Using a Bayesian Model to Combine LDA Features with Crowdsourced Responses

Edwin Simpson and Steven Reece

Department of Engineering Science, University of Oxford, UK.
{edwin, reece}@robots.ox.ac.uk

Antonio Penta and Sarvapali Ramchurn

School of Electronics and Computer Science, University of Southampton, UK.
{ap7, sdr1}@ecs.soton.ac.uk

February 5, 2013

Abstract

This paper describes a crowdsourcing system that integrates machine learning techniques with human classifiers, showing how to apply a Bayesian approach to classifier combination to the challenge of crowdsourcing document topic labels. First, we use a number of NLP techniques to extract informative document features. We then screen and select workers using Amazon Mechanical Turk to label selected documents. We then apply Independent Bayesian Classifier Combination (IBCC) to classify the complete set of documents in a semi-supervised manner, taking into account the unreliability of crowd-sourced labels. More documents are then selected intelligently for labeling by the crowd. We demonstrate superior results using IBCC compared to a two-stage classifier and strong performance with only 16% documents labelled by the crowd.

1 Introduction

Information Retrieval is becoming a serious challenge in the face of Big Data and the Internet of Things where information is generated by people, communities, sensors, and agents on the web. The Crowdsourcing track of the Text Retrieval Conference has the following objectives:

- Develop crowdsourcing techniques to label 15424 documents
- Use human-machine collaboration to minimise cost and maximise efficiency

This chapter reports on our approach to achieving these objectives using a combination of Natural Language Processing (NLP), Machine Learning, and crowdsourcing.

The TREC challenge was to judge 18260 topic-document pairs. Ten topics were randomly chosen by the TREC organising committee. Each topic had a title, description and narrative and these were used

to determine if a document was relevant to a topic. Examples of topics included *definition of creativity* and *recovery of treasure from sunken ships*. The documents came from the TREC corpus (TREC 8), previously labelled by the National Institute of Standards and Technology (NIST) experts. Document sources included the *Financial Times*, *Los Angeles Times* and *Federal Register* articles. As of writing we are aware that out of 17 groups that participated in the TREC challenge we came in the top four according to the NIST committee's evaluation of our results using standard statistical measures (chiefly the Receiver Operating Curve Area Under Curve (AUC)).

This chapter is structured as follows: Section 2 presents a brief overview of our approach to solving the TREC crowdsourcing challenge problem. Section 3 then describes the natural language processing applied to the documents to extract distinct features for each topic type. Section 4 describes in detail our approach to crowdsourcing using Amazon Mechanical Turk which includes a description of the user interface we built into Amazon Mechanical Turk for document labelling, a method for evaluating a turker's trust and a method for paying turkers upon completion of their labelling tasks. Section 5 describes two classifiers that were developed for predicting labels for unlabelled documents. These classifiers where the *Independent Bayes Classifier Combination* (IBCC) algorithm and a traditional two-stage Bayesian classifier. A mechanism which uses the output of these classifiers to select documents for crowdsourced labelling is then presented in Section 6. The efficacy of the IBCC and two-stage classifier are compared on the TREC challenge problem in Section 7. Finally, we discuss our solution to the TREC challenge problem in Section 8 and offer some ideas for future research.

2 Overview of the Document Classification System

We called on the crowd to label a subset of documents from which we inferred probabilities of topic relevance of the unlabelled documents. We developed a document labelling system comprising the following modules:

1. Feature extraction.
2. Crowdsourcing.
3. Classification to infer probabilities of class labels for unlabelled documents.

Stage 1 was performed initially off-line. Stages 2 and 3 then iterated until all documents were labelled with sufficient confidence.

For the feature extraction module we first applied some classical NLP stages and then extracted different features using word-based counting with the computational complexity of this approach mitigated with word clustering. A measure based on the features extracted was also used to define an initial relevance rank to select the first set of documents to be labelled by the crowd.

To crowdsource the labelling of documents we developed the following components using Amazon Mechanical Turk (AMT) as the crowdsourcing platform:

1. a *Bayesian trust model* which computes the level of trust we can assign to a turker to complete specific "gold" tasks (which are tasks for which the ground truth topic labels are known)
2. a *hiring process* which is the workflow for hiring workers
3. an *HIT (Human Intelligence Task) interface* which controls the presentation of the labelling task to maximize efficient labelling, perform verification and avoid boredom by the turker.

Both the turkers' confidence and accuracy in their gold responses were combined to give an overall indication of the workers' trustworthiness at labelling a document correctly. At each iteration of the system we choose a set of documents to be passed to the AMT such that the documents whose labelling would

yield the greatest potential information gain across the corpus were preferred. The turkers then responded with both a single topic classification (or no topic) and confidence in their labelling.

We have developed a novel Bayesian unsupervised classifier which accommodates agents' own confidence in their responses as well as an assessment of their reliability obtained from the trust model. This classifier uses these unreliable labels extracted from the crowd to learn the correspondence between features and topics. The occurrence of features is assumed to be conditionally independent and the likelihoods used in the Bayesian classifier are learned directly from the data using an unsupervised formalism couched in a Variational Bayes learning framework. The classifier is then used to assign a probability that each document in the corpus belongs to a topic. We describe each of the above modules in the following sections.

3 Natural Language Processing and Feature Extraction

Before extracting the features from the documents, we pre-processed the documents using classical Natural Language Processing (NLP) steps *Part of Speech* (PoS), *Lemmatization* and *Stemming* [1]. The PoS reduced our set of words to nouns and verbs using a maximum entropy classifier. [1] Lemmatization and Stemming were then used to reduce inflectional forms and derivationally related forms of a word to a common base form. For example, the forms "am, are, is " become "be" and "cat, cats, cat's, cats' " become "cat". In particular, we used Pling-stemming [2] for nouns and the lemmatization method from WordNet [3] for verbs.

We compared two feature extraction methods based on word weight models: *Term Frequency-Inverse Document Frequency* (TF-IDF) [4] and *Divergence from Randomness* (DFR) [5]. We observed better performance from the DFR model on a subset of TREC-7 collections but could not manage the huge matrices in central memory for the next stages of the NLP process. So, we used a cluster strategy to aggregate similar words. However, classical clustering algorithms are computationally costly as they use pair-wise distances between words and, for that reason, we clustered using a latent topic model. Latent topics are the foundation of *Latent Dirichlet Allocation* (LDA) [6] for which a document can be approximated as a mixture distribution of topics where each topic is a distribution of words. We applied LDA [2] in our feature extraction process by considering each document as a feature vector generated from the topic distribution. We experimented with different numbers of topics (i.e. 500, 1000, 1500, 2000) [3] and, on a subset of TREC-7 collections, we observed that 2000 topics gave the best performance.

We also used LDA to select the first documents to send to the crowd, aiming to find some examples of documents that match the query topics in the TREC challenge. For each TREC relevance topic, we ranked the documents in the collection using the measure:

$$r(d, L, \tau) = \sum_{i=1}^{N} \sum_{j=1}^{K} \rho(w_i, T_j, \tau)\theta(T_j; d)$$

where d is a document, L=$\{w_1, \ldots, w_N\}$ is a set of words extracted from a TREC topic description, $\tau \in \mathbb{N}$ is a threshold (in our experiments set to 50), T_i is the i-th LDA topic, $\theta(i; d)$ is the probability of the i-th LDA topic given the document d and

$$\rho(w_i, T_j, \tau) = \begin{cases} 1 & \text{if } w_i \text{ belongs to the set of the most } \tau \text{ probable words of } T_j \\ 0 & \text{elsewhere.} \end{cases}$$

[1] We used the implementation and trained model from the Stanford Core NLP package (http://nlp.stanford.edu/software/corenlp.shtml)

[2] We used the implementation in the MALLAT package (http://mallet.cs.umass.edu/)

[3] We were limited by the dimension of our matrices to 2000 as our upper bound.

4 Crowdsourcing Techniques

We chose to use Amazon Mechanical Turk to crowdsource the document labelling because of the versatile and ease of use of the application programming interface (API) that is available to this service and also because it provided access to the largest number of workers without polluting trust metrics. Given the large size of the document set, it was deemed inefficient to crowdsource the labelling of documents by consensus. That is, to ask multiple turkers to label the same document and aggregate the votes to produce a single label. In more detail, the crowdsourcing approach has to balance the complexity and potential monetary cost of crowdsourcing against the quality of performance of the individual turkers. Indeed, the labelling task requires the reader to read every sentence and check whether a particular sentence could render the document relevant to a particular topic. The majority of documents were not more than a page long but a significant number were also more than two or three pages long. This would require significant effort from the reader to read every sentence and this task would, therefore, take considerably longer to complete. It was also obvious that, if a particular turker had to verify lots of documents, this process would quickly become boring and reduce the effectiveness of the turker.

To address the above issues, we developed a number of mechanisms to ensure:

1. The best turkers were chosen (i.e., those that can be trusted to perform tasks correctly).
2. The task of labelling a document was made easier for the turkers.
3. The turkers were encouraged to perform tasks quickly.

We describe each of these mechanisms in the following sections.

4.1 Trust Mechanism

Before we assigned paid tasks to workers, we ran a screening process to exclude those turkers that were likely to produce uninformative responses. In the screening process, workers had to assign the correct topics to 10 "Gold task" documents for which the true topic label was known. Workers also marked their decisions as confident or not confident. We then calculated the trust $T(k)$, i.e. the probability that a worker k was correct,

$$T(k) = ((n_{cc}^{(k)} + 1) + G(n_{cn}^{(k)} + 1) + (1 - G)(n_{in}^{(k)} + 1))/(N^{(k)} + 4) \tag{1}$$

where $n_{cc}^{(k)}$, is the number of tasks where turker k was correct and confident, $n_{cn}^{(k)}$ is the number of tasks where the turker was correct but not confident, $n_{in}^{(k)}$ is the number of tasks where the turker was incorrect and not confident. $N^{(k)}$ is the total number of tasks they actually performed. The variable G distributes the weight of a"not confident" answer between the possible responses. In preliminary tests, 70% of not confident responses were correct, so we set $G = 0.7$.

4.2 Derivation of Trust Mechanism

Trust in a turker is defined as the belief that the label assigned to a document by the turker is the correct label for that document and follows from a general definition of trust [7]. Formally, given an assigned label, r_w, by a turker, w, and some true label, t, for a document then the trustworthiness, $T(w)$, of that turker is,

$$T(w) = \sum_j p(r_w = j \mid t = j)p(t = j) \ .$$

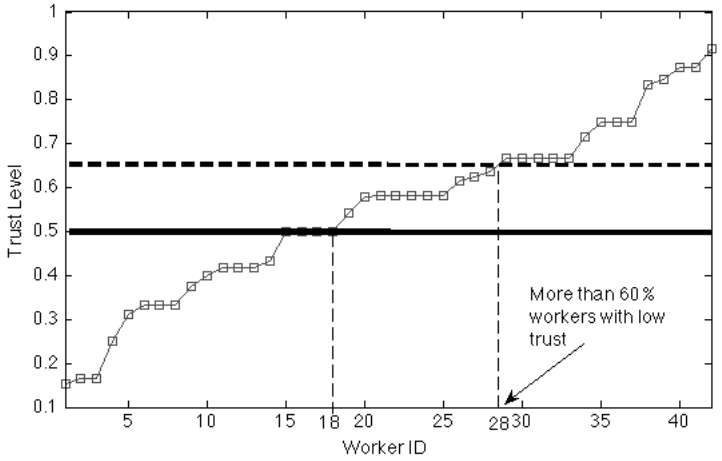

Figure 1: Trust levels of the 42 turkers that performed our gold tasks.

For ease of exposition we drop the index w and assume a uniform prior over the document classes, $p(t = j)$. We now expand this definition of trust to incorporate the intuition behind decisions labelled as"confident" or "not confident". A worker who is not confident about labels which turn out incorrect and confident in labels that turn out correct is more trustworthy than a worker who expresses high confidence in all her labels whether they turn out to be correct or not.

Each turker who labels a Gold standard document returns a report pair $[s, d]$: a document label, s, and a statement of confidence in her labelling d. We assume the turker either correctly labels the document, $s = j$, or incorrectly labels the document, $s = \neg j$. Furthermore, the turker is either confident in her label in which case $d = c$ or unconfident, in which case $d = n$. Now,

$$p(r = j \mid t = j) = \sum_{s \in \{j, \neg j\}} \sum_{d \in \{c,n\}} p(r = j \mid [s, d]) p([s, d] \mid t = j) \tag{2}$$

where $p(r = j \mid [s, d])$ is drawn from a beta distribution, i.e. r is binomial distributed, and $p([s, d] \mid t = j)$ is drawn from a Dirichlet, i.e [s, d] is multinomial distributed. The turker's response pair $[s, d]$ can take one of four values: *cc*, correct and confident labelling in which case $s = j$ and $d = c$; *ic*, incorrect and confident labelling with $s = \neg j$ and $d = c$; *cn*, correct but unconfident labelling in which case $s = j$ and $d = n$ and *in*, incorrect and unconfident labelling in which case $s = \neg j$ and $d = n$.

We assume uninformative priors so that $p(r = j \mid [s, d]) \sim \text{Beta}(1, 1)$ and $p([s, c] \mid t = j) \sim \text{Dir}(1, 1, 1, 1)$ for all values of j. We can thus learn the trust values from the data,

$$\hat{p}(r = j \mid t = j) = \sum_{s \in \{j, \neg j\}} \sum_{d \in \{c,n\}} \hat{p}(r = j \mid [s, d]) \hat{p}([s, d] \mid t = j) \tag{3}$$

$$= \sum_{s' \in \{c,i\}} \sum_{d \in \{c,n\}} G_{s'd} \frac{n_{s'd} + 1}{N + 4} \tag{4}$$

where N is the total number of gold standard test documents submitted to the turker and, for example, n_{in} is the number of documents with low confidence labellings which are incorrectly labelled. The probability $G_{cc} = \hat{p}(r = j \mid [s = j, c]) = 1$ is the turker's expected belief that the document should be labelled j given

that it has been labelled j and she has absolute confidence in that label. Alternatively, $G_{nc} = \hat{p}(r = \neg j \mid [s = j, c]) = 0$ is the turker's belief that the document should be labelled $\neg j$ given that she has labelled it j and she has absolute confidence in that label. $G_{cn} = \hat{p}(r = j \mid [s = j, n]) = G$ is the turker's belief that the document should be labelled j given that she has labelled it j and she has some doubt in that labelling. Finally, $G_{nn} = \hat{p}(r = \neg j \mid [s = j, n]) = (1 - G)$ is the turker's belief that the document should be labelled $\neg j$ given that she has labelled it j and she has some doubt in that labelling.

Thus, inserting G into (4) we arrive the expression in (1). The value for G can be inferred from data comprising the ground truth document labels, t, and worker response pairs $[s, d]$.

4.3 Hiring Process

After computing the trustworthiness for each turker using the mechanism reported above, we then only allowed those with trust greater than 0.5 to perform tasks in order to get more than 50% of the turkers to do tasks. This, however, resulted in very poor performance and the participation threshold was therefore raised to 0.65 and this resulted in only about 40% of the turkers being available to do tasks (see Figure 1). The classifier (to be described later) was then queried for those documents that were considered most important to be labelled. These documents were then chunked into parts not more than $20KB$ in size to make sure they were readable within 5 minutes. Then, depending on the document size, a payment level was determined for the labelling tasks required. We chose the following payment levels depending on document size,

- \$0.03 for tasks less than $5KB$
- \$0.05 for tasks between $5KB$ and $15KB$
- \$0.08 for tasks beyond $15KB$.

Once the tasks were uploaded into AMT, the trusted turkers were notified through the AMT messaging service to start working on the tasks. The trust mechanism was good at filtering turkers and the labelling task was completed to a high quality. However, we noticed that those turkers performing large numbers of tasks generally under performed as they completed lots of tasks, possibly due to boredom.

4.4 HIT Interface

The principle we adopted for our *Human Interface Technology* (HIT) interface was that the task should be straightforward for the user and should include checks to make sure the task was completed to a high level of quality. Therefore, the labelling task required the turker,

1. To read the document which was divided into three sections this is to break down the reading task into more manageable segments and avoid the reader feeling overwhelmed.
2. Paste a key sentence from each section into the page this is meant to check whether the reader actually read the section.
3. Choose a topic for the document as well as assign a confidence level to the label she gave.

5 Document-Topic Classification

We applied the Independent Bayesian Classifier Combination (IBCC) algorithm to automatically label documents using the responses from the turkers. We also compared it to a more traditional two-stage Bayes classifier. Both classifiers were used to infer the probability of relevance of each document to

each topic from the document features identified using the NLP algorithms described in Section 3 and crowdsourced labels.

5.1 IBCC: One-stage Classifier

Members of the crowd can be seen as imperfect *base classifiers*, where each individual has their own probability of producing a particular label. Features occur with different probabilities given each topic, so can also be seen as *base classifiers* ranging from highly indicative of a topic to completely uninformative. Our method learns the probabilities of features and crowd responses, simultaneously combining them to infer topic probabilities. This differs from traditional classification, which has separate training and prediction phases. The one-stage classifier is a *semi-supervised* approach that allows latent structure in the unlabelled documents to influence the results, which is particularly effective when there is limited training data.

The graphical model for IBCC [8] is shown in Figure 2. For a set of documents indexed from 1 to N, the ith document has topic $t_i = 1...J$ that we wish to infer, where J is the number of topics. We assume t_i is generated from a multinomial distribution with class probabilities $\boldsymbol{\kappa} : p(t_i = j|\boldsymbol{\kappa}) = \kappa_j$. For the challenge problem we have assumed that the topics are mutually exclusive based on our understanding of the topics, and that there also exists a "none of the above" topic for documents of unknown topic. The total number of base classifiers (features and turkers) is K and the set of values assigned by base classifiers to documents is $\boldsymbol{c}$. A turker k may produce a label $c_i^{(k)}$ with values $l = 1..L$, where L is the number of topics. A feature k can have values $c_i^{(k)}$ of either 0 or 1. The value $c_i^{(k)}$ is assumed to be generated from a multinomial distribution dependent on the topic of document i, with parameters $\boldsymbol{\pi}_j^{(k)} : p(c_i^{(k)} = l|t_i = j, \boldsymbol{\pi}_j^{(k)}) = \pi_{jl}^{(k)}$. IBCC assumes conditional independence between base classifiers. Parameters $\boldsymbol{\pi}_j^{(k)}$ and $\boldsymbol{\kappa}$ have Dirichlet prior distributions with hyper-parameters $\boldsymbol{\alpha}_{0,j}^{(k)} = [\alpha_{0,j1}^{(k)}, ..., \alpha_{0,jL}^{(k)}]$ and $\boldsymbol{\nu} = [\nu_{0,1}, ...\nu_{0,J}]$ respectively. We refer to the set of $\boldsymbol{\pi}_j^{(k)}$ for all base classifiers and all classes as $\Pi = \left\{\boldsymbol{\pi}_j^{(k)}|j = 1...J, k = 1...K\right\}$ and also denote the hyper-parameters $\boldsymbol{A}_0 = \left\{\boldsymbol{\alpha}_0^{(k)}{}_j|j = 1...J, k = 1...K\right\}$. The joint distribution over all variables for the IBCC model is

$$p(\boldsymbol{\kappa}, \Pi, \boldsymbol{t}, \boldsymbol{c}|\boldsymbol{A}_0, \boldsymbol{\nu}) = \prod_{i=1}^{N}\{\kappa_{t_i} \prod_{k=1}^{K} \pi^{(k)}_{t_i, c_i^{(k)}}\} p(\boldsymbol{\kappa}|\boldsymbol{\nu}) p(\Pi|\boldsymbol{A}_0), \tag{5}$$

A key feature of IBCC is that $\boldsymbol{\pi}^{(k)}$ represents a *confusion matrix* that quantifies the reliability of *each* crowd member and feature. The prior distribution over each $\boldsymbol{\pi}^{(k)}$ is specified through the hyper-parameters $\boldsymbol{A}_0$, which can be regarded as pseudo-counts of prior observations of base classifier outputs. Through our choice of these hyper-parameters, we assume that turkers are likely to be informative but not perfect, allowing us to perform inference without observing true topic labels directly.

5.1.1 Inference over IBCC using Variational Bayes

The unknown variables $\boldsymbol{t}$, Π, and $\boldsymbol{\kappa}$ can be estimated using maximum a posteriori (MAP) estimation [9] or inferred in a Bayesian manner [8] using Gibbs Sampling [10]. Here we applied a principled approximate Bayesian method, *variational Bayes* (VB) [11], as this converges rapidly to a good approximate solution [12]. VB can be seen as a Bayesian generalisation of the *Expectation-Maximisation* (EM) algorithm [13].

For the variational treatment of IBCC, *VB-IBCC*, we assume an approximate form of the posterior distribution over the unknown variables:

$$q(\boldsymbol{\kappa}, \boldsymbol{t}, \Pi) \quad = \quad q(\boldsymbol{t}) q(\boldsymbol{\kappa}, \Pi) \tag{6}$$

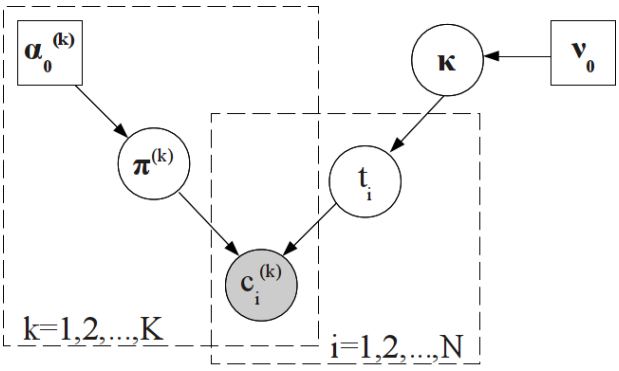

Figure 2: Graphical Model for IBCC. The shaded node represents observed values, circular nodes are variables with a distribution and square nodes are variables instantiated with point values.

The VB algorithm runs by first initialising the variables $\mathbb{E}[\ln \pi_{jl}^{(k)}]$ and $\mathbb{E}[\ln \kappa_j]$ from their prior distributions. We then iterate over a two-stage procedure, updating each of the factors $q(\boldsymbol{t})$ and $q(\boldsymbol{\kappa}, \Pi)$ to their optimal values in turn, using the current expectations over parameters not in that factor.

The optimal factor $q^*(\boldsymbol{t})$ for the topics is defined by taking the joint distribution in (5) and absorbing terms not dependent on $\boldsymbol{t}$ into the normalisation constant:

$$\ln q^*(\boldsymbol{t}) \quad = \quad \mathbb{E}_{\boldsymbol{\kappa},\Pi}[\ln p(\boldsymbol{\kappa}, \boldsymbol{t}, \Pi, \boldsymbol{c})] + \text{const.} \tag{7}$$

This factorises into independent data points:

$$\ln \rho_{ij} = \mathbb{E}_{\kappa_j, \boldsymbol{\pi_j}}[\ln p(\kappa_j, t_i, \boldsymbol{\pi_j}, \boldsymbol{c})] = \mathbb{E}_{\boldsymbol{\kappa}}[\ln \kappa_j] + \sum_{k=1}^{K} \sum_{l=1}^{L} p(c_i^{(k)} = l) \mathbb{E}_{\boldsymbol{\pi}_j}[\ln \pi_{j,c_i^{(k)}}^{(k)}]. \tag{8}$$

For a base classifier that is a turker, the value of $c_i^{(k)}$ is known and $p(c_i^{(k)} = l)$ is either 0 or 1. However, for features we receive probabilities rather than discrete outputs, hence the term $p(c_i^{(k)} = l)$ in the equation above. We then obtain the approximate probability of a topic, which also gives its expected value:

$$q^*(t_i = j) = \mathbb{E}_{\boldsymbol{t}}[t_i = j] = \frac{\rho_{ij}}{\sum_{\iota=1}^{J} \rho_{i\iota}} \,. \tag{9}$$

For the parameters of the model we have the following optimal factor. Terms involving $\boldsymbol{\kappa}$ and terms involving each confusion matrix in Π are independent, so we can factorise $q^*(\boldsymbol{\kappa}, \boldsymbol{\pi})$ further into

$$q^*(\boldsymbol{\kappa}, \Pi) = q^*(\boldsymbol{\kappa}) \prod_{k=1}^{K} \prod_{j=1}^{J} q^* \left(\boldsymbol{\pi}_j^{(k)} \right).$$

In IBCC we assume a Dirichlet prior for $\boldsymbol{\kappa}$, which gives us a Dirichlet posterior for the optimal factor

$$q^*(\boldsymbol{\kappa}) \propto \text{Dir}(\boldsymbol{\kappa} | \nu_1, ..., \nu_J) \qquad\qquad \nu_j = \nu_{0,j} + N_j \tag{10}$$

where $\boldsymbol{\nu}$ is updated by adding the data counts to the prior counts ν_0 and $N_j = \sum_{i=1}^{N} \mathbb{E}_{\boldsymbol{t}}[t_i = j]$ is the expected number of documents of each topic. The expectation of $\ln \boldsymbol{\kappa}$ required to update (8) is therefore:

$$\mathbb{E}[\ln \kappa_j] = \Psi(\nu_j) - \Psi\left(\sum_{\iota=1}^{J} \nu_\iota\right) \tag{11}$$

where Ψ is the standard digamma function [14]. For the confusion matrices $\boldsymbol{\pi}_j^{(k)}$, the priors are also Dirichlet distributions, giving us a posterior Dirichlet distribution of the form

$$q^*\left(\boldsymbol{\pi}_j^{(k)}\right) = \text{Dir}\left(\boldsymbol{\pi}_j^{(k)} | \alpha_{j1}^{(k)}, ..., \alpha_{jL}^{(k)}\right) , \qquad \alpha_{jl}^{(k)} = \alpha_{0,jl}^{(k)} + N_{jl}^{(k)}. \tag{12}$$

where $\boldsymbol{\alpha}_j^{(k)}$ is updated by adding data counts to prior counts $\alpha_{0,j}^{(k)}$, and $N_{jl}^{(k)}$ is defined as

$$N_{jl}^{(k)} = \sum_{i=1}^{N} p(c_i^{(k)} = l) \mathbb{E}_{\boldsymbol{t}}[t_i = j] \tag{13}$$

i.e. the number of times that classifier k has assigned value l to a document of class j. The expectation required for (8) is given by

$$\mathbb{E}\left[\ln \pi_{jl}^{(k)}\right] = \Psi\left(\alpha_{jl}^{(k)}\right) - \Psi\left(\sum_{m=1}^{L} \alpha_{jm}^{(k)}\right). \tag{14}$$

5.2 Two-Stage Bayes Classifier

The traditional approach to classification is two-stage. In the first stage labelled *training* data is used to build models for the probability of each class and the probability of features conditioned on each class. The second stage uses this model to infer class probabilities of unlabelled documents.

We assume that the true label t is generated from a multi-nomial distribution with probability κ: $p(t = j \mid \kappa) = \kappa_j$. We also assume that the observed features $\mathbf{c}$ are binary so that $c_i = 1$ indicates the presence of feature i in a document and $c_i = 0$ indicates its absence. Thus, the observed features are generated from binomial distributions dependent on the class of the true label with parameters β: $p(c_i = b \mid t = j, \beta) = \beta_{jib}$. The parameters β have Beta prior distributions with hyperparameters μ and κ have Dirichlet prior distributions ν. Values for μ and ν are inferred from the training data as we will demonstrate. Given some test (or unlabelled) document the distribution over the true label t^* of this document given the features $\mathbf{c}^*$ for that document is,

$$\begin{aligned}
p(t^* = j \mid \mathbf{c}^*, \mu, \nu) &= \int_\pi \int_\kappa p(\mathbf{c}^* \mid t^* = j, \beta) p(t^* = j \mid \kappa) Be(\beta \mid \mu) Dir(\kappa \mid \nu) d\beta \, d\kappa \\
&= \left[\int_\beta p(\mathbf{c}^* \mid t^* = j, \beta) Be(\beta \mid \mu) d\pi\right] \left[\int_\kappa p(t^* = j \mid \kappa) Dir(\kappa \mid \nu) d\kappa\right] \\
&= \prod_i \left[\int_{\beta_{jib}} \beta_{jib} Be(\beta \mid \mu) d\beta\right] \left[\int_\kappa \kappa_j Dir(\kappa \mid \nu) d\kappa\right]
\end{aligned}$$

In the training stage, the labelled data is used to infer the expected likelihoods β and class probabilities κ. For any variable θ and multi-variate ϕ, $\int \theta Be(\theta \mid \cdot) d\theta$ and $\int \phi Dir(\phi \mid \cdot) d\phi$ are the means of a Beta and

Dirichlet distribution, respectively. Assuming uninformative priors for β and κ and training data $\{t, c\}$, the posterior means are:

$$\int \beta_{jib} Be(\beta \mid \mu(t,c)) d\beta = \frac{N_{jb}^{(i)} + 1}{\sum_{d=0}^{1}(N_{jd}^{(i)} + 1)}, \qquad \int \kappa_j Dir(\kappa \mid \nu(t,c)) d\kappa = \frac{N_j + 1}{\sum_k (N_k + 1)}$$

where $N_{jb}^{(i)}$ is the number of documents labelled j that contain feature $c_i = b$ in the training data, $\{t, c\}$, and N_j is the number of documents labelled j.

Stage two is the prediction phase and for test features $\mathbf{c}^*$ from a document the probability that the document is class j is,

$$p(t^* = j \mid \mathbf{c}^*, \mu, \nu) = \prod_i \left[\frac{N_{j\ c_i}^{(i)} + 1}{\sum_{d=0}^{1}(N_{jd}^{(i)} + 1)} \right] \left[\frac{N_j + 1}{\sum_k (N_k + 1)} \right].$$

In summary, we can determine the posterior document class probability, $p(t^* = j \mid \mathbf{c}^*, \mu, \nu)$ exactly and efficiently using only the sufficient statistics of Beta and Dirichlet distributions.

6 Selecting Documents to Pass to the Crowd

We aimed to maximize the information gained from crowdsourcing by choosing documents with particular features, using approximate methods to reduce computational expense. Each time we receive a set of labels from the crowd, we must select n more documents to label. Documents may be labelled multiple times as the turkers are not reliable. Optimally, we would select the set of n documents that maximise the expected information gain over the set of all documents. However, to calculate the expected information gain, we must re-run the classifier for each of the possible labels we could receive for every combination of n documents. This becomes prohibitively expensive with more than a few documents, so we turn to approximate methods. Learning the label of a document affects the counts N_j and $N_{jl}^{(k)}$ in (10) and (12). First, we consider the information gain $I(t_i; c_x^{(k)})$ about the topic of a single document:

$$\mathbb{E}[I(t_i; c_x^{(k)})] \leq \sum_{j=1}^{J} -p(t_i = j) \ln p(t_i = j) \tag{15}$$

If $\mathbb{E}[I(t_i; c_x^{(k)})]$ is low, we assume that the overall information gain for all documents $\mathbb{E}[I(\boldsymbol{t}; c_x^{(k)})]$ is unlikely to be high as we expect to make only small updates to the counts N_j and $N_{jl}^{(k)}$ given the new label. Therefore we make the document selection problem tractable by considering n_{filter} documents with highest $\mathbb{E}[I(t_i; c_x^{(k)})] > \theta$. The assumption requires that the documents are evenly dispersed in feature space so that we do not include only outliers. We use a further approximation to (15) and identify most of the documents with high entropy by finding those with the lowest probability for their most likely class.

Information gain from labelling a document i is higher when we have few previous examples of the features of i. Therefore, we can choose documents with different sets of features to explore the feature space fully. To select maximally different documents, we apply k-means clustering to group the documents according to their features, then select a document from each cluster with approximate highest entropy. This way, each document is selected to reduce entropy significantly for all documents in its cluster. If some clusters are much larger than others, this method can be sub-optimal as the information gain for a small

cluster may be less than from labelling a second document in a larger cluster. However, in this application, features are generated to give a reasonable distribution across documents. Although this method is highly approximate, we found a noticeable improvement over random selection. To summarise, the steps of the algorithm are as follows.

1. Receive a set of labels from the crowd.
2. Run the classifier over current set of data to obtain probabilities of topics for all documents.
3. Approximate $I(t_c; c_x^{(k)})$ upper bound using the probabilities of the most probable class for each document.
4. Select the n_{filter} documents with the highest approximate information gain.
5. Cluster the n_{filter} documents by features using k-means into $n_{clusters}$ clusters
6. From each cluster, select the document with the highest approximate information gain.
7. Send the selected documents to the crowd and repeat the process.

7 Results

Fourteen out for forty two turkers had a trust value which exceeded the cut-off of 0.65 and the Gold standard test cost $20 dollars. The trustworthy turkers classified 2500 documents (i.e. 16% of the corpus) within 4 hours at a cost of between $0.03 and $0.08 per document. The total spend was $210. However, less than five turkers completed most tasks and one strategic turker was blocked. Since workers got bored over time we issued a bonus of $3 to complete the task to a deadline.

Using the crowdsourced labelled documents as training data we compared IBCC to the traditional two-stage Bayes classifier on the TREC 2012 crowdsourcing track dataset, which contained 18260 topic-document pairs. Within each pair, the classifiers calculated the relevance of the document to the topic. IBCC outperformed the alternative classifier. The area under curve (AUC) of the Receiver Operating Characteristic for IBCC was 0.806 against 0.774 for the two stage classifier. The AUC for each individual topic and for each classifier is shown in Figure 3.

8 Discussion and Conclusions

We concluded that Amazon Turk may not be the best approach to crowdsourcing the labelling of documents with a small set of topics when the corpus contains a large number of irrelevant documents. In this context, there is a need to maintain attention levels, for example through *Gamification* (or GWAP-based, Games With A Purpose) of the task, whereby the task is re-written as a game.

We also concluded that the IBCC classifier was about 4% more accurate at inferring the class of each document than the traditional two-stage classifier. There are two reasons for this difference in performance. First, the two-stage approach uses only the labelled document data to infer the classifier model parameters. However, the IBCC uses the test data (i.e. the unlabelled data) as well as the labelled data to infer the classifier model parameters. Thus, although the test data is unlabelled the features can provide a strong indication of the class membership. Consequently, this information provided by the test data can be used along with the labelled data to provide a more accurate model of the class prior and feature-class likelihood than would be possible using the two-stage approach.

The second advantage of IBCC is that it models the unreliability of crowdsourced labels compared to the ground truth. In contrast, the class labels referred to within the two-stage model are the class labels

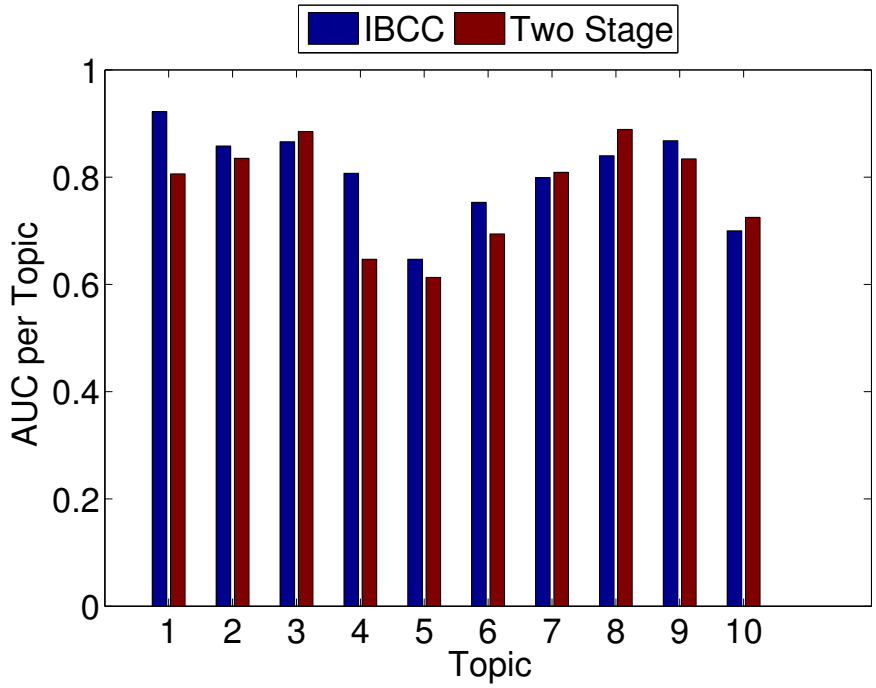

Figure 3: AUC for each of the ten topics within the TREC 2012 challenge problem. Shown are the AUCS for the IBCC and the two-stage approach.

provided by the turker. Thus, the feature-class likelihood is interpreted as the probability of a feature occurring in a document given the turker has labelled the document as belonging to a class. The two-stage model, as formulated in this chapter, cannot make explicit reference to the ground-truth document labels as, in the TREC challenge data set, there is no training data labelled with the ground truth. Consequently, for our two-stage classifier to provide accurate predictions of the document labels we must ensure that the crowd's responses are statistically similar to the ground truth dictated by the NIST committee as we have no opportunity to filter out untrustworthy turkers using the two-stage approach. Thus, the initial filtering of trustworthy turkers described in this chapter is a critical component of the two-stage approach.

This paper describes our approach to solving TREC crowdsourcing challenge of 2012, and demonstrates each of the algorithmic components required for a successful system, demonstrated by our placement as 2nd out of 17 participating groups in terms of AUC curves. We intend to revisit our solution and produce an integrated design for the components of the system based around the IBCC model. For example, using IBCC to derive the trust values for workers. The system also collected confidence responses, and future work could consider how to exploit these with extensions to IBCC.

We aim to apply our approach to more challenging hard/soft data fusion problems, again in collaboration with Southampton University. Specifically, we will apply these algorithms to the Ushahidi data set. This dataset is particularly relevant to Orchid it is an example of a scenario in disaster recovery which is one of Orchids application domains. Examples of soft information in Ushahidi setting are human translations of free-text to constrained text (i.e. feature-spaces) and examples of hard data are the times and GPS locations of events.

Acknowledgements

We gratefully acknowledge funding from the UK Research Council EPSRC for project ORCHID, grant EP/I011587/1.

References

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

[2] Fabian M. Suchanek. Leila: Learning to extract information by linguistic analysis. In *In Workshop on Ontology Population at ACL/COLING*, pages 18–25, 2006.

[3] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[4] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[5] Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Information System*, 20(4):357–389, October 2002.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal Machine Learning Result*, 3:993–1022, March 2003.

[7] S. Reece, A. Rogers, S. Roberts, and N. R. Jennings. Rumours and reputation: Evaluating Multi-Dimensional Trust within a Decentralised Reputation System. In *In Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems. In*. press, 2007.

[8] Z. Ghahramani and H. C. Kim. Bayesian classifier combination. *Gatsby Computational Neuroscience Unit Technical Report GCNU-T., London, UK:*, 2003.

[9] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *J. Mach. Learn. Res.*, 11:1297?1322, August 2010.

[10] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(6):721 –741, November 1984.

[11] H. Attias. *Advances in Neural Information Processing Systems 12*, chapter A Variational Bayesian Framework for Graphical Models, pages 209–215. 2000.

[12] C. M. Bishop. *Pattern recognition and machine learning*. Information Science and Statistics. Springer Science+Business Media, LLC, 4 edition, 2006.

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, January 1977.

[14] M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*. Dover Publications, June 1965.