annotators.

We sample a data set of 431,525 comments on 89,355 posts of companies and influencers in LinkedIn. 128,587 of them are comments on "company updates", 48,207 of them are on "influencer updates" and 254,731 are on "influencer articles". In order to track the performance of crowd annotators, we construct a ground truth data set with 2,835 comments. The ground truth data set is annotated by 9 trained LinkedIn employees (experts) using the same codebook and interface as used for the crowdsourcing experiments. Each comment is classified as either *inappropriate* or *acceptable*. The average Cohen's kappa for all annotator pairs is 0.7881.

[3] `http://press.linkedin.com/about`

[4] LinkedIn influencers are a selected set of industry and thought leaders.

2.2 Batch Active Learning

Active learning algorithms iteratively choose unlabeled data items for labeling by an external oracle (*e.g.* expert human annotators) and then fold the new labeled data back into the training data set. A model is retrained on the updated training set. When more than one data item is chosen in an iteration it is called *batch* active learning. More precisely, suppose the unlabeled data set is U, labeled set is L, and the labels of L are in y_L. A typical active learning algorithm chooses a subset $A \subset U$ with a given size k and obtains their labels y_A from an oracle. The selected data is then moved from U and added to L with their labels y_A concatenated to y_L.

Existing active learning algorithms often assume the annotation provided by the oracle is reliable. However, this assumption is rarely true, especially when crowds are employed as the oracle. Let A be some subset of data items and y'_A (different from y_A, which is the ground truth labeling) is their annotations drawn from some round of crowdsourcing. Then, treating crowdsourcing as a random process, we regard y'_A as a random variable described by distribution $q(y'_A|A)$. In this light, our objective is now to design an active learning algorithm which leverages $q(y'_A|A)$ while choosing unlabeled data items in each iteration. More formally,

Problem 1. **Batch active learning from biased crowds.** Let L be a data set with known labels y_L, and U be an unlabeled data set where each item i in L and U is described by feature vector X_i. For a set of data A consisting of k items, a crowd provides a (biased) annotation drawn from distribution $q(y'|A)$. The objective is to choose the best subset $A^* \subset U$ satisfying $|A| = k$, so that after adding A and the annotation given by the biased oracle y'_A into L and y_L, the performance of classifier trained on the new labeled data is optimized.

Before we develop the active learning algorithm, we need to first understand the crowds' annotation behavior, and model their annotating distribution $q(y'|A)$.

2.3 In-Batch Annotation Bias

Annotation bias has been observed in multiple fields, including machine translation [4], image labeling [27], and information retrieval [18]. Most studies focus on annotation bias on a single data item, resulted from variance across annotators. This bias can often be corrected either by collecting data from multiple annotators or carefully choosing annotators.

However, due to limits of time and cost, or the nature of data, data items are also often organized into small batches, and presented to the annotators at the same time. This particular way of presenting data items can introduce additional annotation bias, which cannot be fixed using multiple annotators. Suppose the batch is represented by a set of data items $b = \{b_i\}$. If the annotation from the crowd annotators on a certain data item b_i in the batch, represented by y'_i, follows a different distribution from the annotation on the same data item but displayed alone, or from the distribution when it is assembled with different other data items in batch b', then there is a certain kind of annotation bias introduced by the way we display data items. For example, in our inappropriate comment identification task, if an acceptable comment is presented along with a whole batch of inappropriate comments, it might be more likely to be mistakenly chosen as inappropriate; it could also be more likely to be correctly labeled, if other comments in the same batch happened to provide some examples as *reminders* of some subtle rules of inappropriate comments (*e.g.* the comment containing URL in Table 1). In contrast, if a comment is presented alone, annotators are more likely to make relatively independent judgments, but also lose the possibility to be implicitly taught by other comments in the same batch. We refer to this potential annotation bias as "in-batch annotation bias".

Verifying in-batch annotation bias. Based on the ground truth data for inappropriate comment identification task, we conduct experiments verifying different factors that may result in in-batch annotation bias.

We first study how the batch size can affect annotating behavior of crowds. Each time we take a comment as a *data item of interest*, then sample uniformly at random from the other comments of the same post, to construct two batches of data items, with sizes $k_1 < t_k$ and $k_2 \geq t_k$ respectively, where t_k is the batch size threshold which defines the treatment. The comment in the batch of size k_1 is *untreated*, and is put into the control group, while the comment in the batch of size k_2 is *treated*, and is placed into the test group.

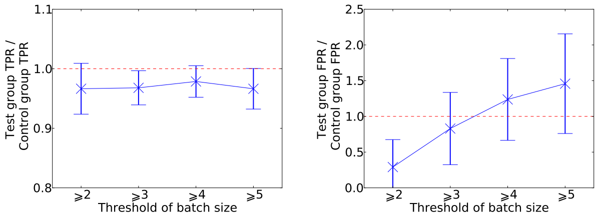
Another factor to be explored is the number of positives (*i.e.* inappropriate comments in our settings) in the batch. We take the same *data item of interest* as above then randomly sample $\pi_1 < t_\pi$ and $\pi_2 \geq t_\pi$ items from all the inappropriate comments in the same post, joined with $k - \pi_1 - 1$ and $k - \pi_2 - 1$ acceptable comments, also randomly sampled from the same post to form two batches in the control and test group respectively. Here k is fixed to be five, and $0 \leq t_\pi \leq k - 1$ is a given threshold.

We mix batches in both groups together and collect their annotations from CrowdFlower with the tasks structured to prevent the same annotators from labeling a particular data item multiple times. We compare the probability of each data item of interest being annotated as inappropriate in both the control and test groups. As we already know the ground truth of each comment, we separately study data items with different ground truth labels. Specifically, for a positive data item of interest, we compare its true positive rate in the control group TPR_1 and in the test group TPR_2 by calculating the ratio TPR_2/TPR_1; for a negative data item of interests we compare its false positive rate in control and test groups by FPR_2/FPR_1.

Observation results. In Figure 1 and 2 we plot the ratios of true positive and false positive rates TPR_2/TPR_1 and FPR_2/FPR_1 for each treatment with thresholds t_k and t_π respectively. In Figure 1(a), we do not observe significant [5] bias under treatments of different batch size thresholds, which means the true positive rate is not likely to be influenced by the batch size. On the other hand, in Figure 1(b), we see that presenting data items with other data items can reduce the false positive rate as compared to presenting only a single data item. This indicates that annotation bias exists when data items are organized into batches.

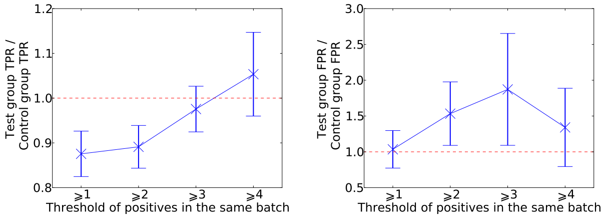
Moreover, Figure 2(a) shows TPR_2/TPR_1 is significantly less than one, when the treatment is set to "having one or more other positive data items in the same batch", which

[5] By "significant", we mean the ratio 1.0 is not within the 95% confidence interval for the observed results.



(a) TPR_2/TPR_1 (b) FPR_2/FPR_1

Figure 1: Comparing ratios of true positive rates (TPR) and false positive rates (FPR) when annotating with different thresholds t_k on the batch sizes.



(a) TPR_2/TPR_1 (b) FPR_2/FPR_1

Figure 2: Comparing ratios of true positive rates (TPR) and false positive rates (FPR) when annotating with different thresholds t_π on the number of positives.

means that positive data items are less likely to be identified correctly when there are other positive data items in the same batch. This conclusion is still significant when t_π is increased to two; however there is not sufficient evidence to support this when t_π is greater than two. On the other hand, Figure 2(b) shows that FPR_2/FPR_1 is significantly larger than one, when t_π is set to either two or three. This suggests that negative data items appearing along with a number of other positive data items can also influence crowd annotation. The observational results verify that annotation towards data items in the same batch can influence each other. In certain scenarios, the in-batch annotation bias can adversely impact annotation quality.

3. A FACTOR GRAPH MODEL FOR BATCH ANNOTATION

An intuitive explanation to the phenomenon illustrated in Figure 2 is that when an annotator works on a batch of data items, she may have a prior distribution of counts of different labels in a batch. That is, for a batch of comments, an annotator may be reluctant to mark many comments as inappropriate due to a prior belief that inappropriate comments are rare. Although the prior could be overcome when the labels are certain, this “inertial thinking” can overwhelm the annotation of difficult or uncertain data items. Based on this intuition, we introduce a factor graph model to describe the annotating distribution with in-batch annotation bias. We then use the model to characterize the bias in actual crowd-annotated data.

Model description. We represent a set of data items by d-dimensional feature vectors $X = \{x_i\}$, where $x_i \in \mathbb{R}^d$. The data items are organized into batches $B = \{b_j\}$ of size k, where each batch is a list of data items $b_j = [b_j^{(1)}, \cdots, b_j^{(k)}]$ with size k. Without loss of generality, we suppose each batch b_j receives one annotation represented by $y'_j = [y_j'^{(1)}, \cdots, y_j'^{(k)}]$, where $y_j'^{(i)} \in \mathcal{Y}$, and $\mathcal{Y}$ is the set of possible labels. If each batch can receive multiple annotations, we can produce multiple batch-annotation pairs with the same batch data associated with different annotations. The set of annotations is denoted as $Y' = \{y'_j\}$. Notice that we are modeling the annotation $y_j'^{(i)}$ of data item $b_j^{(i)}$, which is different from the ground truth label $y_{b_j^{(i)}}$.

In the graphical representation of the proposed batch annotation model shown in Figure 3, each annotation of a single data item $y_j'^{(i)}$ is represented by a random variable (clear circles), and the feature vector $x_{b_j^{(i)}}$ is represented by k observed variables (shaded circles). For simplicity, we only draw the model for a single batch b_j with a fixed size k, and assuming $b_j = [1, 2, \cdots, k]$.

In addition, there are k factor functions $\phi\left(x_i, y_j'^{(i)}\right)$, modeling the correlation between human annotation and data item features:

$$\phi(x_i, y_j'^{(i)}) = \exp\left[\alpha^\top f(x_i, y_j'^{(i)})\right] \tag{1}$$

where α is a vector of weighting parameters; $f(x_i, y_j'^{(i)})$ maps the feature vector according to $y_j'^{(i)}$, which in the binary classification case can simply be $f(x_i, y_j'^{(i)}) = y_j'^{(i)} x_i$.

To further consider the in-batch annotation bias, an additional factor function $\gamma\left(y'_j\right)$, defined as:

$$\gamma\left(y'_j\right) = \exp\left[\beta^\top g(y'_j)\right] \tag{2}$$

where in this model, we simply consider the count of different labels appearing in this annotation. Thus $g(y'_j)$ can be an indicator function, where each element corresponds to a possible distribution over counts of different labels within a size-k batch. For example, in a binary classification task, $g(y'_j)$ can be a $(k+1)$-dimensional vector, where the (k_1+1)-th element is 1 when $|y'_j| = k_1$, leaving all the other elements as 0. β is a vector of weighting parameters.

The likelihood function can be written as:

$$\begin{aligned}\mathcal{L} &= \prod_j \frac{1}{Z_j} \prod_{i=1}^{k} \phi(x_{b_j^{(i)}}, y_j'^{(i)}) \cdot \gamma\left(y'_j\right) \\ &= \prod_j \frac{1}{Z_j} \exp\left[\sum_{i=1}^{k} \alpha^\top f(x_{b_j^{(i)}}, y_j'^{(i)}) + \beta^\top g(y'_j)\right]\end{aligned} \tag{3}$$

where $Z_j = \sum_{y'} \exp\left[\sum_{i=1}^{k} \alpha^\top f(x_{b_j^{(i)}}, y'^{(i)}) + \beta^\top g(y')\right]$ is the normalizing factor for each annotation of a batch.

Model learning. We learn the model through gradient descent. By defining $\theta = \left[\alpha^\top \; \beta^\top\right]^\top$ and $h(b_j, y'_j) = \left[\sum_{i=1}^{k} f(x_{b_j^{(i)}}, y_j'^{(i)})^\top \; g(y'_j)^\top\right]^\top$, our learning objective is to find the best θ^* to optimize the log-likelihood.

The simplified expression of log-likelihood is:

$$\log \mathcal{L} = \sum_j \left[\theta^\top h(b_j, y'_j) - \log Z_j\right]$$

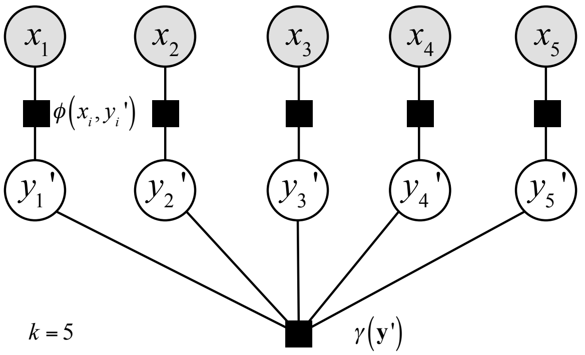


Figure 3: A graphical representation of the batch annotation model ($k = 5$).

And the gradient can be calculated by

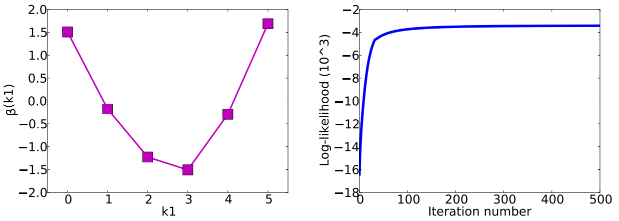
$$\begin{aligned}\frac{\partial \log \mathcal{L}}{\partial \theta^{(m)}} &= \sum_j \left[h^{(m)}(b_j, y'_j) - \sum_{y'} \frac{\exp\left(\theta^\top h(b_j, y')\right)}{Z_j} h^{(m)}(b_j, y') \right] \\ &= \sum_j \left[h^{(m)}(b_j, y'_j) - \mathbb{E}_{y'}\left[h^{(m)}(b_j, y') \right] \right]\end{aligned}$$

Calculation of the gradient is exponential to the batch size k. This isn't problematic, however, as at least for our tasks, k is small—typically $k \leq 5$. Thus, the computational cost is acceptable.

Parameter analysis. We learn the model based on annotations from crowds on the task of inappropriate comment identification. There are 6,121 annotations on 1,000 batches of size $k = 5$. Each batch of comments was constructed randomly without replacement. We encode the annotation by the label set $\mathcal{Y} = \{0, 1\}$, where 1 (positive) represents "inappropriate comment", and 0 (negative) represents "acceptable comments". Figure 4(b) presents the log-likelihood curve during the training process, which clearly shows that the model converges within 200 iterations.

Figure 4(a) shows the parameter β of the learned model, where $\beta(k_1)$ is the weight for annotation y' satisfying $||y'||_1 = k_1$ (*i.e.* the $(k_1 + 1)$-th element of vector β). The greater $\beta(k_1)$ is, the more likely annotators are to produce an annotation with k_1 positive labels among all the k data items. This suggests that annotators for this task tend to make polarized annotations, namely either none or all of the data items are positive. A possible reason is when an annotator already sees four of the data items in a batch are positive (negative), she may unjustly annotate the other data items also as positive (negative) without carefully checking the item itself.

Time complexity analysis. We provide a time complexity analysis of training the batch annotation model. Suppose the size of training data set is n. The dimension of parameter α is $d \times (|\mathcal{Y}| - 1)$, while the dimension of parameter β is $\binom{k+1}{|\mathcal{Y}|-1}$. For each batch-annotation pair, calculating the gradient requires $O(|\mathcal{Y}|^k)$ steps. Thus the time complexity for an iteration of model training is $O(n|\mathcal{Y}|^k + d|\mathcal{Y}| + \binom{k+1}{|\mathcal{Y}|-1})$. As both k and $|\mathcal{Y}|$ are small (effectively constant), the model training is computable, and will only increase linearly with regard to the feature number d.



(a) Parameter analysis (b) Convergence analysis

Figure 4: Factor graph model parameter and convergence results when trained to characterize in-batch bias in crowdsourced annotation.

4. ACTIVE LEARNING WITH IN-BATCH ANNOTATION BIAS

In this section, we propose an active learning algorithm taking advantage of in-batch annotation bias. For all the possible annotations on a subset of data items, we calculate the utility of each annotation weighted by its probability, where the utility is measured by the training likelihood on labeled data and the entropy of the unlabeled data based on the classifier updated by the subset. Then we optimistically choose the largest weighted utility as the quality function for this subset of data items. The active learning algorithm presented below is summarized in Algorithm 1.

Suppose the predictive probability of data item i is represented by $p(y|x_i; w_L)$, where w_L is the parameter for a classifier trained on data set L with labels y_L. Denote the annotating probability of a batch of data items as $q(y'_A|x)$, where y'_A contains the annotation of all the data items in A. An intuitive objective is to find a subset $A \subset U$ satisfying $|A| = k$, to optimize the expected utility of the classifier over the annotation distribution:

$$A^* = \arg\max_A \sum_{y'_A \in \mathcal{Y}^k} q(y'_A|x_A) F(L \cup A, y_L \cup y'_A, U \setminus A)$$

where $F(L \cup A, y_L \cup y'_A, U \setminus A)$ measures both the likelihood of labeled data and the uncertainty of all the unlabeled data based on the classifier trained on the labeled set extended by A and its annotation y'_A:

$$\begin{aligned}F(L \cup A, y_L \cup y'_A, U \setminus A) = \Bigg[&(1 - \lambda) \sum_{l \in L \cup A} \log p(y_l|x_l; w') \\ &+ \lambda \sum_{u \in U \setminus A} \sum_{y_u \in \mathcal{Y}} p(y_u|x_u; w') \log p(y_u|x_u; w') \Bigg]\end{aligned}$$

where λ is a given constant between 0 and 1, and w' is the classifier parameters trained on $L \cup A$, where A is annotated by y'_A.

By employing the factor graph model to estimate the annotating probability q, we have:

$$q(y'_A|x_A) = \frac{1}{Z_A} \exp\left[\sum_{a \in A} \alpha^\top f(x_a, y'_a) + \beta^\top g(y'_A) \right]$$

where Z_A is the normalizing factor.

However, directly optimizing this objective function is computationally difficult. Instead, we optimize the following objective:

$$\max_{A, y' \in \mathcal{Y}^k} q(y'|x_A) \left[F(L \cup A, y_L \cup y', U \setminus A) - F(L, y_L, U) \right]$$

which optimistically evaluates a subset A by the maximum product of the possible gain given a certain annotation y', and the probability that this annotation happens. A similar strategy is employed in [9]. Note that we are taking the increment of the F function by adding A instead of directly taking its value. Because given a subset A, if there exists an annotation y' that causes a drop of F function, then it should be penalized more as the probability of this annotation increases. In contrast, if annotating A with y' increases the F function, we favor y' as its probability increases.

Optimizing this objective function is still NP-hard as one has to enumerate all the possible subsets of U. Instead, we represent the chosen set A and their labels y'_A by a continuous matrix $S \in \mathbb{R}^{|U|\times|\mathcal{Y}|}$ where each row corresponds to an unlabeled data item in U, and a $|\mathcal{Y}|$-dimensional vector ν, where each element ν_y indicates the count of chosen data items that are annotated with label y. Without loss of generality, we assume all the unlabeled data items in U are indexed by $1, \ldots, |U|$. The following constraints must be satisfied for S and ν to be valid:

1. $0 \leq S \leq 1$;
2. $\forall y \in \mathcal{Y}, \sum_i S_{i,y} = 1$;
3. $\sum_y \nu_y = k$, so that the total size of chosen set is k;
4. $\forall i, \sum_y \nu_y S_{i,y} \leq 1$, to make sure each data item can only be chosen once.

Thereby the y-th column of S is a vector indicating which data item will be chosen and annotated as y, and $\sum_y \nu_y S_{i,y}$ for each i represents how likely i will be chosen in A. We denote the set of all the S that satisfying all the above constraints with regard to a given ν as Φ_ν.

By representing A and y'_A by S and ν, we further relax the q function with regard to S and ν by:

$$\hat{q}(S, \nu|X) = \frac{1}{Z_{S,\nu}} \exp\left[\sum_{i,y} S_{i,y}\nu_y \alpha^\top f(x_i, y) + \beta^\top \hat{g}(\nu)\right] \quad (4)$$

where the output of $\hat{g}(\nu)$ is equal to $g(y')$ for any y' with exactly the same count of different labels as ν indicates. And the normalizing factor can be estimated by:

$$Z_{S,\nu} = \sum_{\mu} \exp\left[\sum_i \sum_{y,\tilde{y}} S_{i,y}\mu_{\tilde{y},y}\alpha^\top f(x_i, \tilde{y}) + \beta^\top \hat{g}(\mu^\top \mathbf{1})\right] \quad (5)$$

where $\mathbf{1}$ is an all-1 vector with $|\mathcal{Y}|$ dimensions, and μ is a $|\mathcal{Y}|$-by-$|\mathcal{Y}|$ matrix, with each element as a non-negative integer, satisfying $\mu\mathbf{1} = \nu$. Although calculating the normalizing factor needs to enumerate all the possible μ, it is still computationally feasible as both the label space size $|\mathcal{Y}|$ and batch size k are usually small.

Similarly, we can relax the F function with regard to L and U. For simplicity, we omit the variable L, y_L, U as they do not change during our optimization:

$$\begin{aligned}\hat{F}(S, \nu) = \Big[&(1-\lambda)\sum_{l\in L} \log p(y_l|x_l; w') \\ &+ (1-\lambda)\sum_{i,y} S_{i,y}\nu_y \log p(y|x_i; w') \\ &+ \lambda \sum_{i,y} S_{i,y}(1-\nu^\top \mathbf{1}) p(y|x_i; w') \log p(y|x_i; w')\Big]\end{aligned} \quad (6)$$

```
Input: Initial labeled set $L$, unlabeled set $U$, batch size
       $k$, iteration number $N$, step length $\eta$, $\lambda$, Trained
       annotation model $\theta$
for $n \leftarrow 1, \ldots, N$ do
    Train classifier based on $L$;
    $\mathcal{O}^* \leftarrow -\infty$;
    // Enumerate $\nu$
    forall $\nu$ do
        Initialize $S$;
        // Optimize by $S$
        repeat
            Calculate $\nabla_S \mathcal{O}$ according to Eq. (8);
            $Z \leftarrow S + \eta \nabla_S O$;
            $S \leftarrow \arg\min_{S' \in \Phi_\nu} \frac{1}{2}||S' - Z||_F^2$;
            Update classifier $w'$ w.r.t. $S$, $\nu$;
        until converged;
        if $\mathcal{O}(S, \nu) > \mathcal{O}^*$ then
            $O^* \leftarrow \mathcal{O}(S, \nu)$;
            $S^* \leftarrow S$;
    Extract $A$ from $S^*$;
    Acquire labels of $A$ from crowds;
    Add $A$ and their labels into $L$ and remove $A$ from $U$;
```

Algorithm 1: Active learning with in-batch annotation bias.

Now we can optimize

$$\mathcal{O}(S, \nu) = \hat{q}(S, \nu|X)\left[\hat{F}(S, \nu) - F(L, y_L, U)\right] \quad (7)$$

where w' is the classifier parameter trained on the union of L and a weighted data set which assumes the unlabeled data are labeled as indicated by S and weighted by corresponding $\nu_y S_{i,y}$.

The proposed objective function can be optimized by enumerating all the possible ν, and optimizing S given ν, as both the batch size k and label space size $|\mathcal{Y}|$ is not large. Fixing ν, the optimization problem may be solved by gradient descent. It can still be challenging to calculate the gradient, as w' is a function of S. However, as our adjustment of S is subtle, we can assume w' to be fixed for the gradient, but update w' by retraining the model after S is adjusted. The gradient can be calculated as:

$$\begin{aligned}\frac{\partial \mathcal{O}(S, \nu)}{\partial S_{i,y}} &= \frac{\partial \hat{q}(S, \nu|X)}{\partial S_{i,y}}\left[\hat{F}(S, \nu) - F(L, U)\right] \\ &\quad + \hat{q}(S, \nu|X)\frac{\partial \hat{F}(S, \nu)}{\partial S_{i,y}}\end{aligned} \quad (8)$$

where $\partial\hat{q}/\partial S_{i,y}$ and $\partial\hat{F}/\partial S_{i,y}$ can be obtained from Equation (4), (5) and (6). In order to take care of all the constraints of S, since the feasible set is the intersection of multiple linear inequalities, we can project the updated (possibly not feasible) Z into the feasible set after each update, by finding the closest point S in the feasible set Φ_ν, which is a quadratic optimization problem minimizing the Frobenius norm $||S - Z||_F$ with constraints.

Once the solution S^* and ν^* that optimize the objective function above is obtained, we can choose set A accordingly. One way is to calculate the vector $\sigma = S^*\nu^*$ and take the top-k data items i with the top σ_i values in σ vector, as subset A. Another way is, for each $y \in \mathcal{Y}$, take the top-ν_y^* data items i with maximal $S_{i,y}$ values. We selected the second method as we find it more comfortable with our intuitions regarding the underlying processes.

Time complexity analysis. To be general, we assume the training complexity of the classifier is $O(\tau(|L|))$. At each iteration, we need to enumerate all the possibie ν, which is $\binom{k+1}{|\mathcal{Y}|^2-1}$. Optimizing S needs to calculate $\nabla_S \mathcal{O}$, which takes $O(|U||\mathcal{Y}| + |\mathcal{Y}|^k)$. Updating the classifier with regard to S takes $O(\tau(|L| + |\mathcal{Y}||U|))$. Assuming the total iterations to run before S converge is T. Extracting A from S^* is $O(|U|)$. Thus, at each iteration, the time complexity of selecting a batch of data items with size k, is $O\left(\tau(|L|) + \binom{k+1}{|\mathcal{Y}|^2-1} T\left(|U||\mathcal{Y}| + |\mathcal{Y}|^k + \tau(|L| + |\mathcal{Y}||U|)\right)\right)$. Notice that k and $|\mathcal{Y}|$ are very small in most cases. For example, in our inappropriate comment identification task, $|\mathcal{Y}| = 2$ and $k = 5$, both $k^{|\mathcal{Y}|}$ and $|\mathcal{Y}|^k$ is less than 10^2.

5. EXPERIMENTAL RESULTS

We test the effectiveness of our proposed active learning algorithm by conducting real experiments with a crowdsourcing service platform, on live LinkedIn data.

Data set. We sample a subset of 7,012 comments from a collection of posts with more than 5 comments, as the pool of data items for active learning. Among which, 30 comments are labeled by LinkedIn employees, serving as the initial labeled data set L for active learning, while the others (U) are left unlabeled. An additional subset of 1,715 comments are sampled from the ground truth data set labeled by LinkedIn employees (Cf. Section 2), where 1,372 of them become the test data set for evaluating the classifier performance and the other 343 comments form a validation data set.

Comparing method. We compare the performance of our proposed method with random selection and two of the existing algorithms:

- *Random (RND) Baseline.* Data items are picked at random. In particular, posts are selected uniformly at random without replacement. For each selected post, comments are again selected uniformly at random without replacement.

- *Maximum Uncertainty (MU).* Selects those data items for which the current classifier is most uncertain where uncertainty is measured by entropy and calculated as follows:

  $$A^* = \arg\max_{A \in U} \sum_{a \in A} \sum_{y \in \mathcal{Y}} -p(y|x_a, w_L) \log p(y|x_a, w_L)$$

  where $|A|$ is our batch size k.

- *Discriminative Active Learning (DA).* This method uses a similar objective function as the proposed method (WDA) but without weighting F with annotating probability q. It is equivalent to the discriminative active learning algorithm proposed in [9].

* *Weighted Discriminative Active Learning (WDA).* This is our proposed method discussed in Section 4.

Experimental setup. For each of the above algorithms, we run for 20 iterations, starting from the same initial training set L consisting of 30 annotated data items drawn from the expert labeled ground truth. The base classifier used is logistic regression with L2 regularization as implemented in Weka [6]. In our task we only display comments (data items) on a common post in a particular batch. This is implemented for the above algorithms as follows: at each iteration for each algorithm we restrict the chosen set A_p to be a subset of comments on some post p. This is repeated for every post with more than k (set to 5 for all experiments) comments left in the unlabeled data set U. After all the batches (A_p) are generated, they are ranked by the active learning objective function and the top 50 batches are sent to the crowdsourcing platform for annotation. After the data items are annotated, they are added into the training data set and removed from the unlabeled data set. The classifier is then retrained on the new training data set.

We utilize CrowdFlower as our crowdsourcing platform. As discussed in Section 2, the codebook definitions for inappropriate comments and examples of both inappropriate and acceptable comments is presented to the annotators at each iteration. Each annotator is first evaluated with ten "test questions", *i.e.* batches of data items with known annotation, for the crowdsourcing system to assess their "trust". Only annotators who can correctly answer at least 70% of the test questions can proceed to the actual annotation task. Each batch of data items must be labeled by at least five annotators before it is returned from CrowdFlower. We determine the final label for each data item with majority voting of the returned annotations.

Evaluation. We evaluate the performance of each classifier at each iteration with the following metrics calculated against the test set:

- *Area under ROC (AUC).*

- *$F_{0.5}$-score ($F_{0.5}$).* Calculated with $F_{0.5} = \frac{(1+0.5^2) \times precision \times recall}{0.5^2 \times precision + recall}$ as we value precision more than recall.

- *Optimized F_1-score (F_1^*).* As the data distribution can be extremely imbalanced in our task, we optimize each classifier's threshold in terms of F_1-score with the validation set, instead of using the default of 0.5. We then test the classifier's F_1 performance with that threshold.

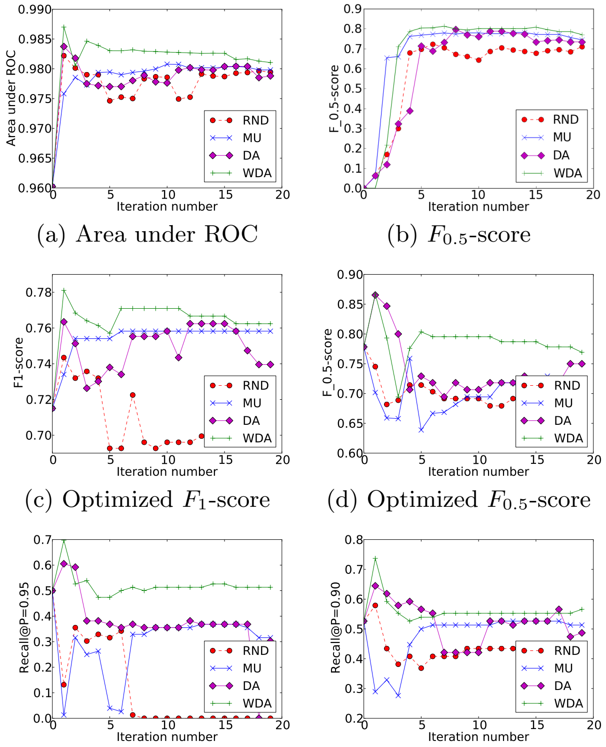
- *Optimized $F_{0.5}$-score ($F_{0.5}^*$).* Similarly, we can optimize the $F_{0.5}$-score based on the validation set and evaluate on the test data set.

- *Recall @ Precision (R@P).* We find the recall at a minimum precision by adjusting the classifier threshold and reporting the highest recall with precision greater than a certain value. In these experiments, we use Recall@0.95 and Recall@0.90.

We also evaluate each algorithm by its average classifier performance across all iterations. For optimized F-score, the classifier threshold is recalculated at each iteration.

Performance comparison. Table 2 shows the average classifier performance for each of our evaluated algorithms. All of the active learning algorithms outperform the random baseline. Among all the active learning algorithms, WDA outperforms the other baseline methods for both optimized F-measures as well as recall at a fixed precision. It also achieves a performance close to maximum uncertainty in terms of $F_{0.5}$-score, without optimizing the threshold. Especially in terms of R@.95, WDA achieves an

[6] http://www.cs.waikato.ac.nz/~ml/weka/



(a) Area under ROC

(b) $F_{0.5}$-score

(c) Optimized F_1-score

(d) Optimized $F_{0.5}$-score

(e) Recall@Precision=0.95

(f) Recall@Precision=0.90

Figure 5: Comparing learning curves of different active learning algorithms.

Table 2: Performance of active learning algorithms. All results are shown as percents.

Method	AUC	$F_{0.5}$	F_1^*	$F_{0.5}^*$	R@.95	R@.90
RND	97.73	57.74	70.75	69.95	11.45	43.49
MU	97.86	**68.58**	75.40	69.83	29.74	48.16
DA	97.82	60.84	74.80	73.98	37.50	52.17
WDA	**98.17**	68.39	**76.45**	**78.69**	**51.97**	**56.05**

average performance of 51.97%, while maximum uncertainty does not reach 30%. Most interestingly, it outperforms DA on all measures giving further evidence that our factor graph model is indeed learning and leveraging the in-batch annotation bias. Learning curves of different measures are also shown in Figure 5. After iteration seven, RND's performance on F_1 and R@.95 plummets. We suspect this is due to noisy annotation. Simultaneously, the recall for WDA at 95% precision converges to a stable value much higher than the other methods.

Case study. To better understand what is happening with the annotations, we performed a case study comparing the WDA and non-WDA annotations. We compare several pairs of batches constructed by WDA and other baselines having at least one comment in common but annotated differently. Some examples are shown in Table 3. In the first pair, WDA selected five comments predicted to be identically annotated as inappropriate while RND, picking at random, chose a mix with three actually inappropriate comments (the last three). The comment in common is correctly annotated as inappropriate in the WDA constructed batch but not in the randomly constructed batch. The third comment in the random batch is actually also inappropriate according to our codebook yet was mislabeled by a majority of the crowd annotators. WDA tends to construct batches where the biased annotation probability and the classifier predicted probability are relatively coherent, which in turn helps crowds perform more correct annotation.

The second example compares WDA to DA. The fifth comment in the WDA constructed batch is actually a blatant solicitation. It starts with "I am Madame Clarisse..." but ends with advertising loan services. For DA, on the other hand, this same comment is not identified as inappropriate. We notice that there is another inappropriate comments in the same batch, which is correctly identified. As we observed in Figure 4(a), annotators are more reluctant to label two inappropriate comments than only one in a batch.

6. RELATED WORK

Annotation bias. A number of studies have been conducted on evaluating data quality collected from crowds, and modeling annotator behaviors to optimize the data quality. Snow *et al.* [21] explored the performance with non-expert annotators in several NLP tasks. Raykar *et al.* [14, 15, 16] studied how to learn a model with noisy labeling. Specifically, they employ a logistic regression classifier, and insert hidden variables indicating whether an annotator tells the truth. Whitehill *et al.* [28] modeled the annotator ability, data item difficulty, and inferred the true label from the crowds in a unified model. Venanzi *et al.* [23] proposed a community-based label aggregation model to identify different types of workers, and correct their labels correspondingly. However, they do not study the case when data items are grouped into batches. Das *et al.* [5] addressed the interactions of opinions between people connected by networks. Demeester *et al.* [6] discussed the disagreement between different users on assessment of web search results. Their studies also focus on understanding annotator behavior, but none of them consider the case when multiple data items are organized into a batch.

Scholer *et al.* [17, 18] studied the annotation disagreements in a relevance assessment data set. They discovered the correlation between annotations of similar data items. They also explored "threshold priming" in annotation, where the annotators tend to make similar judgments or apply similar standard on consecutive data items they review. However, their studies focus more on qualitative conclusions, without a quantitative model to characterize and measure the discovered factors. Carterette *et al.* [2] provided several assessor models for the TREC data set. Mozer *et al.* [13] studied the similar "relativity of judgments" phenomenon on sequential tasks instead of small batches. Also, they focused more on debiasing not active learning.

Batch active learning. Active learning has been extensively studied. Settles *et al.* [19] summarized a number of active learning strategies. Batch active learning, in contrast to traditional active learning settings, aims to choose a set of data items to query, which proposes some unique challenges. Some strategies focus on optimizing the helpfulness of a data batch. Hoi *et al.* [11, 10] utilized Fisher information matrix to choose the data items that are likely to change the classifier parameters most. Brinker *et al.* [1] studied batch active learning for SVM and aimed to max-

Table 3: Case study results. The left column shows the batches constructed by WDA method, while the right column shows the batches constructed by different baselines. y_i' gives the crowdsourced annotation with "Yes" marking inappropriate comments and "No" for acceptable comments. Comments whose label is bolded are common to both batches.

WDA		RND	
y_i'	Comment content	y_i'	Comment content
Yes	please check my profile. harvindersinghlongowal@yahoo.com	No	when fresher will get experience until they don't give opportunity.
Yes	Dear sir, I am interested, please check up my profile	No	only experience no fresher .... very badddddddddddddddd
Yes	Dear sir I interest your job. please check my details.	No	review my stats and kindly notify for further proceedings
Yes	Interested. Please see my profile.	Yes	please check my profile, mobile number is. 09559186335
Yes	Hello Sir, Please check my profile. I am ready to work in any part of world. Thank you,	**No**	Hello Sir, Please check my profile. I am ready to work in any part of world. Thank you,

WDA		DA	
y_i'	Comment content	y_i'	Comment content
No	What we need*	No	Exactly, what I needed to read today. It really is almost like scary when you get a gift like this read. ... *[Omitted]*
No	Thank You... sounds familiar...Seen 3 recession...been last person in company (28 years old) before company bancrupty, giving 260 people notice before that, part of the job was to sold the assets, basically buried the company... Never get fired after that...:-) But i can relax now only in different atmosphere...vacation..summer house without phone...:-)	Yes	Good day, My Name is Peter Williams, am a supplier We suppliers all kind of metals and plastic and scrap, If you are interested in any product kindly send me email and the materials you want so i could send you our quotation okay. If you are interested in any of materials, Kindly revert back to me via "infocuixin.sl@gmail.com" Thanks
No	That was the message I needed. I will not dwell...I love my job!	No	Thanks for the article, Sallie. I like your secret #3 above-responsibility to those who have helped you before. ... *[Omitted]*
No	I don't relax well either, why do that when I could go for a walk?	No	Yes, we do!! Please tell her to check us out at EllevateNetwork.com
Yes	I am Madame Clarisse Sweet, ... *[Omitted]* I'd like to know if you have found an investor to assist you in realizing your projects otherwise the jeans Jeanne couple can help you if this is the couple who helped me find my loan of 75,000 € It I can help you as you who is in need he can be reached at the following email: naillejvalane@gmail.com	**No**	I am Madame Clarisse Sweet, ... *[Omitted]* I'd like to know if you have found an investor to assist you in realizing your projects otherwise the jeans Jeanne couple can help you if this is the couple who helped me find my loan of 75,000 € It I can help you as you who is in need he can be reached at the following email: naillejvalane@gmail.com

imize the diversity within the selected set of data samples. Guo *et al.* [9] proposed discriminative active learning strategy by formulating the problem as an optimization problem. A number of strategies also aim at choosing the most representative data batch with regard to the unlabeled data set. Yu *et al.* [30] proposed a transductive experimental design, which prioritizes the data samples that represent the hard-to-predict data. Chattopadhyay et al. [3] tried to choose the data batch to minimize Maximum Mean Discrepancy (MMD) to measure the difference in distribution between the labeled and unlabeled data. There are studies addressing both intuitions. Wang *et al.* [25] designed a framework to minimize the upper bound of empirical risk, which aims to find a batch of data items that are both discriminative and representative. However, all of the above studies assume reliable oracles—which never are in multiple-annotator scenarios such as crowdsourcing.

Active learning with crowds. Crowdsourcing serves as a potentially ideal oracle for active learning. Two perspectives have been explored; first is how to select data items to query when the oracles are noisy. Sheng *et al.* [20] provided an empirical study on the performance of repeated labeling and developed an active learning strategy addressing both the "label uncertainty" and "model uncertainty". Second is how to select annotators for crowdsourcing. Donmez *et al.* [7] studied this problem, by modeling the querying problem as a multi-armed bandit problem. Each annotator is regarded as as a bandit, and a binary reward function is defined based on whether the oracle provides a correct label. Yan *et al.* [29] explore both the problem of selecting query samples and selecting oracles, in context of a logistic regression classifier. Kajino *et al.* [12] proposed a convex optimization function for active learning in crowds. However, none of them leverages in-batch bias for active learning.

7. CONCLUSION

In this paper, we study the in-batch data annotation bias from crowds, which occurs when data items are organized into small batches. Our experimental results on a crowdsourcing platform suggest that the annotation of data items is impacted by other data items presented in the same batch. We propose a factor graph based batch annotation model, to capture the in-batch annotation bias of crowds. Based on the annotation model, we propose a novel active learning algorithm, which takes advantage of our batch annotation model to construct batches of unlabeled data items that reduce annotation error and improve classifier performance.

It may not be the case that the particular bias we found in our task generalizes to other tasks—this requires more experimentation. Regardless, it is reasonable to expect *some* form of in-batch bias. Our factor graph model can be trained on a preliminary annotation to detect whatever in-batch bias is present and then our proposed active learning algorithm can leverage that model to improve performance.

The proposed annotation model has broad application as strict independence between annotations of different data items can not be achieved in many annotation tasks. Without much training, crowd annotators are more likely to be steered by in-batch bias into making unnecessary mistakes. Although the batch annotation model is learned from the annotations of inappropriate comments in online social networks, we argue that the conclusion drawn from the model is not task-specific and should be applicable to most crowdsourcing labeling tasks. For example, it would be interesting to use the model to explore in-batch bias in conference paper reviewing. This paper has focused on one annotation task. A next step is to evaluate the proposed algorithms on other tasks. It is also interesting to evaluate if the model parameters can be directly transferred between tasks without retraining.

8. REFERENCES

[1] K. Brinker. Incorporating diversity in active learning with support vector machines. In *ICML*, volume 3, pages 59–66, 2003.

[2] B. Carterette and I. Soboroff. The effect of assessor error on ir system evaluation. In *SIGIR*, pages 539–546. ACM, 2010.

[3] R. Chattopadhyay, Z. Wang, W. Fan, I. Davidson, S. Panchanathan, and J. Ye. Batch mode active sampling based on marginal probability distribution matching. In *KDD*, pages 741–749. ACM, 2012.

[4] T. Cohn and L. Specia. Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In *ACL*, pages 32–42, 2013.

[5] A. Das, S. Gollapudi, R. Panigrahy, and M. Salek. Debiasing social wisdom. In *KDD*, pages 500–508. ACM, 2013.

[6] T. Demeester, R. Aly, D. Hiemstra, D. Nguyen, D. Trieschnigg, and C. Develder. Exploiting user disagreement for web search evaluation: an experimental approach. In *WSDM*, pages 33–42. ACM, 2014.

[7] P. Donmez, J. G. Carbonell, and J. Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD*, pages 259–268. ACM, 2009.

[8] R. G. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. In *NIPS*, pages 558–566, 2011.

[9] Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In *NIPS*, pages 593–600, 2008.

[10] S. C. Hoi, R. Jin, and M. R. Lyu. Large-scale text categorization by batch mode active learning. In *WWW*, pages 633–642, 2006.

[11] S. C. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, pages 417–424. ACM, 2006.

[12] H. Kajino, Y. Tsuboi, and H. Kashima. A convex formulation for learning from crowds. In *AAAI*, 2012.

[13] M. C. Mozer, H. Pashler, M. Wilder, R. V. Lindsey, M. C. Jones, and M. N. Jones. Decontaminating human judgments by removing sequential dependencies. *NIPS*, 23, 2010.

[14] V. C. Raykar and S. Yu. Ranking annotators for crowdsourced labeling tasks. In *NIPS*, pages 1809–1817, 2011.

[15] V. C. Raykar, S. Yu, L. H. Zhao, A. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *ICML*, pages 889–896. ACM, 2009.

[16] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *JMLR*, 11:1297–1322, 2010.

[17] F. Scholer, D. Kelly, W.-C. Wu, H. S. Lee, and W. Webber. The effect of threshold priming and need for cognition on relevance calibration and assessment. In *SIGIR*, pages 623–632. ACM, 2013.

[18] F. Scholer, A. Turpin, and M. Sanderson. Quantifying test collection quality based on the consistency of relevance judgements. In *SIGIR*, pages 1063–1072. ACM, 2011.

[19] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52:55–66, 2010.

[20] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD*, pages 614–622. ACM, 2008.

[21] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP*, pages 254–263, 2008.

[22] H. Su, J. Deng, and L. Fei-Fei. Crowdsourcing annotations for visual object detection. In *Human Computation Workshops at AAAI*, 2012.

[23] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *WWW*, pages 155–164, 2014.

[24] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. In *CVPR*, pages 1449–1456. IEEE, 2011.

[25] Z. Wang and J. Ye. Querying discriminative and representative samples for batch mode active learning. In *KDD*, pages 158–166. ACM, 2013.

[26] F. L. Wauthier and M. I. Jordan. Bayesian bias mitigation for crowdsourcing. In *NIPS*, pages 1800–1808, 2011.

[27] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. The multidimensional wisdom of crowds. In *NIPS*, pages 2424–2432, 2010.

[28] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043, 2009.

[29] Y. Yan, G. M. Fung, R. Rosales, and J. G. Dy. Active learning from crowds. In *ICML*, pages 1161–1168, 2011.

[30] K. Yu, J. Bi, and V. Tresp. Active learning via transductive experimental design. In *ICML*, pages 1081–1088. ACM, 2006.