Language Understanding in the Wild: Combining Crowdsourcing and Machine Learning

Edwin Simpson
University of Oxford, UK
edwin@robots.ox.ac.uk

Matteo Venanzi
University of Southampton, UK
mv1g10@ecs.soton.ac.uk

Steven Reece
University of Oxford, UK
reece@robots.ox.ac.uk

Pushmeet Kohli
Microsoft Research,
Cambridge, UK
pkohli@microsoft.com

John Guiver
Microsoft Research,
Cambridge, UK
joguiver@microsoft.com

Stephen J. Roberts
University of Oxford, UK
sjrob@robots.ox.ac.uk

Nicholas R. Jennings
University of Southampton, UK
nrj@ecs.soton.ac.uk

ABSTRACT

Social media has led to the democratisation of opinion sharing. A wealth of information about public opinions, current events, and authors' insights into specific topics can be gained by understanding the text written by users. However, there is a wide variation in the language used by different authors in different contexts on the web. This diversity in language makes interpretation an extremely challenging task. Crowdsourcing presents an opportunity to interpret the sentiment, or topic, of free-text. However, the subjectivity and bias of human interpreters raise challenges in inferring the semantics expressed by the text. To overcome this problem, we present a novel Bayesian approach to language understanding that relies on aggregated crowdsourced judgements. Our model encodes the relationships between labels and text features in documents, such as tweets, web articles, and blog posts, accounting for the varying reliability of human labellers. It allows inference of annotations that scales to arbitrarily large pools of documents. Our evaluation using two challenging crowdsourcing datasets shows that by efficiently exploiting language models learnt from aggregated crowdsourced labels, we can provide up to 25% improved classifications when only a small portion, less than 4% of documents has been labelled. Compared to the six state-of-the-art methods, we reduce by up to 67% the number of crowd responses required to achieve comparable accuracy. Our method was a joint winner of the CrowdFlower - CrowdScale 2013 Shared Task challenge at the conference on Human Computation and Crowdsourcing (HCOMP 2013).

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741689.

General Terms

Crowdsourcing, machine learning, variational Bayes, classifier combination, text classification, sentiment analysis, human computation

1. INTRODUCTION

Social media provides an increasingly rich source of information about public opinion and current events, which can be valuable to professionals across a wide range of industries. For example, Twitter[1] can reflect the public's sentiment about the weather, such as in the data collected during the CrowdScale 2013 Shared Task challenge[2], opinion of major health emergencies such as the H1N1 flu pandemic [6], or knowledge of disaster events such as Typhoon Haiyan [5]. Mining this large body of unstructured data requires an understanding of the language used in each specific context. For example, the *sentiment* of a document, which reflects the author's attitudes or opinion of a subject, is captured in the language they use. However, that relationship between sentiment and language typically depends on factors such as the viewpoint and the gender of the authors and the context of their writing. For example, distinctive terms such as "love" and "dude" are more frequently used by female and male Twitter users, respectively, to refer to the same concept of a friend or a family member [15]. Similarly, reports posted by members of the public to Ushahidi after the 2010 Haiti earthquake used a type of language that is significantly different to that seen in other locations and other types of emergency [22]. This diversity in social media text inhibits the performance of any generic method for automated document classification "in the wild". However, this problem can be alleviated by human interpreters who can use their background knowledge and natural language understanding skills to recognise the sentiment of documents and adapt to the diverse language used in different contexts.

Interpreting sentiment or relevance of a piece of text is highly subjective and, along with variations in annotators' skill levels, it can result in disagreement. To overcome this

[1] www.twitter.com
[2] www.crowdscale.org

problem, existing methods for crowdsourced document classification require labels from multiple annotators for every document in the corpus [28, 26], which can be prohibitively costly or time consuming [22]. Fortunately, the occurrence of certain terms in each document also provides weak indications of the sentiment of a document, which can be used to reduce the cost of employing human interpreters to annotate the entire corpus. Therefore, we propose a hybrid approach to large scale document classification that integrates human intelligence with automated analysis of text.

In this paper we present *Bayesian Classifier Combination with Words (BCCWords)*, a framework for combining annotations from a crowd of workers with text features to classify a corpus of documents. This approach is an example of an emerging research area known as *human-agent collectives* [12]. We introduce a scalable Bayesian inference mechanism for BCCWords, which learns posterior distributions over the workers' reliability and document classifications, given the documents' text features and a set of crowdsourced annotations. Our method not only allows us to handle the varying error rates and bias of individual members of the crowd, but also allows us to annotate an entire set of documents when only a subset have been labelled by the crowd, by leveraging the inferred language model to automatically annotate the remaining documents.

In more detail, we make the following contributions to the state-of-the-art:

1. We present a novel generic model, BCCWords, that combines human and computer interpretations of free text documents and infers their sentiment.

2. We present a novel scalable variational Bayes inference algorithm, BCCWords–VB, for training the BCCWords model. This algorithm was first demonstrated at the CrowdScale 2013 Shared Task Challenge and was a joint winner.

3. We derive an efficient inference decomposition method that allows our algorithm to perform batch inference over hundreds of thousands of documents and demonstrate inference with $569,786$ crowdsourced sentiment judgements for $98,979$ documents in approximately 20 minutes on a standard laptop.

4. We present an exhaustive evaluation of our algorithm on two real datasets of text annotations and compare it against six state-of-the-art methods for crowd-based text classification and data aggregation. Specifically, our evaluation shows that our algorithm is up to 25% more accurate when only a small portion, less than 4% of the documents, have been labelled and that our algorithm reduces by up to 67% the amount of crowd labels required to achieve comparable accuracy with standard methods.

The paper is structured as follows. We review the literature on language modelling and aggregation models for crowdsourced judgements in Section 2. Section 3 presents our model in detail, then Section 4 provides mathematical details for our variational inference algorithm. Section 5 demonstrates the efficacy of our approach by comparing it against state-of-the-art benchmarks on two real world crowdsourcing datasets. Finally, we conclude and discuss future work in Section 6.

2. AGGREGATING JUDGEMENTS

Many applications in the literature have employed crowdsourcing, whereby multiple people process each document or data point [13, 1, 33]. A key challenge in such crowdsourcing applications is to mitigate the bias of subjective labellers. Previous work has addressed this problem by combining crowd responses to obtain reliable aggregate classifications. However, as yet, these methods have not exploited the language used in the text to further assist in interpreting the text. We propose to use the variations in language associated with sentiment to reduce errors and bias arise when employing members of the public to perform labelling tasks.

A further challenge with real world applications of document crowdsourcing is the cost of employing a sufficient number of annotators to rapidly label a large dataset. For example, the Ushahidi dataset comprises at least $40,000$ text messages which had to be interpreted in the first month after the earthquake in Haiti, which proved to be infeasible [22]. However, a suitable language model would enable automated analysis at much greater scale and allows the annotators to focus their efforts on the most difficult documents. We therefore propose a learning method for harnessing the skills of human labellers to learn a bespoke language model from much larger sets of documents.

A number of methods have been used in the literature to address the challenge of aggregating annotations from the crowd, including the simple technique of *majority voting* [18]. However, simple majority voting treats all annotators as equally reliable and does not provide any meaningful measure of confidence in the combined decision to account for conflicts in judgement or low annotator skill levels. To overcome this problem, probabilistic methods have been developed which learn the skill levels or bias of each annotator and aggregate their decisions accordingly [26, 7, 32, 25, 31]. These methods are prone to error when only small amounts of gold labels are available as they do not consider uncertainty in skill levels and other model parameters. For example, when only one label is obtained from a worker, these methods may infer that the worker is either perfectly reliable or totally incompetent when, in reality, the worker is neither. This is a common problem with approaches to inference that use maximum likelihood or maximum a-posteriori solutions [4]. In order to overcome this limitation, algorithms for aggregating crowdsourced data including *SFilter* [8] and *Bayesian Classifier Combination (BCC)* [14, 30] capture the uncertainty in the workers' skill levels or bias, as well as the uncertainty in the aggregated labels. Unfortunately, these methods do not exploit the text features of documents, and consequently require each document to be labelled by the crowd, often multiple times, to obtain confident classifications.

Previous work has introduced methods for automatic text classification based solely on word content, such as the bag-of-words classifier [11]. Although such methods have been applied to automated sentiment analysis, they need a language model for each application context [21]. This often requires large amounts of training data and substantial effort by the system designer to cope with the diversity in language [17]. In contrast our approach uses a crowd of human annotators to learn a language model rapidly and cheaply. In the following sections we develop the BCCWords model and then demonstrate its efficacy with benchmark methods.

3. THE BCCWORDS MODEL

In this section we describe our novel BCCWords model. This model is an extension of the independent Bayesian classifier combination (IBCC) model presented in [28] which classified data points using only crowdsourced labels. BCCWords models the crowd as multiple heterogeneous classifiers, and uses both the crowd's responses and the word structure of documents to classify them. An advantage of BCCWords is that it can be inferred in a *semi-supervised* or *unsupervised* manner. It does not require separate training and test phases but uses a single, combined learning phase over all available data. The semi-supervised approach simultaneously learns from labelled training data and the latent structure in the entire dataset, making it particularly suitable when gold-standard data is limited.

We start by introducing our notation. There is a crowd of K annotators expressing their judgement about the correct classification of N documents over a range of C possible classes. The classes may represent sentiment classes, topic labels, or other types of annotation. Each document, i, has an unknown true class $t_i \in C$. The judgement of annotator k for document i is denoted as label $l_i^{(k)}$, where $l_i^{(k)} \in C$. Also, we assume that the nth word, $w_{i,n}$, of document i takes a value d from a dictionary of size D words. For notational simplicity, we assume a dense set of judgements in which each annotator rates all N documents. However, as will become clear in section 4.1, our model naturally supports sparsity in the dataset, which is the case for the CrowdFlower dataset used in Section 5.

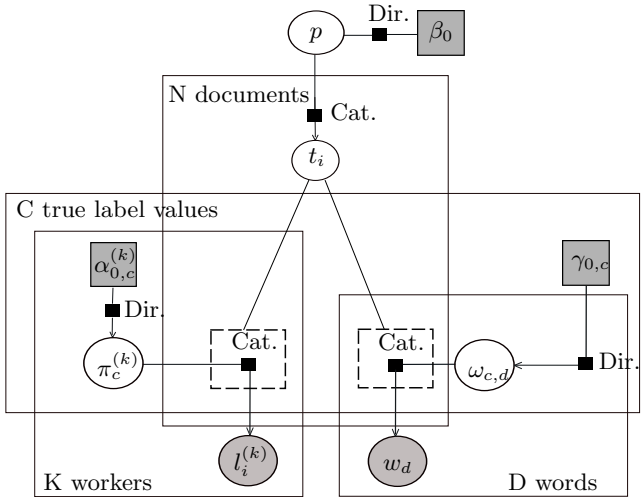


Figure 1: The factor graph of BCCWords. The circular, shaded nodes represent observed variables and the square, shaded nodes represent the hyperparameters. The plates describe (i) the set of K annotators, (ii) the N documents, (iii) the C possible true label values and (iv) the D words contained in the dictionary of terms used in the documents.

The factor graph of our Bayesian Combination Model, BCCWords, is shown in Figure 1, and the model is described as follows. We assume that each annotator draws judgements for documents of class $t_i = c$ from a categorical distribution with parameters $\boldsymbol{\pi}_c^{(k)}$:

$$l_i^{(k)} | \boldsymbol{\pi}_c^{(k)}, t_i \sim \text{Cat}(\boldsymbol{\pi}_c^{(k)})$$

where $\boldsymbol{\pi}_c^{(k)}$ is the accuracy vector of annotator k for documents of class c. That is, each element of $\boldsymbol{\pi}_c^{(k)}$ specifies the probability that annotator k will give a judgement c' when presented with a document whose true class is c:

$$\pi_{c,c'}^{(k)} = p(l_i^{(k)} = c' | t_i = c)$$

where p is the Dirichlet distribution. The set of accuracy vectors $\boldsymbol{\pi}_c^{(k)}$ for all c is called the *confusion matrix* representing k's reliability. In Figure 1, the annotator confusion matrices are shown in the left-hand plate for all K annotators, depicting how the response of an annotator depends on the true class t_i of the document they are judging. The use of confusion matrices allows our model to combine annotators of very different skill levels, and is able to handle those who make random guesses or whose responses are the opposite of what we expect. Furthermore, a confusion matrix accounts for the personal bias of an annotator, since a tendency to select a more positive or negative judgment, c', than other members of the crowd, when presented with documents of true class c, will result in an increased likelihood, $\pi_{c,c'}^{(k)}$. A personal bias toward selecting judgement c' for all documents will result in high likelihoods $\pi_{c,c'}^{(k)}$ for all true classes c, thus the model will lean that the label c' from annotator k is less strongly discriminative.

Our language model is defined as follows. Given a document i of class c we assume that the probability that the nth word is d (i.e. $w_{i,n} = d$) follows a categorical distribution with parameters $\boldsymbol{\omega}_c = \{\omega_{c,d} \forall d\}$:

$$w_{i,n} | \omega_{c,d}, t_i \sim \text{Cat}(\boldsymbol{\omega}_c),$$

where $\omega_{c,d} = p(w_{i,n} = d | t_i = c)$ is the probability that a randomly-drawn word from a document of class c is the word d. This probabilistic representation of text in documents corresponds to a mixture of bag-of-words model [11], where each mixture component is a bag-of-words model associated with one particular object class. The word distributions are represented in Figure 1 in the right-hand plate, showing the variables corresponding to the D words in the dictionary.

We assume that the true class label for each document, t_i, is drawn from a categorical distribution with parameters $\boldsymbol{\rho}$: $t_i | \boldsymbol{\rho} \sim \text{Cat}(\boldsymbol{\rho})$. The parameters $\boldsymbol{\rho}$ can be regarded as the proportion of documents in each class, so that $\rho_c = p(t_i = c | \boldsymbol{\rho})$. These parameters are shown at the top of Figure 1.

To model the uncertainty in the latent variables in our model, we assign conjugate Dirichlet prior distributions to $\boldsymbol{\pi}_c^{(k)}$, $\boldsymbol{\omega}_c$ and $\boldsymbol{\rho}$, for each class $c \in C$ and annotator $k \in K$:

$$\begin{aligned} \boldsymbol{\pi}_c^{(k)} | \boldsymbol{\alpha}_{0,c}^{(k)} &\sim \text{Dir}(\boldsymbol{\alpha}_{0,c}^{(k)}) \\ \boldsymbol{\rho} | \boldsymbol{\beta}_0 &\sim \text{Dir}(\boldsymbol{\beta}_0) \\ \boldsymbol{\omega}_c | \boldsymbol{\gamma}_{0,c} &\sim \text{Dir}(\boldsymbol{\gamma}_{0,c}), \end{aligned}$$

where $\boldsymbol{\alpha}_{0,c}^{(k)}, \forall c$ is the per-annotator confusion matrix hyperparameter, and $\boldsymbol{\gamma}_{0,c}$ is the hyperparameter for the bag-of-words distribution for each class c. These hyperparameters have intuitive interpretations as prior *pseudo-counts*, meaning that their values are equivalent to a number of prior observations, which represent the strength of prior beliefs. When implementing BCCWords, the diagonal values of hyperparameters $\boldsymbol{\alpha}_{0,c}$ of the confusion matrices can be set to higher values than the off-diagonals, encoding the prior belief that annotators are expected to be better than random. The hyperparameters for the word distributions, $\boldsymbol{\gamma}_{0,c}$, and

the class proportions, $\boldsymbol{\beta}_0$, can both be set so that the priors are uniform. This reflects an initial lack of information in the word structure of the documents and the class distribution of the documents.

To enable us to perform Bayesian inference over our model, we first specify the complete joint distribution:

$$p\left(\boldsymbol{l}, \boldsymbol{t}, \boldsymbol{\pi}_1^{(1)},.., \boldsymbol{\pi}_C^{(K)}, \boldsymbol{\omega}_1,.., \boldsymbol{\omega}_C, \boldsymbol{\rho} | \boldsymbol{\alpha}_{0,1}^{(1)},.., \boldsymbol{\alpha}_{0,C}^{(K)}, \boldsymbol{\beta}_0, \boldsymbol{\gamma}_{0,1},.., \boldsymbol{\gamma}_{0,C}\right)$$
$$= \prod_{i=1}^{N} \left\{ \rho_{t_i} \prod_{k=1}^{K} \pi^{(k)}_{t_i, l_i^{(k)}} \cdot \prod_{n=1}^{W_i} \omega_{t_i, w_{i,n}} \right\} p(\boldsymbol{\rho}|\boldsymbol{\beta}_0)$$
$$\prod_{c=1}^{C} \left\{ p(\boldsymbol{\omega}_c|\boldsymbol{\gamma}_{0,c}) \prod_{k=1}^{K} p(\boldsymbol{\pi}_c^{(k)}|\boldsymbol{\alpha}_{0,c}^{(k)}) \right\} \quad (1)$$

where $\boldsymbol{l}$ is the set of labels from all annotators for all documents, and W_i is the number of words in document i. The next section describes a method that uses this joint distribution to estimate the posterior distribution over the unknown variables in our model.

4. EFFICIENT VARIATIONAL INFERENCE

The BCCWords model presented in the previous section is tuned, or *inferred*, by learning the parameters of the posterior distribution over its unknown variables, so that the model fits the data. In this section we describe an efficient method for inference using *variational Bayes* (VB) [4]. The next subsection presents details of the VB algorithm for BCCWords. Then, Section 4.2 describes how this BCCWords-VB algorithm can be extended to batch processing and subsequently can scale to large datasets when computer memory capacity is limited.

Variational Bayesian inference is an approximate method for obtaining a strict lower bound of the true (log) joint posterior. VB explicitly takes uncertainty into account at all levels of inference, allowing us to marginalise (albeit under the VB approximations) over unknown variables, rather than selecting the single most likely value. The approximation offers huge speed ups over Monte Carlo sampling based Bayesian methods [28], and the performance degradation appears small in BCC models. Hence, VB is our preferred algorithm for working with potentially large sets of documents. In our experiments we implement our VB algorithm using Infer.NET [20], which is a framework that enables rapid development and running of Bayesian inference in graphical models. In particular, the Infer.NET inference engine enables us to switch between alternative inference algorithms for BCCWords, including Gibbs sampling [9] and Expectation propagation [19], that are potentially more accurate but much slower than VB and less suitable for performing inference over large–scale datasets.

The variational Bayesian inference algorithm uses an approximation to the joint probability distribution, $q(\boldsymbol{t}, \boldsymbol{\theta}) = q(\boldsymbol{t})q(\boldsymbol{\theta})$, that factorises between the true classes of the documents, $\boldsymbol{t} = \{t_i \forall i\}$, and the set of model parameters, $\boldsymbol{\theta} = \{\boldsymbol{\pi}_c^{(k)} \forall c \forall k, \boldsymbol{\omega}_c \forall c, \boldsymbol{\rho}\}$. The algorithm iterates between updating the approximate posterior distribution over the true classes of the documents, $q(\boldsymbol{t})$, and the model parameters $q(\boldsymbol{\theta})$, until it converges. The theory behind variational inference guarantees that each iteration reduces the Kullback-Leibler divergence [16] between the approximate solution and the true posterior at each iteration, so that the approximation becomes closer to the exact solution with each iteration. The updates can be viewed as passing messages between the true class labels $\boldsymbol{t}$ and model parameters $\boldsymbol{\theta}$. As we will see in the detailed explanation below, the specific forms of the factors $q(\boldsymbol{t})$ and $q(\boldsymbol{\theta})$ arise naturally from the BCCWords model and its choice of distributions. The conditional independence relations in our model allow us to further factorise the distribution over the model parameters, $q(\boldsymbol{\theta})$ without additional approximation:

$$q(\boldsymbol{\theta}) = q(\boldsymbol{\rho}) \prod_{c \in C} \left\{ q(\boldsymbol{\omega}_c) \prod_{k \in K} q(\boldsymbol{\pi}_c^{(k)}) \right\}$$

This means we can update each subset of model parameters separately, and each of these factors will exchange different messages with $q(\boldsymbol{t})$. This type of algorithm is also known as *variational message passing (VMP)* and has an efficient scalable implementation which is described in Section 4.2.

4.1 The BCCWords–VB Algorithm

We now present the implementation of the variational inference approach to BCCWords described in the previous section. Specifically, we describe the details of the iterative updates within a step-by-step description of the inference algorithm based on the VB equations that we derive for BCCWords.

Inputs: the algorithm takes as input a data set of annotators' responses, $\boldsymbol{l}$, and where available, a set of known target labels which are gold-standard training labels. We note that gold labels are not necessary and the algorithm can operate in unsupervised mode. To run the algorithm, we must also select values for $\boldsymbol{\alpha}_{0,c}^{(k)}, \forall k$, $\boldsymbol{\beta}_0$ and $\boldsymbol{\gamma}_{0,c}$, as described above. A number of techniques can be used to initialise these hyperparameters when the choice of values is unclear [3].

Step 1. Initialisation: initialise approximate posterior distributions over the model parameters, $\boldsymbol{\theta}$. The choice of initial distributions affects the number of iterations required for convergence. In our implementation we initialise the posterior distributions over the model parameters by setting them to their prior distributions.

Step 2. Update true class predictions: update the approximate posterior $q(t_i)$ over the class of each document, $i \in N$. For any document i which has a gold label, the value of t_i is known so we do not need to update $q(t_i)$. Instead we set $q(t_i = c) = 1$ where c is the observed value of t_i, and $q(t_i \neq c) = 0$ for all other class values. For all other documents, we obtain the current estimate $q^*(t_i)$ of the probability that the true class of i is c:

$$q^*(t_i = c) = \frac{r_{i,c}}{\sum_{c' \in C} r_{i,c'}}, \quad (2)$$

where $r_{i,c}$ is the expectation of the likelihood, given by

$$r_{i,c} = \mathbb{E}_{\boldsymbol{\rho}, \boldsymbol{\pi}_c, \boldsymbol{\omega}_c}[\ln p(t_i = c, \rho_c, \boldsymbol{\pi}_c, \boldsymbol{l}, \boldsymbol{\omega}_c)]$$
$$= \mathbb{E}_{\boldsymbol{\rho}}[\ln \rho_c] + \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{\pi}_c^{(k)}}[\ln \pi^{(k)}_{c, l_i^{(k)}}] + \sum_{n=1}^{W_i} \mathbb{E}_{\boldsymbol{\omega}_c}[\ln \omega_{c, w_{i,n}}]. \quad (3)$$

The expectations in this equation are found using the current estimates of the distributions over the model parameters, and are defined explicitly in the subsequent steps of

the algorithm. These terms can be seen as messages from the model parameters to the true class labels, $\boldsymbol{t}$.

Equation 2 can then be used to determine the messages to pass to the model parameters $\boldsymbol{\theta}$, which are expectations over the sufficient statistics of the set of true class labels for all documents. The message for $\boldsymbol{\rho}$ contains expected counts of each true class:

$$N_c = \sum_{i=1}^{N} q(t_i = c).$$

The message for the confusion matrices contains the counts of each judgement label $c' \in C$ given the true label $c \in C$:

$$N_{c,c'}^{(k)} = \sum_{i=1}^{N} \delta_{l_i^{(k)},c'} \; q(t_i = c) \tag{4}$$

where $\delta_{l_i^{(k)},c'}$ is the Kronecker delta and is unity if $l_i^{(k)} = c'$ and zero otherwise. Similarly, the message for the word distributions contains counts of word occurrences in each class:

$$N_{c,d} = \sum_{i=1}^{N} \sum_{n=1}^{W_i} \delta_{w_{i,n},d} \; q(t_i = c). \tag{5}$$

Step 3. Update confusion matrices: update the approximate posterior $q(\boldsymbol{\pi}_c^{(k)})$ for each class $c \in C$ and each annotator $k \in K$. The prior distributions over the confusion matrices are Dirichlet distributions, which are conjugate to the categorical distributions. This means that the posterior distributions over the confusion matrices are also Dirichlets, with updated parameters:

$$q^*\left(\boldsymbol{\pi}_c^{(k)}\right) = \mathrm{Dir}\left(\boldsymbol{\pi}_c^{(k)} | \alpha_{c,1}^{(k)}, ..., \alpha_{c,L}^{(k)}\right), \tag{6}$$

where L is the cardinality of C and $\boldsymbol{\alpha}_c^{(k)}$ is calculated by adding counts from the true class label message to the prior pseudo-counts $\alpha_{0,c}^{(k)}$:

$$\alpha_{c,c'}^{(k)} = \alpha_{0,c,c'}^{(k)} + N_{c,c'}^{(k)}. \tag{7}$$

A more detailed derivation of these iterative update equations can be found in [28]. We can now calculate the message to send back to the true labels, which is the expectation term required for Equation (3):

$$\mathbb{E}\left[\ln \pi_{c,c'}^{(k)}\right] = \Psi\left(\alpha_{c,c'}^{(k)}\right) - \Psi\left(\sum_{b=1}^{L} \alpha_{c,b}^{(k)}\right), \tag{8}$$

where $\Psi(.)$ is the standard *digamma* function.

Step 4. Update word distributions: update the approximate posterior $q(\boldsymbol{\omega}_c)$ for each row $c \in C$ to current estimate $q^*(\boldsymbol{\omega}_c)$. Again, we have a posterior Dirichlet distribution due to the use of conjugate exponential-family distributions in our model:

$$q^*\left(\boldsymbol{\omega}_c\right) = \mathrm{Dir}\left(\boldsymbol{\omega}_c | \gamma_{c,1}, ..., \gamma_{c,D}\right), \tag{9}$$

where the parameters are updated by $\gamma_{c,d} = \gamma_{0,c,d} + N_{c,d}$. The message to the true class labels, which is required for Equation (3), contains the terms:

$$\mathbb{E}\left[\ln \omega_{c,d}\right] = \Psi\left(\gamma_{c,d}\right) - \Psi\left(\sum_{d'=1}^{D} \gamma_{c,d'}\right). \tag{10}$$

Step 5. Update class proportions: update the approximate posterior $q(\boldsymbol{\rho})$ using the Dirichlet parameter update:

$$q^*\left(\boldsymbol{\rho}\right) = \mathrm{Dir}\left(\boldsymbol{\rho} | \beta_1, ..., \beta_C\right) \tag{11}$$

where the parameters are updated by $\beta_c = \beta_{0,c} + N_c$. The message from this parameter is:

$$\mathbb{E}\left[\ln \rho_c\right] = \Psi\left(\beta_c\right) - \Psi\left(\sum_{b=1}^{L} \beta_b\right). \tag{12}$$

So, for one iteration of the algorithm, we calculate the updated parameters, distributions and expectation terms defined in steps 2 to 5.

Step 6. Check convergence: if the target label distributions $q^*(t_i = c)$ have not converged to a stable solution within a given tolerance, repeat the algorithm from Step 2.

Outputs: predictions of the document class labels, given by the current estimates of $q(t_i = c) \forall i \forall c$, their posterior expectations. The algorithm also outputs approximate posterior distributions over the model parameters, $q(\boldsymbol{\pi}_c^{(k)})$, $q(\boldsymbol{\omega}_c)$, and $q(\boldsymbol{\rho})$ for each row $c \in C$ and each annotator $k \in K$.

4.2 Scalability Through Inference Decomposition

Performing a task such as sentiment analysis or disaster report analysis can require us to work with extremely large datasets with vast memory requirements. A major source of memory usage is the large set of annotator confusion matrices that the inference algorithm must iteratively update. For example, the Ushahidi dataset, gathered after the Haiti earthquake, was interpreted by approximately 700 workers [22]. To resolve memory exhaustion difficulties of VB inference at scale, this section proposes a scalable version of the BCCwords-VB algorithm, *Scalable BCCwords* (ScalBCCWords), which can be run on a single computer. ScalBCCWords is identical to BCCWords except that we decompose the entire data set into a set of *batches* of data by distributing the annotators across P partitions. During each iteration of the VB inference algorithm ScalBCCWords switches each batch in and then out of memory in turn. Batches produce messages that summarise each portion of data and occupy considerably less computer memory than the entire data set.

We chose to distribute the workers between the batches so that each batch contains all the responses from workers in that batch. This partitioning criterion is sensible as each batch only has to represent a subset of annotators, and thus only represents a small set of confusion matrices. The splits can be chosen to meet memory constraints. The corresponding factor graph is shown in Figure 2, showing how all partitions share the same class distribution of documents and the word distributions conditioned on document class.

When batch p is processed, the pseudo counts $N_{c,c'}^{(k)}$ are calculated using Equation (4), for all $k \in p$. The log confusion matrix messages, $M_{p,c,i}$, for batch p for each class c and document i are calculated as follows,

$$M_{p,c,i} = \sum_{k \in p} \mathbb{E}\left[\ln \pi_{c,l_i^{(k)}}^{(k)}\right]$$

using Equation 8 to calculate the expected log confusion matrix. The log confusion matrix for batch p is then deleted

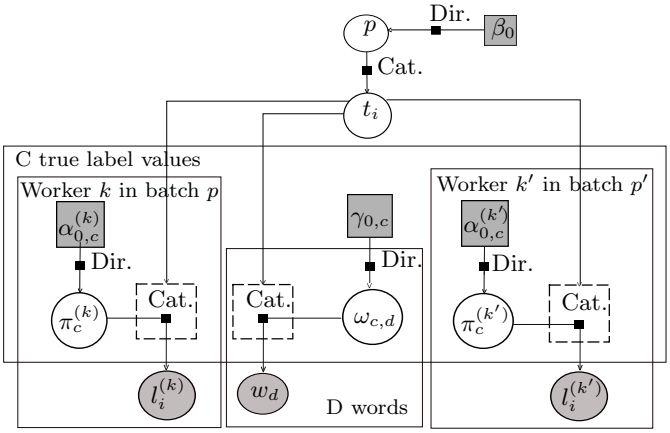


Figure 2: Factor graph for Scalable BCCwords. The four plates included in the graph describe (i) the set of workers K in the batch p, (ii) the set of workers K' in the batch p', (iii) the C possible true label values and (iv) the D words contained in the dictionary of terms used in the tweets. The plate for the N documents is omitted for simplification.

from memory. Once a message is obtained for each batch, the log true class prediction probability is calculated using,

$$r_{i,c} = \mathbb{E}_{\boldsymbol{\rho}}[\ln \rho_c] + \sum_{p \in P} M_{p,c,i} + \sum_{n=1}^{W_i} \mathbb{E}_{\boldsymbol{\omega}_c}[\ln \omega_{c,w_{i,n}}]$$

as per Equation 3. Hence, scalBCCWords is mathematically equivalent to BCCWords and both methods converge to the same solution. The remaining steps of the scalBCCWords algorithm are identical to those of BCCWords. We note that ScalBCCwords may process the batches in any order and not all the batches need be updated during each iteration of the VB algorithm. In our experiments, we provide an empirical evaluation of both our algorithms showing the advantage of ScalBCCWords-VB in memory occupancy.

5. EMPIRICAL EVALUATION

We evaluate the efficacy of our approach, ScalBCCWords, using two real-world datasets, comparing performance against the following five rival benchmark approaches. We note that BCCWords-VB and ScalBCCWords produce the same classifications as they are mathematically equivalent and therefore only ScalBCCWords results are shown.

Majority Voting (MV) is a popular and simple algorithm for obtaining a single decision from multiple opinions provided by a crowd [18, 29]. MV greedily assigns a class to each document by choosing the label with the most votes from the crowd. All votes are considered with uniform weight, thus treating all annotators as equally reliable. Typically, no measure of uncertainty in the final decision is provided.

Vote Distribution treats the fraction of votes in support of each class as the probability of that class. It therefore represents a simple technique for estimating the empirical probability that a document has a particular true label, assuming that all annotators are equally reliable.

Bag-of-words Classifier + MV trains a bag-of-words classifier by treating the majority vote as gold-labelled training data [11]. Therefore, this approach learns a language model that can be used to classify documents that have not yet been labelled by the crowd, but does not account for varying reliability of the crowd labellers when training the model.

Dawid & Skene is a model for combining labels from multiple classifiers, using confusion matrices to model the reliability of individual labellers [7]. The learning algorithm for Dawid & Skene does not account for uncertainty in the confusion matrices or other model parameters, which can lead to errors when gold-labelled data is limited.

Independent Bayesian Classifier Combination (IBCC) learns the confusion matrices using variational Bayes (VB). Therefore, in contrast to Dawid&Skene, it handles model uncertainty and can operate in unsupervised settings when gold-labelled examples are unavailable. However, this model does not consider text features and relies solely on labels provided by the crowd.

Community-Based Bayesian Classifier Combination (CBCC) is an extension of IBCC that models communities of workers with similar confusion matrices. It learns both the confusion matrices of each community and each worker but, like IBCC, it does not account for text features in the documents [30]. We run CBCC with three communities as suggested in the original paper, for both CF and SP.

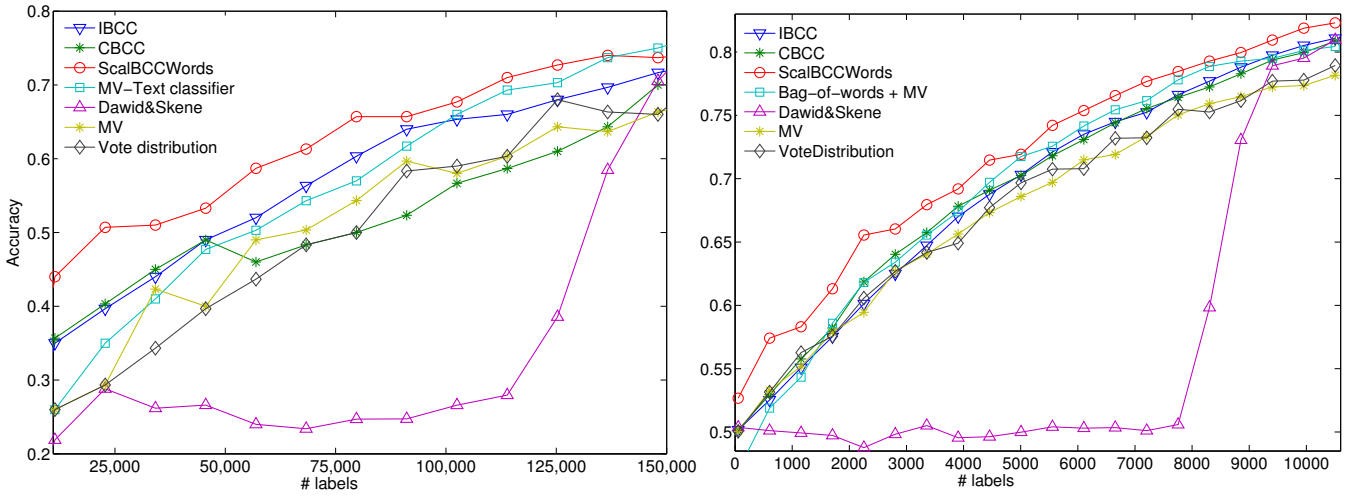
In our experiments, we set the priors for IBCC, CBCC and ScalBCCWords as follows. For the class proportions, $\boldsymbol{\rho}$, we used unbiased priors by setting the values of $\beta_{0,c}$ to be equal for all classes. For the workers' confusion matrices, we used informative priors, setting the diagonal counts $\alpha_{0,c,c}^{(k)}$ to $C + 1.5$, with the off-diagonals set to 1. This means that workers are initially assumed to be reasonably accurate. For ScalBCCWords, the word distributions were given uninformative priors, by setting uniform values for $\gamma_{0,c,d}$ for all words $d \in D$.

5.1 Datasets

We evaluate our approach using two crowdsourcing datasets, which provide real sentiment judgements obtained from human workers. The two datasets demonstrate our approach on two very different kinds of document, with distinct sentiment analysis problems.

The CrowdFlower dataset (CF) was provided by CrowdFlower[3] as part of the 2013 Crowdsourcing at Scale shared task challenge. The dataset contains 569,375 judgements for 98,980 tweets. This dataset includes 300 tweets with gold-standard sentiment labels, which correspond to 1,720 judgements from 461 workers. The judgements reflect the sentiment of tweets discussing the weather, and can take values from four sentiment categories: negative (0), neutral (1), positive (2), tweet not related to weather (4) and cannot tell (5). This dataset therefore concerns a multi-class labelling problem.

[3] www.crowdflower.com



(a) CrowdFlower (CF)

(b) Sentiment Polarity (SP)

Figure 3: Accuracy of seven methods measured with increasing proportions of labels for both datasets.

The Sentiment Polarity dataset (SP) contains annotations for a set of 5,000 sentences from movie reviews, extracted by [23] from the website RottenTomatoes[4]. This dataset has gold-standard sentiment labels for all the movie reviews assigned by the website, which marked them as either "fresh" (positive) or "rotten" (negative). A set of 27,747 sentiment judgements were collected from 203 workers using Amazon Mechanical Turk (AMT)[5] by [27]. The SP dataset therefore presents a binary sentiment analysis problem, with workers forced to select either positive (1) or negative (0), with no option to express their uncertainty.

The vocabulary of a real-world text corpus is often extremely large, so most practical deployments of language modelling methods employ a set of heuristic pre-processing steps to remove noisy data that would otherwise add unnecessary computation and memory costs. In our experiments, the dictionary for both datasets was obtained using the standard approach of stemming the text through the Porter's stemmer algorithm [24], then removing common stop words, before extracting the 300 words with the highest term frequency x inverse document frequency (TF-IDF) score [2]. TF-IDF is a heuristic for selecting words that are important in distinguishing documents within a corpus, where term frequency is the number of occurrences of word d within the corpus, and inverse document frequency $= \log(N/N_d)$, where N_d is the number of documents containing d. While ScalBCCWords is agnostic to the type of features supplied, these standard text pre-processing steps are used to allow the experiments to focus on comparing crowdsourced sentiment classification methods.

5.2 Performance Comparisons

We investigated how the effectiveness of the language model learnt by ScalBCCWords varies with the number of labels supplied by the crowd. To do this we compared the performance of the alternative methods on our two datasets and evaluated the efficacy of each using four standard metrics:

Accuracy is the proportion of documents that were correctly labelled. For methods such as ScalBCCWords that output probabilities we assign the label with the highest probability.

Average recall is the mean across all classes of the recall rate, defined as the fraction of positive instances of a given class that were correctly labelled.

Negative log probability density (NLPD) is an error measure – the lower the better – based on how much weight a classifier gave to the correct class of each document as defined in [30].

AUC is the area under the curve of the receiver operating characteristic (RoC), which is the probability that a randomly-chosen positive example is assigned a higher probability than a randomly-chosen negative example [28]. This is a measure of an algorithm's ability to differentiate between classes, regardless of whether the classes are imbalanced. For the CF dataset, we provide the mean AUC over pairs of classes, using the method of [10].

The experiment is run iteratively, starting by running each method with 2% randomly-chosen judgements from the crowd, then evaluating the classification efficacy. We then increase the number of judgements by adding a further 2% randomly-selected labels from the crowd and re-running all the methods. This process is repeated until all crowdsourced labels have been used by the prediction methods.

Figure 3 shows the accuracy for the six methods for both datasets, which improves for all methods as they get more data. In particular, ScalBCCWords has highest accuracy for a small number of labels, demonstrating the added value of the language model. ScalBCCWords maintains the highest accuracy throughout, although IBCC, CBCC and Dawid & Skene catch up for large numbers of crowdsourced la-

[4] www.rottentomatoes.com
[5] www.mturk.com

Method	CF (20% labels) Accuracy	Avg. recall	NLPD	AUC	SP (20% labels) Accuracy	Avg. recall	NLPD	AUC
Majority vote	0.630	0.556	1.375	0.731	0.718	0.718	1.170	0.715
Vote distribution	0.653	0.592	1.227	0.749	0.706	0.706	0.945	0.738
Bag-of-words classifier + MV	0.670	0.608	2.280	0.755	0.726	0.726	0.642	0.792
Dawid&Skene	0.613	0.497	1.276	0.672	0.500	0.500	0.695	0.500
IBCC	0.693	0.553	0.950	0.811	0.738	0.738	**0.506**	0.833
CBCC	0.647	0.534	1.000	0.802	0.728	0.728	0.516	0.830
Scalable BCCwords	**0.717**	**0.578**	**0.909**	**0.836**	**0.755**	**0.755**	0.533	**0.843**

Table 1: Accuracy, average recall and negative log probability density score (NLPD) for the CF and the SP datasets for the six tested methods (one for each row) when using 20% crowdsourced labels. Using this subset of labels, 70% of the documents in both datasets have at least one crowdsourced label.

Method	CF (all labels) Accuracy	Avg. recall	NLPD	AUC	SP (all labels) Accuracy	Avg. recall	NLPD	AUC
Majority vote	0.840	0.764	0.921	0.852	0.885	0.885	0.797	0.885
Vote distribution	0.883	0.779	0.458	0.942	0.887	0.887	0.338	0.947
Bag-of-words classifier + MV	0.867	0.764	0.921	0.859	0.885	0.885	0.797	0.891
Dawid&Skene	0.830	0.745	0.459	0.897	0.914	0.914	**0.340**	**0.957**
IBCC	0.860	0.763	**0.437**	0.935	**0.915**	**0.915**	0.374	**0.957**
CBCC	0.886	0.746	0.526	**0.942**	**0.915**	**0.915**	0.383	**0.957**
Scalable BCCwords	**0.890**	**0.807**	0.591	0.877	**0.915**	**0.915**	0.389	**0.957**

Table 2: Accuracy, average recall and negative log probability density score (NLPD) for the CF and the SP datasets for the six tested methods (one for each row) when using all available crowdsourced labels.

bels. The accuracy of ScalBCCWords is 25% higher than CBCC (0.57 vs. 0.40) after 20,000 labels in CF and is 8% higher than CBCC (0.54 vs. 0.50) after 110 labels in SP. Importantly, in order to achieve the same accuracy, ScalBCCWords requires up to 56,935 fewer labels in CF and up to 440 fewer labels in SP compared to the benchmarks. Furthermore, Dawid & Skene initially infers a poor model of worker accuracy due to scarce labels, which leads to poor classification performance. Such a cold start phase is mitigated in the BCC methods by accounting for uncertainty in the workers' accuracy. Both MV and Vote Distributions are more accurate than Dawid & Skene in the initial phase but they are less accurate than all the other methods when a larger amount of crowd labels have been used.

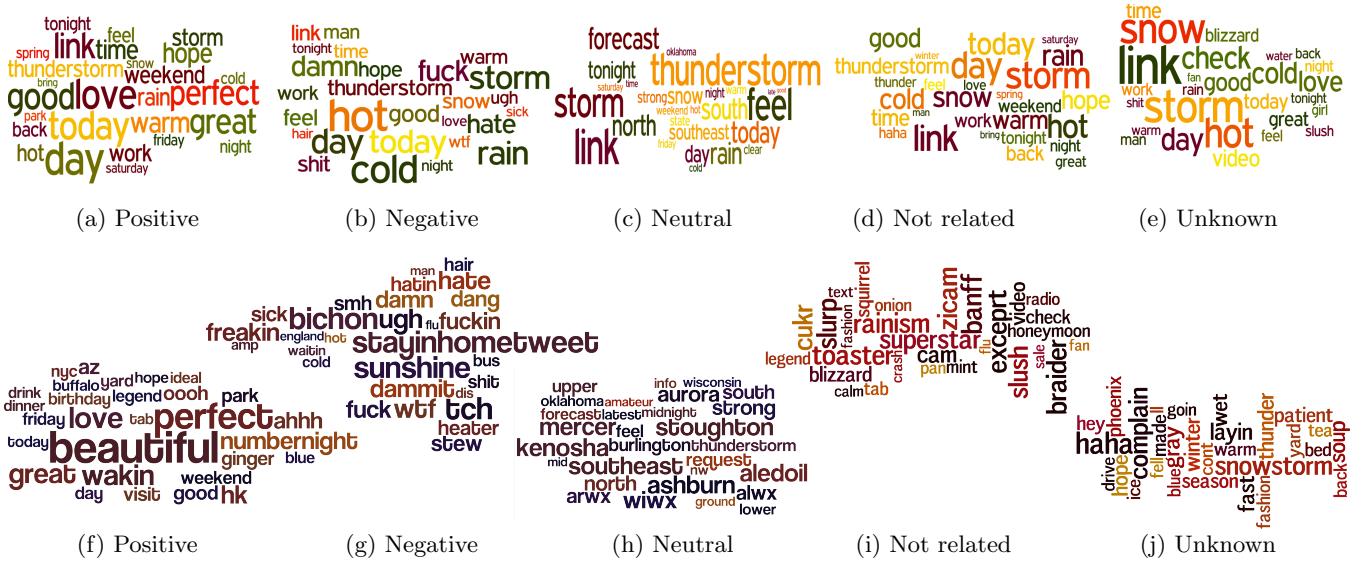
Table 1 shows prediction metrics for both datasets when only 20% of the complete set of crowdsourced labels were used. Using this subset reduces the number of labels available for each document so that only 70% of documents have one or more crowdsourced labels. To classify the 30% of documents with no crowdsourced labels, ScalBCCWords and the bag-of-words classifier apply their language models, while other methods have no information about these documents and assign a default category to all unlabelled documents. ScalBCCWords has the highest accuracy and average recall on both datasets, significantly outperforming the bag-of-words model, which also uses a language model but does not model annotator bias and error rates.

Table 2 shows prediction metrics for both datasets, when all crowdsourced labels were used. ScalBCCWords has the highest accuracy and average recall on both datasets and competitive AUC and NLPD on the SP dataset. This demonstrates that ScalBCCWords performs well even when labels are plentiful (in this case, on average 6 labels per document).

5.3 Language Model

The language model inferred by ScalBCCWords represents the probabilities of each word in the dictionary conditioned on the sentiment classes. In Figure 4, the top row shows the Wordles (word clouds) with the most probable 27 words in the five sentiment classes of the CF dataset. ScalBCCWords is able to identify words that discriminate the sentiment classes, such as "love" and "perfect", which are more likely to occur in the positive sentiment class, whereas words such as "cold" and "hate" are more likely to appear in the negative class. We note that common words such as "day" are highly likely in both positive and negative classes and are therefore not good discriminators in this dataset. However, ScalBCCWords naturally uses the most discriminative words to infer the sentiment class through Equation 2. In the second row of Figure 4, the wordles show the words that most strongly indicate the class, i.e. the words d with highest probability $p(t_i = c|w_{i,n} = d)$ for class c. Here we can see that "day" is not indicative of sentiment class and there is little overlap between the word clouds for each class. Figure 5 shows the Wordles for the SP dataset, with the word "good" being equally likely in both sentiment classes, suggesting that words that seem intuitively positive may be poor discriminators, possibly because their meaning is highly context-dependent.

To validate the quality of the language model inferred by ScalBCCWords using the crowd labels, we compare it to a language model learned by training ScalBCCWords on the gold-standard labels. For both models, we rank words by their probabilities $\omega_{c,d}$ in each class c, to examine which terms the model has inferred are important to each class. Using the non-parametric Kendall's τ rank correlation test, we find a significant positive correlation between the rank-



(a) Positive (b) Negative (c) Neutral (d) Not related (e) Unknown

(f) Positive (g) Negative (h) Neutral (i) Not related (j) Unknown

Figure 4: Top row (a) to (e): word clouds of the most probable 27 words from ScalBCCWords for the sentiment classes of the CF dataset. Word size is proportional to estimated likelihood given the true label class. Second row (f) to (j): word clouds for the most discriminative 27 words for three classes of the CF dataset. Word size is proportional to the class probability given the word. Colours are for legibility only.

	Positive		*Negative*		*Neutral*		*Unknown*		*Not related*	
	Gold	Crowd	Gold	Crowd	Gold	Crowd	Gold	Crowd	Gold	Crowd
Gold	1.0	0.384	1.0	0.337	1.0	0.423	1.0	0.403	1.0	0.373
Crowd	0.384	1.0	0.337	1.0	0.423	1.0	0.403	1.0	0.373	1.0

(a) CrowdFlower (CF)

	Positive		*Negative*	
	Gold	Crowd	Gold	Crowd
Gold	1.0	0.977	1.0	0.975
Crowd	0.977	1.0	0.975	1.0

(b) Sentiment Polarity (SP)

Table 3: The Kendall's τ rank correlation coefficients ($p < 10^{-5}$) for the word distributions, $\omega_{c,d}$, estimated by BCCWords using gold labels and crowd judgements. Colour intensity indicates the correlation strength.

(a) Positive (b) Negative

Figure 5: Word clouds of the most probable 27 words from ScalBCCWords for the sentiment classes of the SP dataset. Word size is proportional to estimated likelihood given the true label class. Colours are for legibility only.

ings obtained from the model trained on the crowdsourced data and the model trained on gold labels, as shown in Table 3 for both datasets. This indicates that ScalBCCWords can effectively train a language model using crowd labels when gold-standard data is unavailable. Kendall's τ is much higher for SP, which reflects the higher accuracy of ScalBCCWords on the SP dataset shown in Table 2. This suggests that the crowd's decisions may more accurately reflect the gold-labelled data when classifying the SP dataset, which has only two diametrically opposed classes, rather than the five less easily distinguished classes of CF.

5.4 Profiling Crowd Workers

Besides predicting document classifications and the language model, ScalBCCWords learns the confusion matrices that characterise the workers' skill levels across sentiment classes. Figure 6 shows example crowd members with very different confusion matrices. For example, subfigure (a) shows a competent annotator who provides accurate labels across all sentiment classes, hence the highly peaked likelihoods along the diagonals. Subfigure (b) shows an annotator whose reliability is inconsistent across the classes, with varying likelihoods of incorrect worker labels. This figure shows that we are able to detect annotators with very different behaviour within our two real-world datasets. The BCCWords model captures not just the overall skill level, but also the

accuracy and bias of the annotator for each specific class, shown by the distribution in each row of Figure 6.

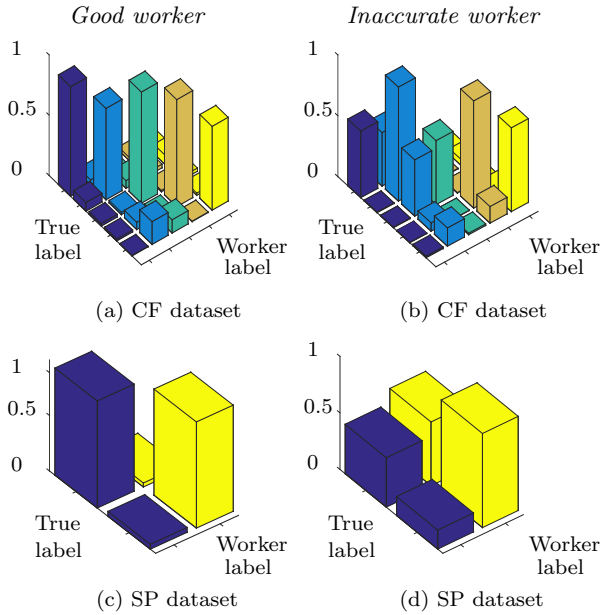


Figure 6: Confusion matrices of four workers, with the likelihood of worker label given true class on the vertical axis. These profiles show two workers who are very well calibrated in their opinions (left) and two workers who provide less accurate labels (right).

5.5 Memory Usage

We compared the memory usage of BCCWords-VB and ScalBCCWords on the CF dataset. Figure 7 shows a plot of memory demand when running the BCCWords-VB algorithm with increasing subsets of labels. In particular, we measured memory demand through the standard memory profiler available in .NET[6] that provides the approximate memory allocated on the garbage collection heaps to store the model instances of BCCWords. Despite the high noise of these counters, which explains the variability of the curves in the graph, we can still observe the general increasing trend of memory usage when using more labels. As shown in Figure 7, the ScalBCCWords algorithm uses up to 80% less memory than the standard implementation of BCCWords (1 GB vs. 200 MB) when the dataset includes 50,000 labels.

6. CONCLUSIONS

This paper presents BCCWords, a novel algorithm for combining crowdsourced annotations with text features in order to determine the sentiment of documents. We presented a scalable variational Bayes inference algorithm for BCCWords and demonstrated how it can be implemented for a large corpus annotated by crowd workers. Our analysis demonstrated that BCCWords is able to identify key differentiating text features, which produce more accurate sentiment classifications when crowdsourced labels are scarce. It is able to classify short messages such as tweets, despite the limited number of text features in these short messages.

[6]CLR Profiler, `clrprofiler.codeplex.com`

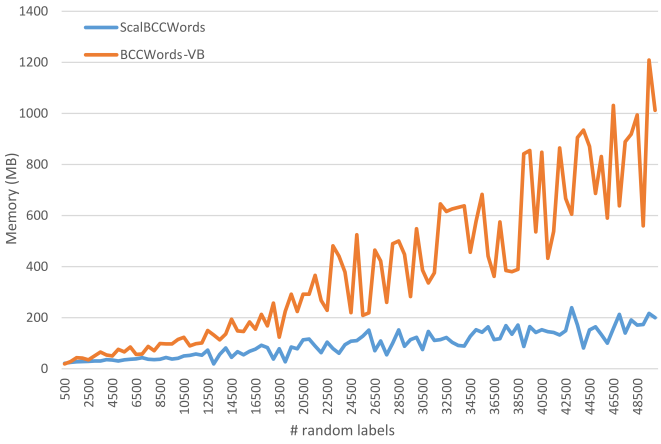


Figure 7: Memory usage of BCCWords (orange line) and the scalable implementation of BCCWords (blue line) measured on real data.

We compared our algorithm with six benchmark methods on two real-world crowdsourcing datasets and showed that our method can improve accuracy by 25% over both standard text classifiers and prominent aggregation models for crowdsourced data with annotations for a small portion of documents. Furthermore, our approach significantly reduces, by up to 67%, the amount of labels that must be obtained through crowdsourcing to achieve comparable accuracy with rival methods.

We are currently investigating other prominent applications of our method: identifying aid requirements in disaster response using reports from members of the public and first responders; evaluating investor sentiment towards companies expressed in free-text reports; and determining student sentiment from online forum postings to aid pastoral care. These domains provide vast amounts of unstructured data that can benefit from insights provided by human annotators and the scalability of automated methods. Future work will evaluate BCCWords with the different types of features available in these domains, including alternative text features, document metadata, and image features.

The way that BCCWords learns the annotator confusion matrices could be modified for problems with ordinal classes, such as those representing different strengths of sentiment, to take advantage of relationships between neighbouring classes when samples of annotator behaviour for each label and true class value are sparse.

7. ACKNOWLEDGMENTS

We thank Gabriella Kazai, Lumi, London, UK for initial discussions. This work was funded by the EPSRC ORCHID programme grant (EP/I011587/1) and Microsoft Research, Cambridge, UK.

8. REFERENCES

[1] Y. Bachrach, T. Graepel, T. Minka, and J. Guiver. How To Grade a Test Without Knowing the Answers – A Bayesian Graphical Model for Adaptive Crowdsourcing and Aptitude Testing. In *Proc. of the 29th Int. Conf. on Machine Learning*, pages 1183–1190. ACM, 2012.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[3] J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *The Journal of Machine Learning Research*, 13:281–305, 2012.

[4] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 4th edition, 2006.

[5] D. Butler. Crowdsourcing Goes Mainstream in Typhoon Haiyan Response. *Nature News*, doi:10.1038/nature.2013.14186, 2013.

[6] C. Chew and G. Eysenbach. Pandemics in the Age of Twitter: Content Analysis of Tweets during the 2009 H1N1 Outbreak. *PloS One*, 5(11):e14118, 2010.

[7] A. P. Dawid and A. M. Skene. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28, Jan. 1979.

[8] P. Donmez, J. Carbonell, and J. Schneider. A Probabilistic Framework to Learn from Multiple Annotators with Time-Varying Accuracy. In *Proc. of the Int. Conf. on Data Mining*, pages 826–837, 2010.

[9] A. Gelfand and A. Smith. Sampling-Based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990.

[10] D. J. Hand and R. J. Till. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Machine learning*, 45(2):171–186, 2001.

[11] Z. S. Harris. Distributional Structure. *Word*, pages 146–162, 1954.

[12] N. R. Jennings, L. Moreau, D. Nicholson, S. D. Ramchurn, S. Roberts, T. Rodden, and A. Rogers. On Human-Agent Collectives. *Communications of the ACM*, 2014.

[13] E. Kamar, S. Hacker, and E. Horvitz. Combining Human and Machine Intelligence in Large-Scale Crowdsourcing. In *Proc. of the 11th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 467–474, 2012.

[14] H. Kim and Z. Ghahramani. Bayesian Classifier Combination. In *Proc. of the 15th Int. Conf. on Artificial Intelligence and Statistics*, page 619, 2012.

[15] F. Kivran-Swaine, S. Brody, and M. Naaman. Effects of Gender and Tie Strength on Twitter Interactions. *First Monday*, 18(9), 2013.

[16] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[17] A. Levenberg, S. Pulman, K. Moilanen, E. Simpson, and S. Roberts. Predicting Economic Indicators from Web Text Using Sentiment Composition. In *Int. Journal of Computer and Communication Engineering*, 2014.

[18] N. Littlestone and M. Warmuth. The Weighted Majority Algorithm. In *30th Annual Symposium on Foundations of Computer Science*, pages 256–261. IEEE, 1989.

[19] T. Minka. Expectation Propagation for Approximate Bayesian Inference. In *Proc. of the 17th Conf. on Uncertainty in Artificial Intelligence*, page 362, 2001.

[20] T. Minka, J. Winn, J. Guiver, and D. Knowles. Infer.NET 2.5. Microsoft Research Cambridge. *See http://research. microsoft. com/infernet*, 2013.

[21] K. Moilanen and S. Pulman. Sentiment composition. In *Proc. of the Recent Advances in Natural Language Processing Int. Conf.*, pages 378–382, 2007.

[22] N. Morrow, N. Mock, A. Papendieck, and N. Kocmich. Independent Evaluation of the Ushahidi Haiti Project. *Development Information Systems.*, 8:2011, 2011.

[23] B. Pang and L. Lee. A Sentimental Education: Sentiment Analysis using Subjectivity Summarization Based on Minimum Cuts. In *Proc. of the 42nd annual meeting on Association for Computational Linguistics*, page 271, 2004.

[24] M. F. Porter. An algorithm for suffix stripping. *Program: Electronic library and Information Systems*, 14(3):130–137, 1980.

[25] V. C. Raykar and S. Yu. Eliminating Spammers and Ranking Annotators for Crowdsourced Labeling Tasks. *Journal of Machine Learning Research*, 13:491–518, 2012.

[26] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning From Crowds. *Journal of Machine Learning Research*, 11:1297–1322, 2010.

[27] F. Rodrigues, F. Pereira, and B. Ribeiro. Learning from Multiple Annotators: Distinguishing Good from Random Labelers. *Pattern Recognition Letters*, 34(12):1428–1436, 2013.

[28] E. Simpson, S. Roberts, I. Psorakis, and A. Smith. Dynamic Bayesian Combination of Multiple Imperfect Classifiers. In *Intelligent Systems Reference Library series: Decision Making and Imperfection*, pages 1–35. Springer, 2013.

[29] L. Tran-Thanh, M. Venanzi, A. Rogers, and N. R. Jennings. Efficient Budget Allocation with Accuracy Guarantees for Crowdsourcing Classification Tasks. In *Proc. of the 12th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 901–908, 2013.

[30] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi. Community-based Bayesian Aggregation Models for Crowdsourcing. In *Proc. of the 23rd Int. Conf. on World Wide Web*, pages 155–164, 2014.

[31] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. The Multidimensional Wisdom of Crowds. In *Advances in Neural Information Processing Systems*, pages 2424–2432, 2010.

[32] 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 *Advances in Neural Information Processing Systems*, pages 2035–2043, 2009.

[33] K. W. Willett, C. J. Lintott, S. P. Bamford, K. L. Masters, B. D. Simmons, K. R. Casteels, E. M. Edmondson, L. F. Fortson, S. Kaviraj, W. C. Keel, et al. Galaxy Zoo 2: Detailed Morphological Classifications for 304,122 Galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860, 2013.