Abstract

Current methods for sequence tagging, a core task in NLP, are data hungry, which motivates the use of crowdsourcing as a cheap way to obtain labelled data. However, annotators are often unreliable and current aggregation methods cannot capture common types of span annotation errors. To address this, we propose a Bayesian method for aggregating sequence tags that reduces errors by modelling sequential dependencies between the annotations as well as the ground-truth labels. By taking a Bayesian approach, we account for uncertainty in the model due to both annotator errors and the lack of data for modelling annotators who complete few tasks. We evaluate our model on crowdsourced data for named entity recognition, information extraction and argument mining, showing that our sequential model outperforms the previous state of the art. We also find that our approach can reduce crowdsourcing costs through more effective active learning, as it better captures uncertainty in the sequence labels when there are few annotations.

1 Introduction

Current methods for sequence tagging, a core task in NLP, are data hungry, which motivates the use of crowdsourcing as a cheap way to obtain labelled data. However, annotators are often unreliable and current aggregation methods cannot capture common types of span annotation errors. To address this, we propose a Bayesian method for aggregating sequence tags that reduces errors by modelling sequential dependencies between the annotations as well as the ground-truth labels. By taking a Bayesian approach, we account for uncertainty in the model due to both annotator errors and the lack of data for modelling annotators who complete few tasks. We evaluate our model on crowdsourced data for named entity recognition, information extraction and argument mining, showing that our sequential model outperforms the previous state of the art. We also find that our approach can reduce crowdsourcing costs through more effective active learning, as it better captures uncertainty in the sequence labels when there are few annotations.

When learning from noisy or small datasets, commonly-used methods based on maximum likelihood estimation may produce over-confident predictions (Xiong et al., 2011; Srivastava et al., 2014). In contrast, Bayesian inference accounts for model uncertainty when making predictions. Unlike alternative methods that optimize the values for model parameters, Bayesian inference integrates over all possible values of a parameter, weighted by a prior distribution that captures background knowledge. The resulting posterior probabilities improve downstream decision making as they include the probability of errors due to a lack of knowledge. For example, during active learning, posterior probabilities assist with selecting the most informative data points (Settles, 2010).

In this paper, we develop Bayesian sequence combination (BSC), building on prior work that has demonstrated the advantages of Bayesian inference for aggregating unreliable classifications (Kim and Ghahramani, 2012; Simpson et al., 2013; Felt et al., 2016; Paun et al., 2018). BSC is the first fully-Bayesian method for aggregating sequence labels from multiple annotators. As a core component of BSC, we also introduce the sequential confusion matrix (seq), a probabilistic model
of annotator noise and bias, which goes beyond previous work by modelling sequential dependencies between annotators’ labels. Further contributions include a theoretical comparison of the probabilistic models of annotator noise and bias, and an empirical evaluation on three sequence labelling tasks, in which BSC with seq consistently outperforms the previous state of the art. We make all of our code and data freely available\(^1\).

2 Related Work

Sheshadri and Lease (2013) benchmarked several aggregation models for non-sequential classifications, obtaining the most consistent performance from that of Raykar et al. (2010), who model the reliability of individual annotators using probabilistic confusion matrices, as proposed by Dawid and Skene (1979). Simpson et al. (2013) showed that a Bayesian variant of Dawid and Skene’s model can outperform maximum likelihood approaches and simple heuristics when combining crowds of image annotators. This Bayesian variant, independent Bayesian classifier combination (IBCC) (Kim and Ghahramani, 2012), was originally used to combine ensembles of automated classifiers rather than human annotators. While traditional ensemble methods such as boosting focus on how to generate new classifiers (Dietterich, 2000), IBCC is concerned with modelling the reliability of each classifier in a given set of classifiers. To reduce the number of parameters in multi-class problems, Hovy et al. (2013) proposed MACE, and showed that it performed better under a Bayesian treatment on NLP tasks. Paun et al. (2018) further illustrated the advantages of Bayesian models of annotator ability on NLP classification tasks with different levels of annotation sparsity and noise.

We expand this previous work by detailing the relationships between several annotator models and extending them to sequential classification. Here we focus on the core annotator representation, rather than extensions for clustering annotators (Venanzi et al., 2014; Moreno et al., 2015), modeling their dynamics (Simpson et al., 2013), adapting to task difficulty (Whitehill et al., 2009; Bachrach et al., 2012), or time spent by annotators (Venanzi et al., 2016).

Methods for aggregating sequence labels in-clude CRF-MA (Rodrigues et al., 2014), a CRF-based model that assumes only one annotator is correct for any given label. Recently, Nguyen et al. (2017) proposed a hidden Markov model (HMM) approach that outperformed CRF-MA, called HMM-crowd. Both CRF-MA and HMM-crowd use simpler annotator models than Dawid and Skene (1979) that do not capture the effect of sequential dependencies on annotator reliability. Neither CRF-MA nor HMM-crowd use a fully Bayesian approach, which has been shown to be more effective for handling uncertainty due to noise in crowdsourced data for non-sequential classification (Kim and Ghahramani, 2012; Simpson et al., 2013; Venanzi et al., 2014; Moreno et al., 2015). In this paper, we develop a sequential annotator model and an approximate Bayesian method for aggregating sequence labels.

3 Modeling Sequential Annotators

When combining multiple annotators with varying skill levels, we can improve performance by modelling their individual noise and bias using a probabilistic model. Here, we describe several models that do not consider dependencies between annotations in a sequence, before defining seq, a new extension that captures sequential dependencies. Probabilistic annotator models each define a different function, \( A \), for the likelihood that the annotator chooses label \( c_\tau \) given the true label \( t_\tau \), for the \( \tau \)th token in a sequence.

Accuracy model (acc): the basis of several previous methods (Donmez et al., 2010; Rodrigues et al., 2013), acc uses a single parameter for each annotator’s accuracy, \( \pi \):

\[
A = p(c_\tau = i|t_\tau = j, \pi) = \begin{cases} \pi & \text{where } i = j \\ \frac{1-\pi}{J-1} & \text{otherwise} \end{cases},
\]

(1)

where \( J \) is the number of classes. This may be unsuitable when one class label dominates the data, since a spammer who always selects the most common label will nonetheless have a high \( \pi \).

Spamming model (spam): proposed as part of MACE (Hovy et al., 2013), this model also assumes constant accuracy, \( \pi \), but that when an annotator is incorrect, they label according to a spamming distribution, \( \xi \), that is independent of

\[http://github.com/ukplab/arxiv2018-bayesian-ensembles\]
the true label, \( t_{\tau} \).

\[
A = p(c_{\tau} = i | t_{\tau} = j, \pi, \xi) = \begin{cases} 
\pi + (1 - \pi)\xi_j & \text{where } i = j \\
(1 - \pi)\xi_j & \text{otherwise}
\end{cases}.
\]  

This addresses the case where spammers choose the dominant label but does not explicitly model different error rates in each class. For example, if an annotator is better at detecting type ‘x’ spans than type ‘y’, or if they frequently miss the first token in a span, thereby labelling the start of a span as ‘O’ when the true label is ‘B-x’, this would not be explicitly modelled by \( spam \).

**Confusion vector (CV):** this approach learns a separate accuracy for each class label (Nguyen et al., 2017) using parameter vector, \( \pi \), of size \( J \):

\[
A = p(c_{\tau} = i | t_{\tau} = j, \pi) = \begin{cases} 
\pi_j & \text{where } i = j \\
\frac{1 - \pi_j}{1 - \sum_{j=1}^{J} \pi_j} & \text{otherwise}
\end{cases}.
\]  

This model does not capture spamming patterns where one of the incorrect labels has a much higher likelihood than the others.

**Confusion matrix (CM)** (Dawid and Skene, 1979): this model can be seen as an expansion of the confusion vector so that \( \pi \) becomes a \( J \times J \) matrix with values given by:

\[
A = p(c_{\tau} = i | t_{\tau} = j, \pi) = \pi_{j,i}.
\]  

This requires a larger number of parameters, \( J^2 \), compared to the \( J + 1 \) parameters of MACE or \( J \) parameters of the confusion vector. Like \( spam \), CM can model spammers who frequently chose one label regardless of the ground truth, but also models different error rates and biases for each class. However, CM ignores dependencies between annotations in a sequence, such as the fact that an ‘I’ cannot immediately follow an ‘O’.

**Sequential Confusion Matrix (seq):** we introduce a new extension to the confusion matrix to model the dependency of each label in a sequence on its predecessor, giving the following likelihood:

\[
A = p(c_{\tau} = i | c_{\tau-1} = \iota, t_{\tau} = j, \pi) = \pi_{j,i,\iota},
\]  

where \( \pi \) is now three-dimensional with size \( J \times J \). In the case of disallowed transitions, e.g. from \( c_{\tau-1} = ‘O’ \) to \( c_{\tau} = ‘I’ \), the value \( \pi_{j,c_{\tau-1},c_{\tau}} \approx 0 \), \( \forall j \) is fixed a priori. The sequential model can capture phenomena such as a tendency toward overly long sequences, by learning that I is more likely to follow another I, so that \( \pi_{O,I,I} > \pi_{O,I,O} \).

A tendency to split spans by inserting ‘B’ in place of ‘I’ can be modelled by increasing the value of \( \pi_{I,B,I} \) without affecting \( \pi_{I,B,B} \) and \( \pi_{I,O,B} \).

The annotator models presented in this section include the most widespread models for NLP annotation tasks, and can be seen as extensions of one another. The choice of annotator model for a particular annotator depends on the developer’s understanding of the annotation task: if the annotations have sequential dependencies, this suggests the seq model; for non-sequential classifications CM may be effective with small (\( \leq 5 \)) numbers of classes; \( spam \) may be more suitable if there are many classes, as the number of parameters to learn is low. However, there is also a trade-off between the expressiveness of the model and the number of parameters that must be learned. Simpler models with fewer parameters may be effective if there are only small numbers of annotations from each annotator. The next section shows how these annotator models can be used as components of a complete model for aggregating sequential annotations.

### 4 A Generative Model for Bayesian Sequence Combination

To construct a generative model for Bayesian sequence combination (BSC), we first define a hidden Markov model (HMM) with states \( t_{n,\tau} \) and observations \( x_{n,\tau} \) using categorical distributions:

\[
t_{n,\tau} \sim \text{Cat}(T_{n,\tau-1}),
\]

\[
x_{n,\tau} \sim \text{Cat}(\rho_{t_{n,\tau}}),
\]

where \( T_{j} \) is a row of a transition matrix \( T \), and \( \rho_{j} \) is a vector of observation likelihoods for state \( j \). For text tagging, \( n \) indicates a document and \( \tau \) a token index, while each state \( t_{n,\tau} \) is a true sequence label and \( x_{n,\tau} \) is a token. To provide a Bayesian treatment, we assume that \( T_{j} \) and \( \rho_{j} \) have Dirichlet distribution priors as follows:

\[
T_{j} \sim \text{Dir}(\gamma_{j}), \quad \rho_{j} \sim \text{Dir}(\kappa_{j}),
\]

where \( \gamma_{j} \) and \( \kappa_{j} \) are hyperparameters.

Next, we assume one of the annotator models described in Section 3 for each of \( K \) annotators. Selecting an annotator model is a design choice, and all can be coupled with the Bayesian HMM above to form a complete BSC model. In
our experiments in Section 6, we compare different choices of annotator model as components of BSC. All the parameters of these annotator models are probabilities, so to provide a Bayesian treatment, we assume that they have Dirichlet priors. For annotator $k$’s annotator model, we refer to the hyperparameters of its Dirichlet prior as $\alpha^{(k)}$. The annotator model defines a categorical likelihood over each annotation, $c^{(k)}_{n,\tau}$.

\[
c^{(k)}_{n,\tau} \sim \text{Cat}(A^{(k)}(t_{n,\tau}, 1, c^{(k)}_{n,\tau-1}), \ldots, A^{(k)}(t_{n,\tau}, J, c^{(k)}_{n,\tau-1})). \tag{9}
\]

The annotators are assumed to be conditionally independent of one another given the true labels, $t$, which means that their errors are assumed to be uncorrelated. This is a strong assumption when considering that the annotators have to make their decisions based on the same input data. However, in practice, dependencies do not usually cause the most probable label to change (Zhang, 2004), hence the performance of classifier combination methods is only slightly degraded, while avoiding the complexity of modelling dependencies between annotators (Kim and Ghahramani, 2012).

**Joint distribution**: the complete model can be represented by the joint distribution, given by:

\[
p(t, A, T, \rho | c, x, \alpha, \gamma, \kappa) \approx \prod_{k=1}^{K} \frac{N}{\rho^{(k)}} \prod_{n=1}^{N} \prod_{\tau=1}^{\rho_{n,\tau}} p(x_{n,\tau} | A^{(k)}(t_{n,\tau}, \rho_{n,\tau})) \prod_{j=1}^{J} p(T_j | \gamma_j) p(\rho_j | \kappa_j) \tag{10}
\]

where $c$ is the set of annotations for all documents from all annotators, $t$ is the set of all sequence labels for all documents, $N$ is the number of documents, $L_n$ is the length of the $n$th document, $J$ is the number of classes, $x$ is the set of all word sequences for all documents and $\rho, \gamma$ and $\kappa$ are the sets of parameters for all label classes.

## 5 Inference using Variational Bayes

Given a set of annotations, $c$, we obtain a posterior distribution over sequence labels, $t$, using **variational Bayes (VB)** (Attias, 2000). Unlike maximum likelihood methods such as standard expectation maximization (EM), VB considers prior distributions and accounts for parameter uncertainty due to noisy or sparse data. In comparison to other Bayesian approaches such as Markov chain Monte Carlo (MCMC), VB is often faster, readily allows incremental learning, and provides easier ways to determine convergence (Bishop, 2006). It has been successfully applied to a wide range of methods, including being used as the standard learning procedure for LDA (Blei et al., 2003), and to combining non-sequential crowdsourced classifications (Simpson et al., 2013).

The trade-off is that we must approximate the posterior distribution with an approximation that factorises between subsets of latent variables:

\[
p(t, A, T, \rho | c, x, \alpha, \gamma, \kappa) \approx \prod_{k=1}^{K} q(A^{(k)}) \prod_{j=1}^{J} \{q(T_j)q(\rho_j)\} \prod_{n=1}^{N} q(t_n). \tag{12}
\]

VB performs approximate inference by updating each variational factor, $q(z)$, in turn, optimising the approximate posterior distribution until it converges. Details of the theory are beyond the scope of this paper, but are explained by Bishop (2006). The VB algorithm is described in Algorithm 1, making use of update equations for the variational factors given below.

**Input**: Annotations $c$, tokens $x$

1. Compute initial values of $\mathbb{E} \ln A^{(k)}, \forall k$
2. Compute initial values of $\mathbb{E} \ln \rho_j, \forall j, \mathbb{E} \ln T_j, \forall j$ from their prior distributions.
3. While not converged $(r_{n,\tau,j}, \forall n, \forall \tau, \forall j)$ do
4. Update $r_{j,n,\tau}, s_{t_{j,n-1,\tau},t_{j,n,\tau}}$, $\forall j, \forall \tau, \forall n, \forall t$, using forward-backward algorithm given $x, c, \mathbb{E} \ln T_j, \forall j$,
5. Update $\mathbb{E} \ln \rho_j, \forall j$ given $x, r_{j,n,\tau}$.

**Output**: Label posteriors, $r_{n,\tau,j}, \forall n, \forall \tau, \forall j$, most probable sequence of labels, $\hat{t}_n, \forall n$ using Viterbi algorithm

**Algorithm 1**: The VB algorithm for BSC.

We compute the posterior probability of each true token label, $r_{n,\tau,j} = \mathbb{E}[p(t_{n,\tau} = j|c)]$, and
of each label transition, \( s_{n, \tau, j, t} = \mathbb{E}[p(t_n, \tau = j, t_{n, \tau} = \iota | \mathcal{C})] \), using the forward-backward algorithm (Ghahramani, 2001). In the forward pass, we compute:

\[
\ln r_{n, \tau, j}^- = \ln \frac{\sum_{\iota=1}^{J} \exp \{ \ln T_{j, \iota} + \ln \rho_j, n, \tau \} \} {\mathbb{E} \ln \rho_j, n, \tau},
\]

and in the backward pass we compute:

\[
\ln \lambda_{n, \tau, j}^- = \ln \sum_{\iota=1}^{J} \exp \{ \ln \lambda_{\iota, \tau+1, t}^- + \mathbb{E} \ln T_{j, \iota} + \ln \rho_j, n, \tau \} \}.
\]

Then we can calculate the posteriors as follows:

\[
r_{n, \tau, j}^- \propto r_{n, \tau, j}^- \lambda_{n, \tau, j},
\]

\[
s_{n, \tau, j, t}^- \propto r_{n, \tau-1, j}^- \lambda_{n, \tau, t}^- \exp\{\mathbb{E} \ln T_{j, t}^- + \ln \rho_j, n, \tau \}.
\]

The expectations of \( \ln T \) and \( \ln \rho \) can be computed using standard equations for a Dirichlet distribution:

\[
\mathbb{E} \ln T_{j, \iota} = \Psi(N_{j, \iota} + \gamma_{j, \iota}) - \Psi\left(\sum_{i=1}^{J} (N_{j, i} + \gamma_{j, i})\right),
\]

\[
\mathbb{E} \ln \rho_j = \Psi(o_{j, w} + \kappa_{j, w}) - \Psi\left(\sum_{v=1}^{V} (o_{j, v} + \kappa_{j, v})\right),
\]

where \( \Psi \) is the digamma function, \( N_{j, \iota} = \sum_{n=1}^{N} \sum_{\tau=1}^{\tau_{n, \iota}} s_{n, \tau, j} \), is the expected number of times that label \( \iota \) follows label \( j \), and \( o_{j, w} \) is the expected number of times that word \( w \) occurs with sequence label \( j \). Similarly, for the seq annotator model, the expectation terms are:

\[
\mathbb{E} \ln A^{(k)}(j, l, m) = \Psi\left(\sum_{m'=1}^{M} (N_{j, l, m'}^{(k)})\right) - \Psi\left(\sum_{m=1}^{M} (N_{j, l, m}^{(k)})\right),
\]

\[
N_{j, l, m}^{(k)} = \alpha_{j, l, m}^{(k)} + \sum_{n=1}^{N} \sum_{\tau=1}^{\tau_{n, \tau, j}} r_{n, \tau, \iota}^- \delta_{\iota, t}^{(k)} \delta_{m, c_{n, \tau-1}^{(k)}},
\]

where \( \delta \) is the Kronecker delta. For other annotator models, this equation is simplified as the values of the previous labels \( c_{n, \tau-1}^{(k)} \) are ignored.

5.1 Most Likely Sequence Labels

The approximate posterior probabilities of the true labels, \( r_{j, n, \tau} \), provide confidence estimates for the labels. However, it is often useful to compute the most probable sequence of labels, \( \hat{t}_n \), using the Viterbi algorithm (Viterbi, 1967). The most probable sequence is particularly useful because, unlike \( r_{j, n, \tau} \), the sequence will be consistent with any transition constraints imposed by the priors on the transition matrix \( T \), such as preventing ‘O’ → ‘I’ transitions by assigning them zero probability.

6 Experiments

Datasets. We compare BSC to alternative methods on three NLP datasets containing both crowdsourced and gold sequential annotations. NER, the CoNLL 2003 named-entity recognition dataset (Tjong Kim Sang and De Meulder, 2003), which contains gold labels for four named entity categories (PER, LOC, ORG, MISC), with crowdsourced labels provided by (Rodrigues et al., 2014). PICO (Nguyen et al., 2017), consists of medical paper abstracts that have been annotated by a crowd to indicate text spans that identify the population enrolled in a clinical trial. ARG (Trautmann et al., 2019) contains a mixture of argumentative and non-argumentative sentences, in which the task is to mark the spans that contain pro or con arguments for a given topic. Dataset statistics are shown in Table 1. The datasets differ in typical span length, with argument components in ARG the longest, while named entities in NER spans are often only one token long.

The gold-labelled documents are split into validation and test sets. For NER, we use the split given by Nguyen et al. (2017), while for PICO and ARG, we make random splits since the splits from previous work were not available.

Evaluation metrics. For NER and ARG we use the CoNLL 2003 F1-score, which considers only

<table>
<thead>
<tr>
<th>Data</th>
<th>#sentences or docs</th>
<th>#annotators</th>
<th>#gold span</th>
<th>span length</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER</td>
<td>6,056</td>
<td>2,800</td>
<td>3,256</td>
<td>47</td>
</tr>
<tr>
<td>PICO</td>
<td>9,480</td>
<td>191</td>
<td>191</td>
<td>312</td>
</tr>
<tr>
<td>ARG</td>
<td>8,000</td>
<td>60</td>
<td>100</td>
<td>105</td>
</tr>
</tbody>
</table>

Table 1: Dataset statistics. Span lengths are means.
Table 2: Aggregating crowdsourced labels: estimating true labels for documents labelled by the crowd.

<table>
<thead>
<tr>
<th></th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
<th>CEE</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
<th>CEE</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
<th>CEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best worker</td>
<td>76.4</td>
<td>60.1</td>
<td>67.3</td>
<td>17.1</td>
<td>64.8</td>
<td>53.2</td>
<td>58.5</td>
<td>17.0</td>
<td>62.7</td>
<td>57.5</td>
<td>60.0</td>
<td>44.20</td>
</tr>
<tr>
<td>Worst worker</td>
<td>55.7</td>
<td>26.5</td>
<td>35.9</td>
<td>31.9</td>
<td>50.7</td>
<td>25.9</td>
<td>31.7</td>
<td>41.0</td>
<td>25.5</td>
<td>19.2</td>
<td>21.9</td>
<td>70.33</td>
</tr>
<tr>
<td>MV</td>
<td>79.9</td>
<td>55.3</td>
<td>65.4</td>
<td>6.24</td>
<td>82.5</td>
<td>52.8</td>
<td>64.3</td>
<td>2.55</td>
<td>40.0</td>
<td>31.5</td>
<td>34.8</td>
<td>6.47</td>
</tr>
<tr>
<td>MACE</td>
<td>74.4</td>
<td>66.0</td>
<td>70.0</td>
<td>1.01</td>
<td>25.4</td>
<td>84.1</td>
<td>39.0</td>
<td>58.2</td>
<td>31.2</td>
<td>32.9</td>
<td>32.0</td>
<td>2.62</td>
</tr>
<tr>
<td>DS</td>
<td>79.0</td>
<td>70.4</td>
<td>74.4</td>
<td>2.80</td>
<td>71.3</td>
<td>66.3</td>
<td>68.7</td>
<td>0.44</td>
<td>45.6</td>
<td>49.3</td>
<td>47.4</td>
<td>0.97</td>
</tr>
<tr>
<td>IBCC</td>
<td>79.0</td>
<td>70.4</td>
<td>74.4</td>
<td>0.49</td>
<td>72.1</td>
<td>66.0</td>
<td>68.9</td>
<td>0.27</td>
<td>44.9</td>
<td>47.9</td>
<td>46.4</td>
<td>0.85</td>
</tr>
<tr>
<td>HMM-crowd</td>
<td>80.1</td>
<td>69.2</td>
<td>74.2</td>
<td>1.00</td>
<td>75.9</td>
<td>66.7</td>
<td>71.0</td>
<td>0.99</td>
<td>43.5</td>
<td>37.0</td>
<td>40.0</td>
<td>3.38</td>
</tr>
<tr>
<td>BSC-acc</td>
<td>83.4</td>
<td>54.3</td>
<td>65.7</td>
<td>0.96</td>
<td>89.4</td>
<td>45.2</td>
<td>60.0</td>
<td>1.59</td>
<td>36.9</td>
<td>32.9</td>
<td>34.8</td>
<td>6.47</td>
</tr>
<tr>
<td>BSC-spam</td>
<td>67.9</td>
<td>74.1</td>
<td>70.9</td>
<td>0.89</td>
<td>46.7</td>
<td>84.1</td>
<td>60.1</td>
<td>1.98</td>
<td>55.7</td>
<td>53.4</td>
<td>54.5</td>
<td>2.80</td>
</tr>
<tr>
<td>BSC-CV</td>
<td>83.0</td>
<td>64.6</td>
<td>72.6</td>
<td>0.93</td>
<td>74.9</td>
<td>67.2</td>
<td>71.1</td>
<td>0.84</td>
<td>37.9</td>
<td>34.2</td>
<td>36.0</td>
<td>4.73</td>
</tr>
<tr>
<td>BSC-CM</td>
<td>79.9</td>
<td>72.2</td>
<td>75.8</td>
<td>1.46</td>
<td>60.1</td>
<td>78.8</td>
<td>68.2</td>
<td>1.49</td>
<td>56.0</td>
<td>57.5</td>
<td>56.8</td>
<td>3.76</td>
</tr>
<tr>
<td>BSC-seq</td>
<td>80.3</td>
<td>74.8</td>
<td>77.4</td>
<td>0.65</td>
<td>70.4</td>
<td>75.3</td>
<td>72.8</td>
<td>0.53</td>
<td>54.4</td>
<td>67.1</td>
<td>60.1</td>
<td>3.26</td>
</tr>
<tr>
<td>BSC-CM-notext</td>
<td>74.7</td>
<td>69.7</td>
<td>72.1</td>
<td>1.48</td>
<td>62.7</td>
<td>74.8</td>
<td>68.2</td>
<td>1.32</td>
<td>55.1</td>
<td>58.9</td>
<td>57.0</td>
<td>2.75</td>
</tr>
<tr>
<td>BSC-CM\T</td>
<td>80.0</td>
<td>73.0</td>
<td>76.3</td>
<td>0.99</td>
<td>65.8</td>
<td>66.7</td>
<td>66.2</td>
<td>0.28</td>
<td>52.9</td>
<td>49.3</td>
<td>51.1</td>
<td>1.69</td>
</tr>
<tr>
<td>BSC-seq-notext</td>
<td>81.3</td>
<td>71.9</td>
<td>76.3</td>
<td>0.52</td>
<td>81.2</td>
<td>59.2</td>
<td>68.5</td>
<td>0.73</td>
<td>36.9</td>
<td>52.0</td>
<td>43.2</td>
<td>5.64</td>
</tr>
<tr>
<td>BSC-seq\T</td>
<td>64.2</td>
<td>44.4</td>
<td>52.5</td>
<td>0.77</td>
<td>51.2</td>
<td>70.4</td>
<td>59.8</td>
<td>1.04</td>
<td>0.11</td>
<td>0.05</td>
<td>0.07</td>
<td>6.38</td>
</tr>
</tbody>
</table>

Table 2: Aggregating crowdsourced labels: estimating true labels for documents labelled by the crowd.

exact span matches to be correct. Incomplete named entities are typically not useful, and for ARG, it is desirable to identify complete argumentative units that make sense on their own. For medical trial populations, partial matches still contain useful information, so for PICO we use a relaxed F1-score, as in Nguyen et al. (2017), which counts the matching fractions of spans when computing precision and recall.

We also compute the cross entropy error (CEE, also known as log-loss). While this is a token-level rather than span-level metric, it evaluates the quality of the probability estimates produced by aggregation methods, which are useful for tasks such as active learning, as it penalises over-confident mistakes more heavily.

**Evaluated Methods.** We evaluate BSC in combination with all of the annotator models described in Section 4. As well-established non-sequential baselines, we include token-level majority voting (MV), MACE (Hovy et al., 2013) which uses the spam annotator model, Dawid-Skene (DS) (Dawid and Skene, 1979), which uses the CM annotator model, and independent Bayesian classifier combination (IBCC) (Kim and Ghahramani, 2012), which is a Bayesian treatment of Dawid-Skene. We also compare against the state-of-the-art HMM-crowd method (Nguyen et al., 2017), which uses a combination of maximum a posteriori (or smoothed maximum likelihood) estimates for the CV annotator model and variational inference for an integrated hidden Markov model (HMM). HMM-Crowd and DS use non-Bayesian inference steps and can be compared with their Bayesian variants, BSC-CV and IBCC, respectively.

Besides the annotator models, BSC also makes use of text features and a transition matrix, T, over true labels. We test the effect of these components by running BSC-CM and BSC-seq with no text features (notext), and without the transition matrix, which is replaced by simple independent class probabilities (labelled \(\notext T\)).

We tune the hyperparameters using grid search on the development sets. To limit the number of hyperparameters to tune, we optimize only three values for BSC: hyperparameters of the transition matrix, \(\gamma_j\), are set to the same value, \(\gamma_0\), except for disallowed transitions, (O-I and transitions between types, e.g. I-PER–I-ORG), which are set to \(1/e^6\); for the annotator models, all values are set to \(\alpha_0\), except for disallowed transitions, which are set to \(1/e^6\), then \(\epsilon_0\) is added to hyperparameters corresponding to correct annotations (e.g. diagonal entries in a confusion matrix). This encodes the prior assumption that annotators are more likely to have an accuracy greater than random, which avoids the non-identifiability problem in which the class labels are switched around.

**Aggregation Results.** This task is to combine multiple crowdsourced labels and predict the true labels. The results are shown in Table 2. BSC-seq outperforms the other approaches, including the previous state of the art, HMM-crowd (sig-
The performance of all methods across the three datasets varies greatly. With NER, the spans are short and the task is less subjective than PICO or ARG, hence its higher F1 scores. PICO uses a relaxed F1 score, meaning its scores are only slightly lower despite being a more ambiguous task. The constitution of an argument is also ambiguous, so ARG scores are lower, particularly as we use strict span-matching to compute the F1 scores. Raising the scores may be possible in future by using expert labels as training data, i.e. as known values of $t$, which would help to put more weight on annotators with similar labelling patterns to the experts.

We categorise the errors made by key methods and list the counts for each category in Table 3. All machine learning methods tested here reduce the number of spans that were completely missed by majority voting. Note that BSC completely removes all “invalid” spans (O-I transitions) due to the sequential model with prior hyperparameters set to zero for those transitions. For PICO and ARG, which contain longer spans, BSC-seq has lower “length error” than other methods, which is the mean difference in number of tokens between the predicted and gold spans. It also reduces the number of missing spans, although in NER and ARG that comes at the cost of producing more false positives (predicting spans where there are none). Overall, BSC-seq appears to be the best choice for identifying exact span matches and reducing missed spans.

### Table 3: Counts of different types of span errors.

<table>
<thead>
<tr>
<th>Method</th>
<th>Data set</th>
<th>exact match</th>
<th>wrong type</th>
<th>partial match</th>
<th>missed span</th>
<th>false +ve</th>
<th>late start</th>
<th>early start</th>
<th>late finish</th>
<th>early finish</th>
<th>fused spans</th>
<th>splits</th>
<th>invalid</th>
<th>length error</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV</td>
<td>NER</td>
<td>2869</td>
<td>304</td>
<td>196</td>
<td>1775</td>
<td>100</td>
<td>96</td>
<td>10</td>
<td>15</td>
<td>85</td>
<td>17</td>
<td>26</td>
<td>81</td>
<td>0.04</td>
</tr>
<tr>
<td>IBCC</td>
<td>NER</td>
<td>3742</td>
<td>386</td>
<td>187</td>
<td>829</td>
<td>345</td>
<td>107</td>
<td>27</td>
<td>43</td>
<td>77</td>
<td>47</td>
<td>29</td>
<td>74</td>
<td>0.12</td>
</tr>
<tr>
<td>HMM-crowd</td>
<td>NER</td>
<td>3650</td>
<td>334</td>
<td>115</td>
<td>1045</td>
<td>210</td>
<td>94</td>
<td>17</td>
<td>18</td>
<td>90</td>
<td>22</td>
<td>8</td>
<td>0.00</td>
<td>0.06</td>
</tr>
<tr>
<td>BSC-CM</td>
<td>NER</td>
<td>3856</td>
<td>362</td>
<td>63</td>
<td>863</td>
<td>315</td>
<td>124</td>
<td>25</td>
<td>63</td>
<td>77</td>
<td>53</td>
<td>13</td>
<td>0.14</td>
<td>0.09</td>
</tr>
<tr>
<td>BSC-seq</td>
<td>NER</td>
<td>3995</td>
<td>353</td>
<td>110</td>
<td>686</td>
<td>357</td>
<td>84</td>
<td>29</td>
<td>25</td>
<td>88</td>
<td>28</td>
<td>26</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>MV</td>
<td>PICO</td>
<td>144</td>
<td>0</td>
<td>60</td>
<td>145</td>
<td>48</td>
<td>9</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>9</td>
<td>40</td>
<td>1.26</td>
</tr>
<tr>
<td>IBCC</td>
<td>PICO</td>
<td>193</td>
<td>0</td>
<td>53</td>
<td>103</td>
<td>100</td>
<td>14</td>
<td>10</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>19</td>
<td>0.45</td>
</tr>
<tr>
<td>HMM-crowd</td>
<td>PICO</td>
<td>189</td>
<td>0</td>
<td>54</td>
<td>106</td>
<td>84</td>
<td>13</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>8</td>
<td>0.00</td>
<td>0.19</td>
</tr>
<tr>
<td>BSC-CV</td>
<td>PICO</td>
<td>156</td>
<td>0</td>
<td>76</td>
<td>117</td>
<td>81</td>
<td>10</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>2.15</td>
<td>0.26</td>
</tr>
<tr>
<td>BSC-CM</td>
<td>PICO</td>
<td>216</td>
<td>0</td>
<td>50</td>
<td>83</td>
<td>157</td>
<td>10</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>17</td>
<td>0.24</td>
<td>0.61</td>
</tr>
<tr>
<td>BSC-seq</td>
<td>PICO</td>
<td>168</td>
<td>0</td>
<td>86</td>
<td>95</td>
<td>67</td>
<td>17</td>
<td>19</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>0.61</td>
<td>0.07</td>
</tr>
<tr>
<td>MV</td>
<td>ARG</td>
<td>17</td>
<td>0</td>
<td>26</td>
<td>14</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>5.27</td>
</tr>
<tr>
<td>IBCC</td>
<td>ARG</td>
<td>27</td>
<td>1</td>
<td>21</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>9</td>
<td>3.43</td>
</tr>
<tr>
<td>HMM-Crowd</td>
<td>ARG</td>
<td>20</td>
<td>0</td>
<td>23</td>
<td>14</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4.87</td>
</tr>
<tr>
<td>BSC-CV</td>
<td>ARG</td>
<td>18</td>
<td>0</td>
<td>25</td>
<td>14</td>
<td>4</td>
<td>12</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>5.37</td>
<td>0.54</td>
</tr>
<tr>
<td>BSC-CM</td>
<td>ARG</td>
<td>35</td>
<td>1</td>
<td>12</td>
<td>9</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2.11</td>
<td>0.63</td>
</tr>
<tr>
<td>BSC-Seq</td>
<td>ARG</td>
<td>39</td>
<td>3</td>
<td>12</td>
<td>3</td>
<td>20</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.46</td>
<td>0.07</td>
</tr>
</tbody>
</table>
tions between the clusters. The third column, for example, shows annotators with a tendency toward B–I transitions regardless of the true label, while other clusters indicate very different labelling behaviour. The model therefore appears able to learn distinct confusion matrices for different workers given previous labels, which supports the use of sequential annotator models.

**Active Learning.** Active learning is an iterative process that can reduce the amount of labelled data required to train a model. At each iteration, the active learning strategy selects informative data points, obtains their labels, then re-trains a labelling model given the new labels. The updated model is then used in the next iteration to identify the most informative data points.

We simulate active learning in a crowdsourcing scenario where the goal is to learn the true labels, $t$, by selecting documents for the crowd to label. Each document can be labelled multiple times by different workers. In contrast, in a passive learning setup, the number of annotations per document is usually constant across all documents. For example, in the PICO dataset, each document was labelled six times. The aim of active learning is to decrease the number of annotations required by avoiding relabelling documents whose true labels can be determined with high confidence from fewer labels.

We simulate active learning using the **least confidence** strategy, shown to be effective by Settles and Craven (2008), as described in Algorithm 2. At each iteration, we estimate $t$ from the current set of crowdsourced labels, $c$, using one of the methods from our previous experiments as the labelling model, then use this model to determine the least confident $batch_size$ documents to be labelled by the crowd. If the simulation has requested all the labels for a document that are available in our dataset, this document is simply ignored when choosing new batches and is not selected again.

We hypothesise that BSC will learn more quickly than non-sequential and non-Bayesian methods in the active learning simulation. Bayesian methods such as BSC account for
uncertainty in the model parameters when estimating confidence, hence can be used to choose data points that rapidly improve the model itself. Sequential models such as BSC can also account for dependencies between sequence labels when estimating confidence.

**Input:** A random initial set of training labels, the same for all methods.

1. Set training set $c = \text{initial set}$
2. while training set size $< \text{max no labels}$
   1. Train model on $c$
   2. For each document $n$, compute $LC_n = 1 - p(t^*_n|c)$, where $t^*_n$ is the probability of the most likely sequence of labels for $n$.
   3. Obtain annotations for batch size documents with the highest values of $LC$ (least confidence), and add them to $c$

**Algorithm 2:** Active learning simulation using least-confidence sampling.

Figure 2 plots the mean F1 scores over ten repeats of the active learning simulation on the NER dataset (for clarity, we only plot key methods). When the number of iterations is very small, neither IBCC nor DS are able to outperform majority vote, and only produce a very small benefit as the number of labels grows. This highlights the need for a sequential model such as BSC or HMM-crowd for effective active learning with small numbers of labels. IBCC learns slightly quicker than DS, while BSC-CV clearly outperforms HMM-crowd: we believe this difference is due to the Bayesian treatment of IBCC and BSC, which means they are better able to estimate confidence than DS and HMM-crowd, which use maximum likelihood and maximum a posteriori inference. BSC-seq produces the best overall performance, and the gap grows as the number of labels increases, since more data is required to learn the more complex annotator model.

7 Conclusions

We proposed BSC, a novel Bayesian approach to aggregating sequence labels that can be combined with several different models of annotator noise and bias. To model the effect of dependencies between labels on annotator noise and bias, we introduced the seq annotator model. Our results demonstrated the benefits of BSC over established non-sequential methods, such as MACE, Dawid-Skene (DS), and IBCC. We also showed the advantages of a Bayesian approach for active learning, and that the combination of BSC with the seq annotator model improves the state-of-the-art over HMM-crowd on three NLP tasks with different types of span annotations.

In future work, we plan to adapt active learning methods for easy deployment on crowdsourcing platforms, and to investigate techniques for automatically selecting good hyperparameters without recourse to a development set, which is often unavailable at the start of a crowdsourcing process.

Acknowledgments

This work was supported by the German Research Foundation through the German-Israeli Project Cooperation (DIP; grant DA 1600/1-1 and grant GU 798/17-1), and by the German Research Foundation (DFG) as part of the QA-EduInf project (grant GU 798/18-1 and grant RI 803/12-1). We would like to thank all of those who developed the datasets used in this work.

References


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


variables. In terms of expectations over the other subsets of the prior distribution defined in Section 4. The parameters of the corresponding distributions chosen for our generative model are conjugate to the distributions over the latent variables excluding \( z \). We perform approximate inference by using coordinate ascent to update each variational factor, \( q(z) \), in turn, taking expectations with respect to the current estimates of the other variational factors. Each iteration reduces the KL-divergence between the true and approximate posteriors of Equation 12, and hence approximates a lower bound on the log marginal likelihood, also called the evidence lower bound or ELBO (Bishop, 2006; Attias, 2000).

### B Update Equations for the Forward-Backward Algorithm

For the true labels, \( t \), the variational factor is:

\[
\ln q(t_n) = \sum_{n=1}^{N} E \ln A^{(k)}(t_n, r, c_n^{(k)}, c_n^{(k)}) + E \ln T_{n, \tau-1, \tau} + \text{const.} \tag{21}
\]

The forward-backward algorithm consists of two passes. The forward pass for each document, \( n \), starts from \( \tau = 1 \) and computes:

\[
\ln r_{n, \tau, j}^- = \sum_{i=1}^{J} \left( r_{n, \tau-1, \iota}^- e^{\ln T_{i,j}} \right) + ll_{n, \tau}(j),
\]

\[
ll_{n, \tau}(j) = \sum_{k=1}^{K} E \ln A^{(k)}(j, c_n^{(k)}, c_n^{(k)}) \tag{22}
\]

where \( r_{n,0,\iota}^- = 1 \) where \( \iota = 'O' \) and 0 otherwise. The backwards pass starts from \( \tau = L_n \) and scrolls backwards, computing:

\[
\ln \lambda_{n, L_n, j}^0 = 0, \quad \ln \lambda_{n, \tau, j} = \ln \sum_{i=1}^{J} \exp \left\{ \ln \lambda_{i, \tau+1, \iota} + E \ln T_{i, j} + ll_{n, \tau+1}(\iota) \right\}. \tag{23}
\]

By applying Bayes’ rule, we arrive at \( r_{n, \tau, j}^- \) and \( s_{n, \tau, j, \iota} \):

\[
r_{n, \tau, j}^- = \frac{r_{n, \tau, j}^\lambda \lambda_{n, \tau, j}}{\sum_{j'=1}^{J} s_{n, \tau, j'} \lambda_{n, \tau, j'}} \tag{24}
\]

\[
s_{n, \tau, j, \iota} = \frac{s_{n, \tau, j, \iota} \lambda_{n, \tau, j}}{\sum_{j'=1}^{J} \sum_{\iota'=1}^{\iota} s_{n, \tau, j', \iota'} \lambda_{n, \tau, j'}} \tag{25}
\]

Each row of the transition matrix has the factor:

\[
\ln q(T_j) = \ln \text{Dir} \left( \{N_{j, \iota} + \gamma_{j, \iota}, \forall \iota \in \{1, \ldots, J\} \} \right), \tag{26}
\]

where \( N_{j, \iota} = \sum_{n=1}^{N} \sum_{\tau=1}^{T_n} s_{n, \tau, j, \iota} \) is the expected number of times that label \( \iota \) follows label \( j \). The forward-backward algorithm requires expectations of \( \ln T \) that can be computed using standard equations for a Dirichlet distribution:

\[
E \ln T_{j, \iota} = \Psi(N_{j, \iota} + \gamma_{j, \iota}) - \Psi \left( \sum_{i=1}^{J} (N_{j, \iota} + \gamma_{j, \iota}) \right), \tag{27}
\]

where \( \Psi \) is the digamma function.

### A Discussion of Variational Approximation

In Equation 12, each subset of latent variables has a variational factor of the form \( \ln q(z) = E[\ln p(z|c, \neg z)] \), where \( \neg z \) contains all the other latent variables excluding \( z \). We perform approximate inference by using coordinate ascent to update each variational factor, \( q(z) \), in turn, taking expectations with respect to the current estimates of the other variational factors. Each iteration reduces the KL-divergence between the true and approximate posteriors of Equation 12, and hence approximates a lower bound on the log marginal likelihood, also called the evidence lower bound or ELBO (Bishop, 2006; Attias, 2000).

**Conjugate distributions:** The prior distributions chosen for our generative model are conjugate to the distributions over the latent variables and model parameters, meaning that each \( q(z) \) is the same type of distribution as the corresponding prior distribution defined in Section 4. The parameters of each variational distribution are computed in terms of expectations over the other subsets of variables.
The variational factor for each annotator model is a distribution over its parameters, which differs between models. For seq, the variational factor is:

$$\ln q(A^{(k)}) = \sum_{j=1}^{J} \sum_{l=1}^{L} \text{Dir}\left( \left[ N_{j,l,m}^{(k)} \forall m \in \{1, \ldots, J \} \right] \right)$$

$$N_{j,l,m}^{(k)} = \alpha_{j,l,m}^{(k)} + \sum_{n=1}^{L} \sum_{\tau=1}^{L} r_{n,\tau,j} \delta_{l,n-1} \delta_{m,\tau},$$

(28)

where $\delta$ is the Kronecker delta. For CM, MACE, CV and acc, the factors follow a similar pattern of summing pseudo-counts of correct and incorrect answers.

C Visualising Annotator Models

Figure 3 provides an alternative visualisation of the seq models inferred by BSC-seq for annotators in the PICO dataset. The annotators were clustered as described in Section 6 of the main paper, and the mean confusion matrices for each cluster are plotted in Figure 3 using 3D plots to emphasise the differences between the likelihoods of annotators in each cluster providing a particular label given the true label value.