

Aggregating and Learning from Multiple Annotators

Silviu Paun, Queen Mary University of London

Edwin Simpson, University of Bristol

EACL 2021

<https://sites.google.com/view/alma-tutorial>

Part 1. Motivation and early approaches to annotation analysis

Introduction

- Labelling data is one of the most fundamental activities in science
- For many years now progress in Computational Linguistics has been driven by the creation of larger and richer corpora
- In the early days, the task of annotating these data was carried out by experts
- Increasingly however the amount of data required to train the state of the art has gone beyond what can be achieved with traditional expert annotation, so the field has started exploring alternative solutions
- Crowdsourcing in particular has seen huge success
- Microtask crowdsourcing via platforms such as Amazon Mechanical Turk remains the most prevalent, but citizen science-based efforts and games with a purpose have also been employed

Example of an annotation task

- Recognizing textual entailment
 - Coders are presented with two sentences and asked to judge whether the second sentence, called a hypothesis, can be inferred from the first
- A positive case of textual entailment:
 - Text: “Crude Oil Prices Slump.”
 - Hypothesis: “Oil prices drop.”
- A case of false entailment:
 - Text: “The government announced last week that it plans to raise oil prices.”
 - Hypothesis: “Oil prices drop.”

A summary of the annotation patterns from the Recognizing Textual Entailment (RTE) dataset (Snow et al., 2008)

| Number of positive annotations | Number of items | Number of items per gold class | |
|--------------------------------|-----------------|--------------------------------|----------------|
| | | Negative class | Positive class |
| 0 | 5 | 5 | 0 |
| 1 | 25 | 25 | 0 |
| 2 | 90 | 90 | 0 |
| 3 | 105 | 102 | 3 |
| 4 | 103 | 92 | 11 |
| 5 | 65 | 50 | 15 |
| 6 | 62 | 18 | 44 |
| 7 | 59 | 11 | 48 |
| 8 | 108 | 3 | 105 |
| 9 | 105 | 4 | 101 |
| 10 | 73 | 0 | 73 |
| Total | 800 | 400 | 400 |

Annotation aggregation

- The annotations, once collected, need to be distilled from noise
- The dataset contains a fair amount of disagreement and the labels proposed by the majority of coders are not always right, at least as judged a posteriori by a panel of experts
- But the correct labels can be found among the annotations; we just need better tools than simple heuristics such as majority voting to identify them
- Following a long line of work, spanning across multiple decades, it has been suggested probabilistic models of annotation can be the right tools for this assignment

Probabilistic models of annotation

- Simply put, a model of annotation is a probabilistic framework for distilling the disagreement between coders from noisy interpretations
 - We can specify our assumptions about the annotation process, e.g., the interactions between the items and the coders
 - Our assumptions are then considered when inferring the corpus labels
- Most models of annotation share these characteristics:
 - The items are assumed to have one correct interpretation, often referred to in the literature as their true class
 - The coders are assumed to have their annotation behaviour dependent on the true class of the items

The generative process of a (completely) pooled model

- The items (e.g., the pairs of sentences in our recognizing textual entailment example) are first assigned a true class (the entailment status of a pair of sentences) based on the prevalence of the true classes in the corpus (the percentage of sentence pairs with positive and negative entailment):

$$c_i \sim \text{Categorical}(\pi)$$

- Subsequently, the model assumes that each annotation is produced according to a population-wide annotation behavior associated with the true class:

$$y_{i,n} \sim \text{Categorical}(\zeta_{c_i})$$

The generative process of a (completely) pooled model

- This simple model does not distinguish between the different abilities the annotators may have and assumes instead they all follow a common behavior
- The population-wide behavior is formulated in terms of a confusion matrix, whose rows are probability distributions governing the annotation behavior of the coders on items of a certain true class
- In our example the confusion matrix captures the probability of labelling some pair of sentences as a negative or a positive instance of entailment when their true status is either positive or negative

A pooled model: parameter estimates

- Fitting the just described model on the RTE dataset results in an estimated prevalence of (0.55, 0.45) and an overall coder specificity of 0.65 and sensitivity 0.84
- These estimates inform us there is a relatively balanced number of positive and negative instances in the corpus and that the coders can identify the positive instances considerably better than the negative ones
- The estimates derived from gold (expert) labels indicate an evenly balanced prevalence and a specificity of 0.66 and sensitivity 0.80
- The inferred textual entailment status of the sentence pairs matches the gold labels on 709 out of 800 cases, for an accuracy of about 89%

Moving on: unpooled models

- A limitation of the model presented before is that it assumes that all coders share the same annotation behaviour, and thus is unable to capture the accuracy and bias of the individual annotators
- In a seminal work, Dawid and Skene (1979) introduced a model in which each annotator is characterized by their own confusion matrix
- Now each annotation of an item is assumed to be produced following the behaviour of its coder on items of that true class:

$$y_{i,n} \sim \text{Categorical}(\zeta_{jj[i,n],c_i})$$

More unpooled models

- Another widely used unpooled model in NLP is the MACE model (Hovy et al, 2013). This model has a simpler parameterization of the accuracy and bias of the annotators formulated in terms of an ability parameter and a spamming behaviour
- When presented with an item, the model assumes the annotator first decides whether or not he knows the answer:

$$z_{i,n} \sim \text{Bernoulli}(\theta_{jj[i,n]})$$

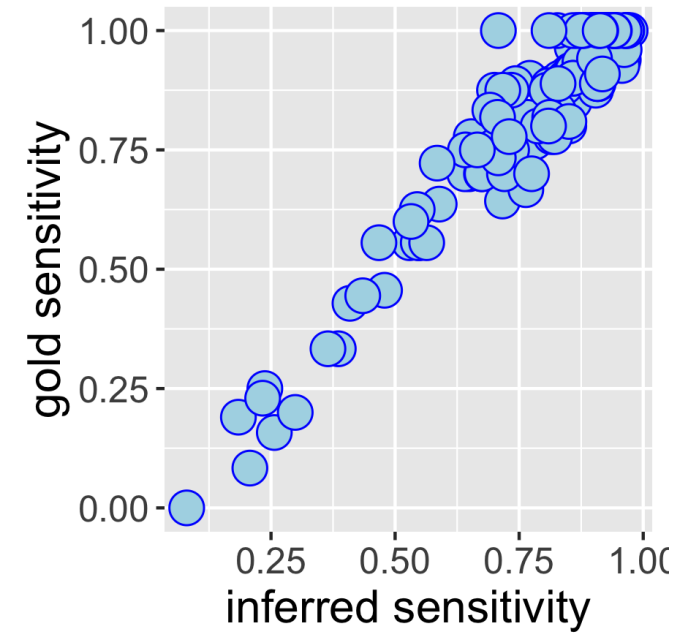
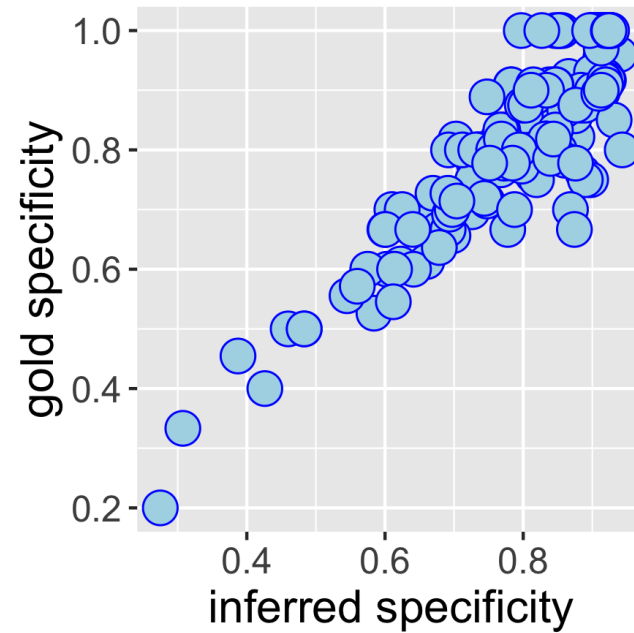
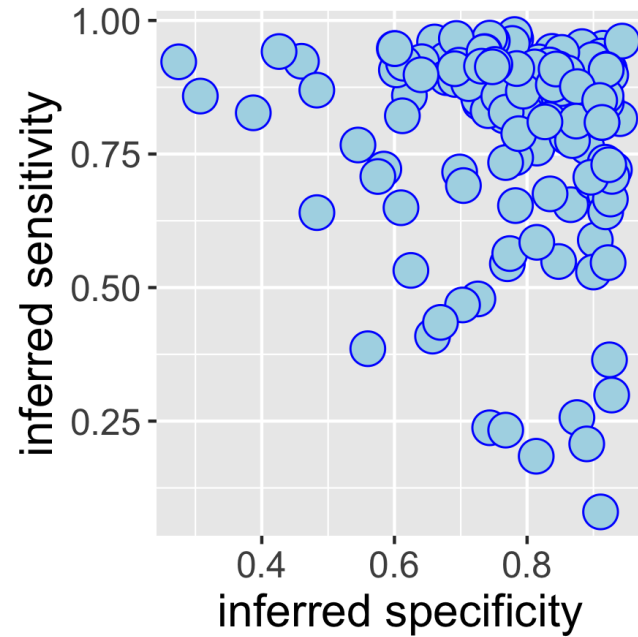
- Annotators of high ability (θ_j close to 1) are more likely to correctly annotate the items; if an annotator knows the answer ($z_{i,n} = 1$) they produce the true class:

$$y_{i,n} = c_i$$

- But when they do not know the answer ($z_{i,n} = 0$), the annotators annotate based on their spamming preference:

$$y_{i,n} \sim \text{Categorical}(\beta_{jj[i,n]})$$

Coder estimates for the model of Dawid and Skene (1979) on RTE dataset



Typical coder profiles extracted from the RTE dataset

| Example of a typically found annotator | Gold* confusion | Dawid and Skene (1979) | Hovy et al. (2013) |
|---|--|--|--|
| the good annotator | $\begin{pmatrix} 0.88 & 0.12 \\ 0.25 & 0.75 \end{pmatrix}$ | $\beta_{17} = \begin{pmatrix} 0.81 & 0.19 \\ 0.24 & 0.76 \end{pmatrix}$ | $\theta_{17} = 0.65$ $\epsilon_{17} = (0.58, 0.42)$ |
| the highly accurate annotator | $\begin{pmatrix} 0.80 & 0.20 \\ 0.00 & 1.00 \end{pmatrix}$ | $\beta_{119} = \begin{pmatrix} 0.94 & 0.06 \\ 0.04 & 0.96 \end{pmatrix}$ | $\theta_{119} = 0.95$ $\epsilon_{119} = (0.47, 0.53)$ |
| the annotator biased towards the first class | $\begin{pmatrix} 1.00 & 0.00 \\ 1.00 & 0.00 \end{pmatrix}$ | $\beta_{88} = \begin{pmatrix} 0.91 & 0.09 \\ 0.92 & 0.08 \end{pmatrix}$ | $\theta_{88} = 0.08$ $\epsilon_{88} = (0.95, 0.05)$ |
| the annotator biased towards the second class | $\begin{pmatrix} 0.20 & 0.80 \\ 0.00 & 1.00 \end{pmatrix}$ | $\beta_{65} = \begin{pmatrix} 0.28 & 0.72 \\ 0.08 & 0.92 \end{pmatrix}$ | $\theta_{65} = 0.25$ $\epsilon_{65} = (0.08, 0.92)$ |

Example of typical annotators found in a dataset. The gold* confusion is computed by matching the annotations of a coder against the gold standard; due to the limited number of annotations this provides only a rough estimate.

Unpooled models: more analysis

- The prevalence of the true classes was estimated to be (0.48, 0.52), a close estimate to the (0.5, 0.5) prevalence present in the gold standard
- Both the model of Dawid and Skene (1979) and that of Hovy et al. (2013) infer the same labels, correctly (according to the gold standard) adjudicating 743 items, for an accuracy of about 0.93
- The models over-perform a simple majority vote baseline by 3 percentage points

The notion of item difficulty

- The models discussed so far assume that the coders annotate independently given the true class of the items
- Recognizing that some items are more challenging than others violates the independence assumption
- Example of an 'easy' judgement:
 - Text: "The three-day G8 summit will take place in Scotland."
 - Hypothesis: "The G8 summit will last three days."
- Example of a 'difficult' judgement:
 - Text: "EU membership is a strategic necessity for Turkey, as Ankara will inevitably suffer greater foreign policy problems in the future unless it makes it into the Union."
 - Hypothesis: "Turkey to join the EU."

Item difficulty (Carpenter, 2008)

- Follows previous work in item response theory and ideal point models
- The probability of an annotator being correct on an item is based on a subtractive relationship between their ability and the difficulty of the item
- For an item whose true class is positive ($c_i = 1$), the model assumes an annotation is supplied based on the sensitivity α_j of the annotator and the difficulty of the item θ_i :

$$y_{i,n} \sim \text{Bernoulli}(\text{logistic}(\alpha_{jj[i,n]} - \theta_i))$$

- If the item was assigned a negative class ($c_i = 0$), the specificity of the annotator β_j is involved instead:

$$y_{i,n} \sim \text{Bernoulli}(1 - \text{logistic}(\beta_{jj[i,n]} - \theta_i))$$

Item difficulty (Carpenter, 2008)

- When the difficulty of the item is 0 the model is equivalent to a log odds reparameterization of the model of Dawid and Skene (1979)
- If the item is easy ($\theta_i < 0$) both the sensitivity and the specificity of the annotator are increased, the annotator having a larger probability to supply the correct answer
- In case of a hard item ($\theta_i > 0$) , the difficulty parameter reduces the sensitivity and the specificity of the annotator, thus increasing their chance of being wrong

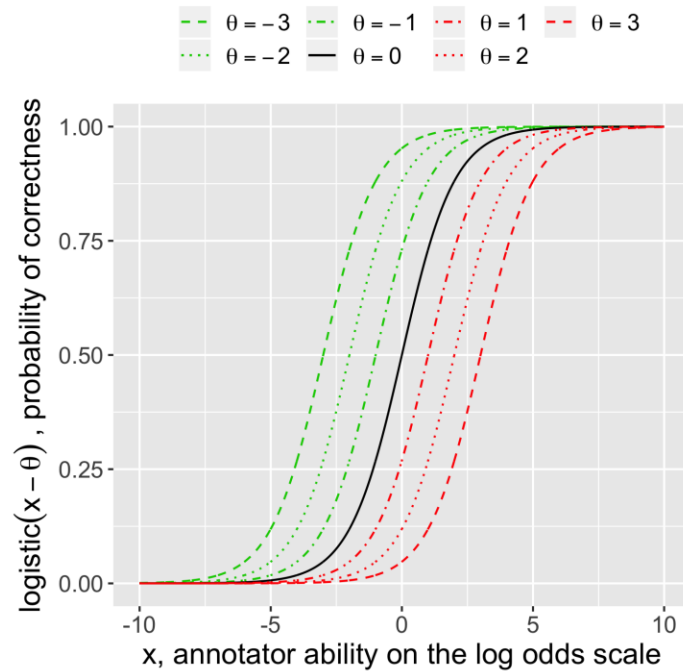
Item difficulty (Whitehill et al, 2009)

- Under GLAD, the difficulty of the items is in a multiplicative relationship with the ability of the coders. Unlike the Carpenter (2008) model, here we have a cross class coder ability parameter, representing their probability of correctly annotating an item.
- The difficulty of an item is modelled with parameter β_i , constrained to the positive side of a log odds scale. The annotations for an item whose true class is c_i , have the following probability of correctness:

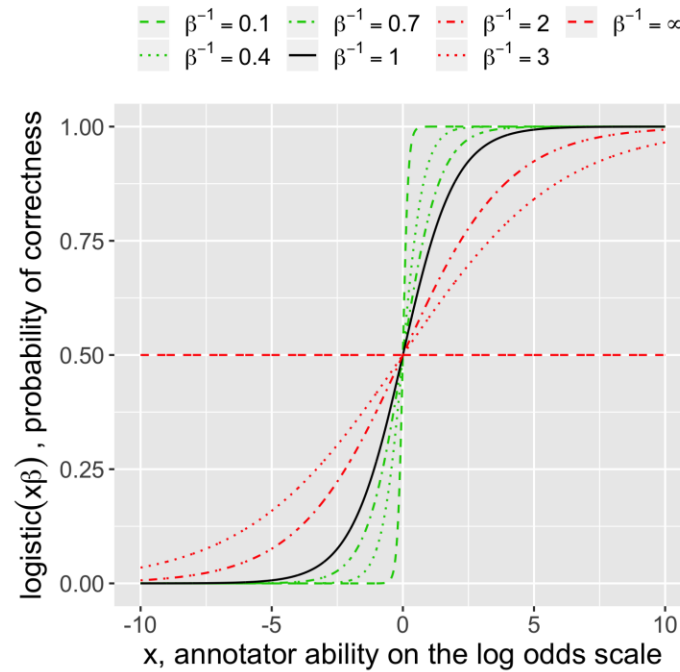
$$p(y_{i,n} = c_i) = \text{logistic}(\alpha_{jj[i,n]} \beta_i)$$

- On easy items ($0 < \beta_i^{-1} < 1$) the annotators' accuracy is increased, reaching perfect accuracy as $\beta_i^{-1} \rightarrow 0$. On more difficult items (characterized by $\beta_i^{-1} > 1$) the coders have a lower probability of correctness, reaching random chance as $\beta_i^{-1} \rightarrow \infty$.

Item difficulty: a geometric interpretation



(a) Subtractive relationship (Carpenter)



(b) Multiplicative relationship (Whitehill)

The effects of item difficulty on the ability of an annotator. Green corresponds to easy items, black to borderline difficulty, and red to hard items.

Item difficulty: a practical example using the RTE dataset

| Positive annotations | No. items | Positive items (gold) | Carpenter (2008) | | Whitehill et al. (2009) | |
|----------------------|-----------|-----------------------|------------------|------------|-------------------------|------------|
| | | | avg. θ_i | pos. items | avg. β_i^{-1} | pos. items |
| 0 | 5 | 0 | -1.55 | 0 | 0.48 | 0 |
| 1 | 25 | 0 | -1.03 | 0 | 0.32 | 0 |
| 2 | 90 | 0 | -0.58 | 0 | 0.29 | 0 |
| 3 | 105 | 3 | -0.14 | 4 | 0.32 | 2 |
| 4 | 103 | 11 | 0.27 | 7 | 0.52 | 3 |
| 5 | 65 | 15 | 0.65 | 13 | 0.78 | 12 |
| 6 | 62 | 44 | 0.78 | 40 | 0.96 | 39 |
| 7 | 59 | 48 | 0.58 | 47 | 0.73 | 51 |
| 8 | 108 | 105 | 0.10 | 106 | 0.45 | 108 |
| 9 | 105 | 101 | -0.36 | 105 | 0.25 | 105 |
| 10 | 73 | 73 | -0.91 | 73 | 0.15 | 73 |

The average difficulty estimated by the models of Carpenter (2008) and Whitehill et al. (2009) for items with a certain amount of disagreement

Looking back for a moment

- We started with a simple model of annotation which assumed all coders have the same annotation behavior
 - That was an example of a completely pooled model
- The models subsequently introduced relaxed the aforementioned assumption and assigned each annotator with their own ability parameters
 - These models are often referred to as unpooled models, i.e., models with individual structures
- Although effective in practice, one downside to having an unpooled model structure is that it can require a good number of observations to properly fit some of the parameters, e.g., fitting the accuracy and bias of the annotators

The issue of data sparsity

- In some crowdsourcing environments this requirement of plentiful observations can be satisfied, e.g., in microtask platforms where you can control for the number of annotations per annotator and per item
- In more open environments, such as in games with a purpose where the data collection process relies on player engagement, sparsity becomes an important factor to be considered
- Sparsity also plays a role in the early days of a crowdsourcing campaign and, more generally, alleviating its effects can reduce the time that is necessary to achieve the desired level of annotation quality and cut down the costs of the project

Hierarchical structures (partial pooling)

- One solution to dealing with sparsity is to extend the models with a hierarchical structure, also referred to as a partially-pooled structure
- This type of structure improves (regularizes) the estimates of the lower level parameters using information about the hierarchy
- It does so by pooling the individual estimates towards the mean of the hierarchical prior
- The level of pooling is as strong as evidenced by the data
 - As already touched on, in situations of sparsity, the lack of observations is compensated using hierarchical information through parameter pooling. In a hierarchical model of coder ability the hierarchical prior informs more strongly the posterior of the lower level annotator estimates.
 - When plenty of observations are available the partially pooled and the unpooled models should perform similarly. In this setting the likelihood of the annotations will dominate over the prior.

Modelling annotator communities

- Typical annotator communities found in a crowdsourcing setup include spammers, adversarial, biased, average or high quality players
- Knowledge of these communities would allow to regularize the behaviour of the annotators towards the profile of the community they are part of
- Venanzi et al. (2014) provide such a model. A nonparametric variant is put forward by Moreno et al. (2015). We briefly introduce below the former.
- Assuming there are C communities in the pool of annotators, and θ encodes their prevalence, the model begins by assigning each coder a community profile:

$$z_j \sim \text{Categorical}(\theta)$$

Modelling (still) annotator communities

- Annotators have their class behaviour drawn from the community profile they were assigned to previously:

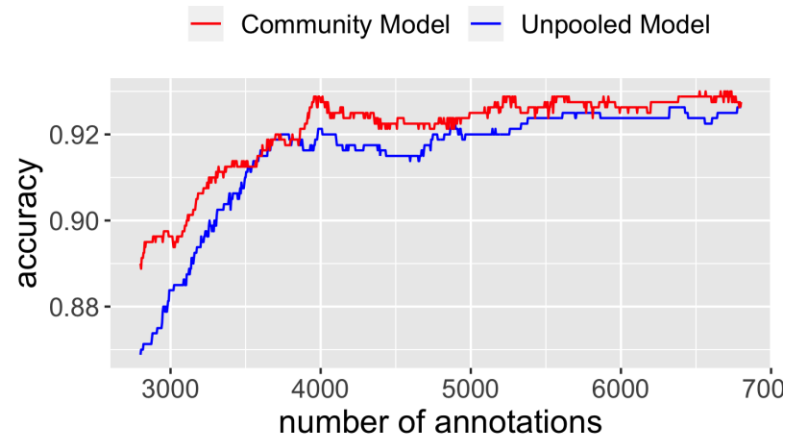
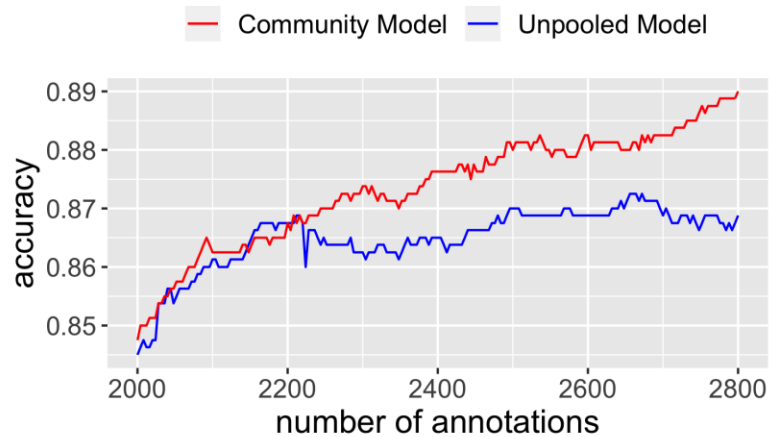
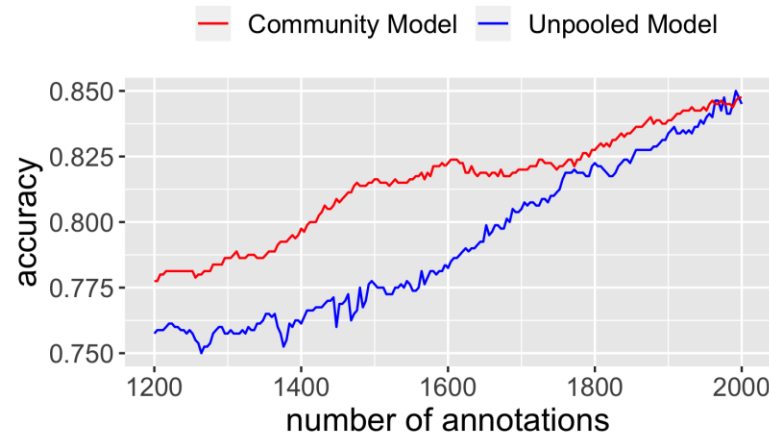
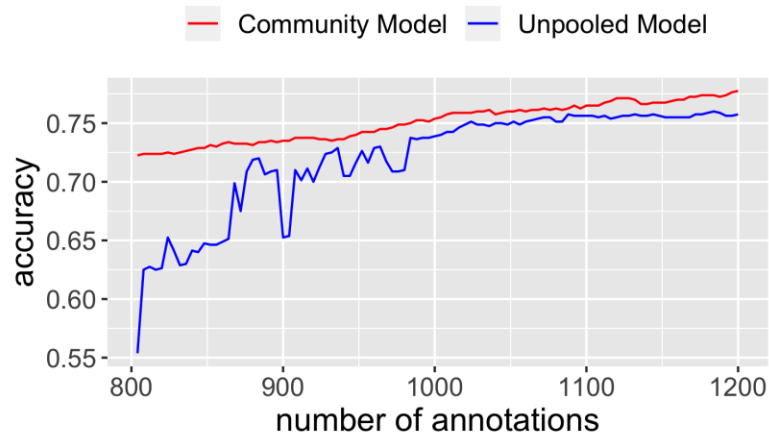
$$\beta_{j,k} \sim \text{MultivariateNormal}(\zeta_{z_j,k}, \Omega_{z_j,k})$$

- The community profiles are parameterized in terms of a location parameter $\zeta_{z_j,k}$ encoding the average behaviour of the coders assigned to them and a covariance parameter $\Omega_{z_j,k}$ intended to capture the level of pooling exerted by the community on its members

Modelling annotator communities: parameter estimates

- Applied to the RTE dataset the community model of Venanzi et al. (2015) gets an accuracy on par with the unpooled model of Dawid and Skene (1975)
 - It's when the data is sparse the partially pooled structure shows its true utility. Wait for the next slide.
- The model inferred two prevalent communities
 - “Good annotators”: 88% prevalence and $\begin{pmatrix} 0.92 & 0.08 \\ 0.13 & 0.87 \end{pmatrix}$ profile
 - “Coders biased towards negative entailment”: 12% prevalence and $\begin{pmatrix} 0.37 & 0.63 \\ 0.18 & 0.82 \end{pmatrix}$ profile

The accuracy of a community model (Venanzi et al., 2014) and its unpooled counterpart (Dawid and Skene, 1979) in an adaptive learning setting (RTE dataset)



Part 2. Advanced models of annotation

Sequence Annotations

- In text annotation schemes such as BIO, the probability of a label depends on what came before it.
- E.g., 'Inside' (I) cannot follow 'Outside' (O):

 O O O B I I I
...the teachers observations. As it was the
 I I I O O O O O
same back then, I ruled out a trauma ...

- Ignoring these dependencies can lead to invalid sequences and disregards indications from preceding labels that annotators may confuse 'B' or 'I' labels.

Badly-formed Spans: Example

- On the right, we get an invalid aggregate sequence using a method like D&S or MACE.
- If we knew that B and I were related, we could include 'As' in the span.
- Annotator 3 has labelled 'observations' as B, so are more like to label 'As' as I.

| | 1 | 2 | 3 | Aggregate |
|---------------|---|---|---|-----------|
| the | O | O | O | O |
| teachers' | O | O | O | O |
| observations. | O | O | B | O |
| As | O | B | I | O |
| it | B | I | I | I |
| was | I | I | I | I |
| the | I | I | I | I |
| same | I | I | I | I |
| back | O | I | I | I |
| then | O | I | I | I |

Condition Random Fields with Multiple Annotators (CRF-MA)

- CRFs assume the probability of a label depends on the previous label and the **features** of the data point.
- Rodrigues et al. (2014) proposed a model for learning CRFs from multiple annotators that resolves the challenges of sequence labels.
- The model assumes that for each sequence, one annotator is reliable, and the others label randomly, with annotators weighted by a performance metric.

$$p(\mathbf{y}^r, \mathbf{x}) = \sum_{r=1}^R \pi_r p_{crf}(\mathbf{y}^r | \mathbf{x}, \boldsymbol{\lambda}) \prod_{j=1, j \neq r}^R p_{rand}(\mathbf{y}^j | \mathbf{x})$$

CRF prediction for the reliable annotator

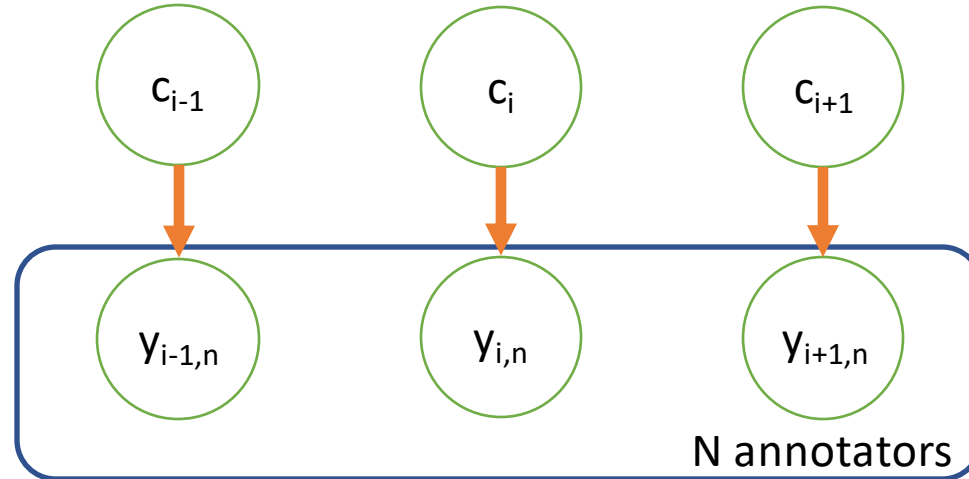
Random predictions

Condition Random Fields with Multiple Annotators (CRF-MA)

- Expectation maximization can be used to estimate both the true label sequence and a CRF model.
- CRF-MA improves on CRF and majority voting on NER with ~5 annotators per document.
- Drawbacks of CRF-MA:
 - The annotator model is not able to capture labelling bias (e.g. spamming), unlike Dawid & Skene's confusion matrices.
 - There is no way to encode prior knowledge on the legality of label transitions.
 - As it is based on CRFs, it suffers from slow training and prediction.

Sequential Version of Dawid & Skene?

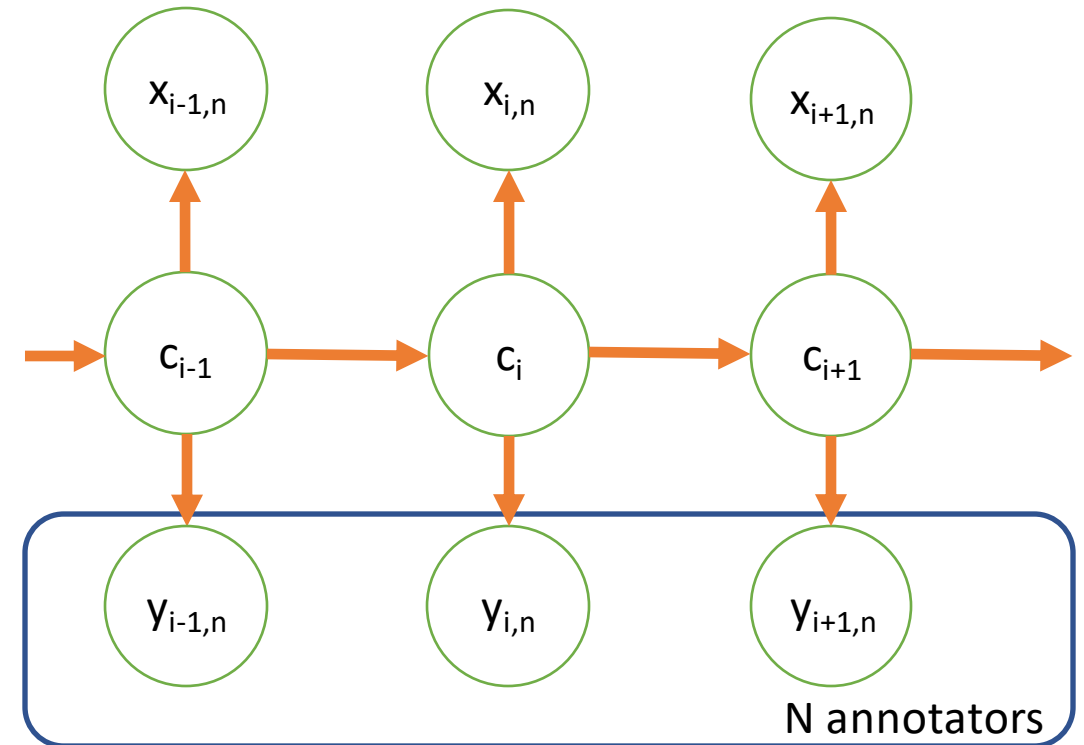
- Can we handle sequential dependencies and still use any of the successful Dawid & Skene-based models for crowdsourcing?
- Dawid & Skene assume independent true labels: $c_i \sim \text{Categorical}(\pi)$.



- Replace this with a generative model of sequential dependencies.

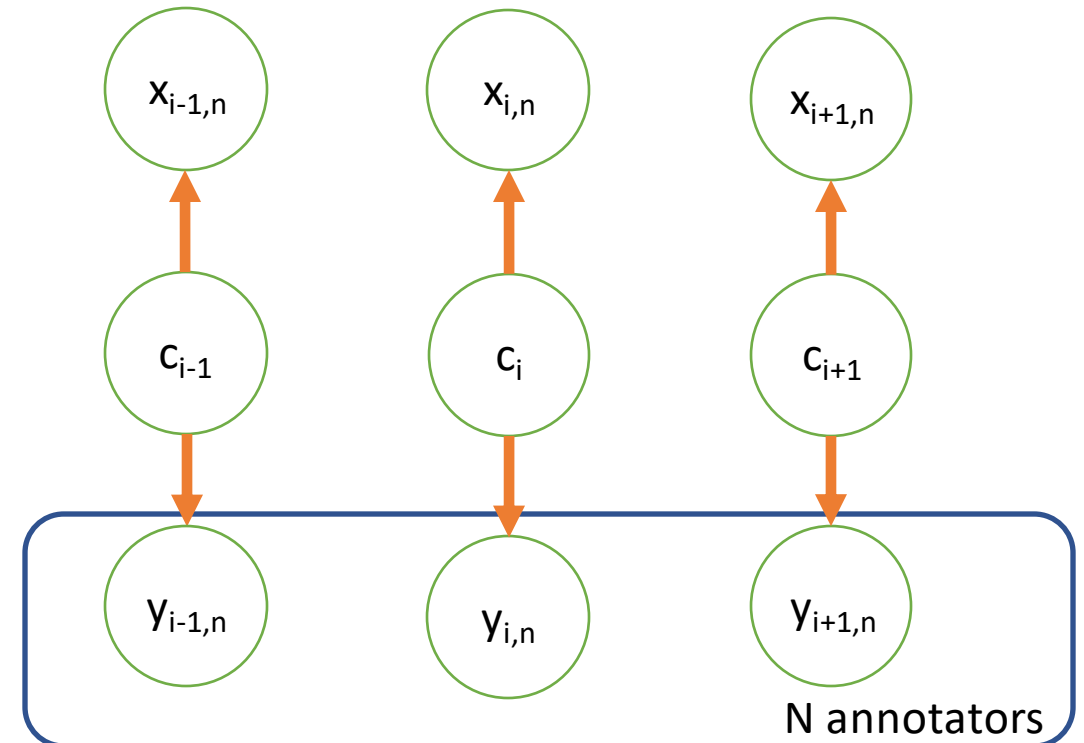
Dawid & Skene + Hidden Markov Models

- Replace $c_i \sim \text{Categorical}(\pi)$ by a hidden Markov model (HMM):
- The HMM's transition probability models probability of true label given its predecessor;
- HMM emission model: probabilities of features given the true label;
- This is the approach of HMM-crowd (Nguyen et al., 2017).



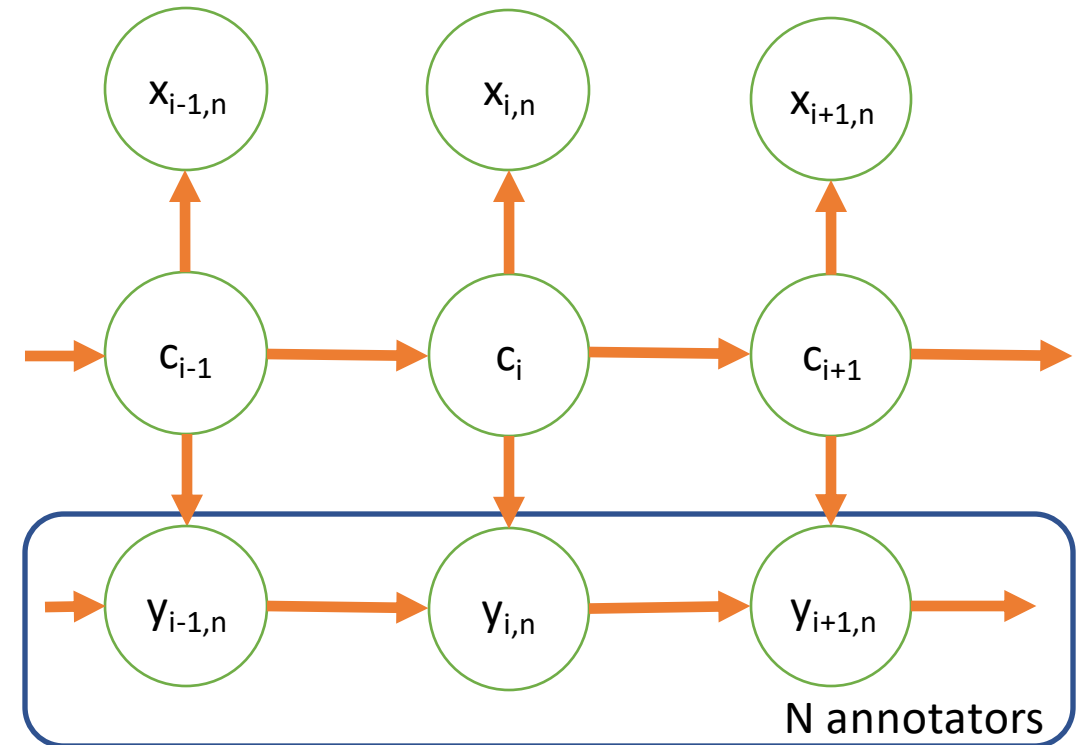
Dawid & Skene + Other True Label Models

- Generally, $c_i \sim \text{Categorical}(\pi)$ can be replaced with any distribution that's appropriate for the data.
- E.g., a text classifier that uses the features, x_i .
- True label models can be plugged in to Dawid & Skene...
- ...We'll come back to this later!



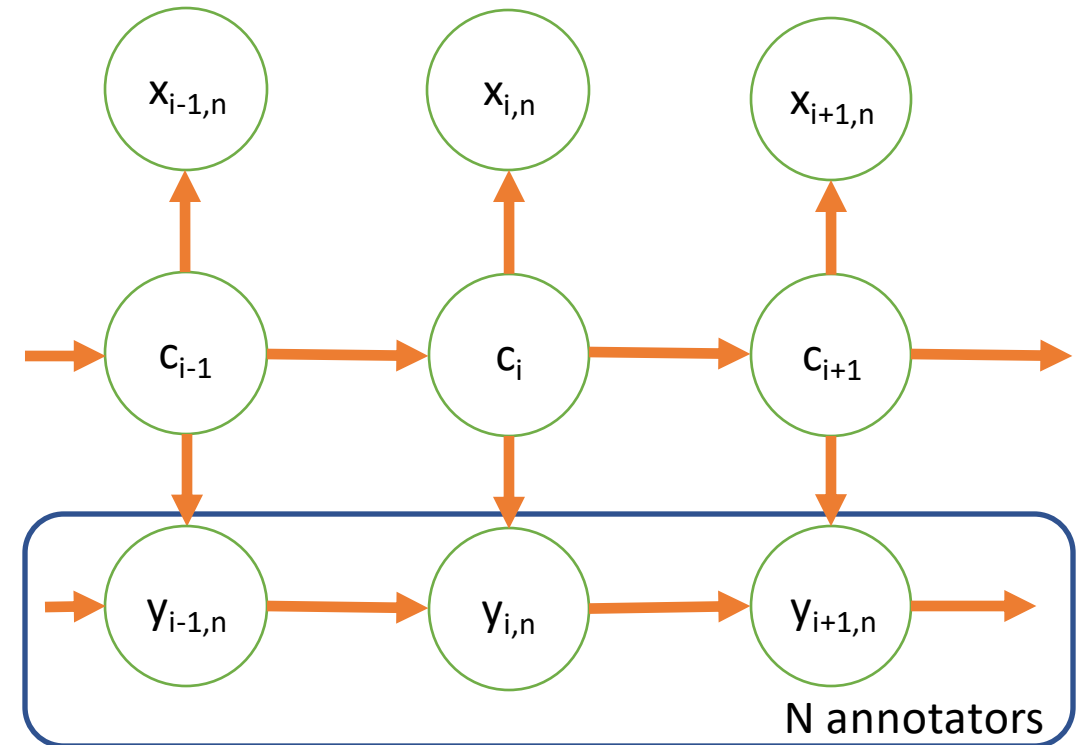
Sequential Confusion Matrices

- Annotators' labels also depend on what came before – they cannot break BIO labelling rules.
- E.g., if annotator misses the first token in a span, they're more likely to label the next one as 'B' when true label is 'I'.
- Some tend to miss starts and ends of spans or split spans.



Sequential Confusion Matrices

- Simpson & Gurevych (2019) allow the confusion matrix ζ to depend on the previous label, $y_{i-1,n}$.
- In effect, the confusion matrices are *unpooled* further so there are separate matrices ζ for each possible value of $y_{i-1,n}$.



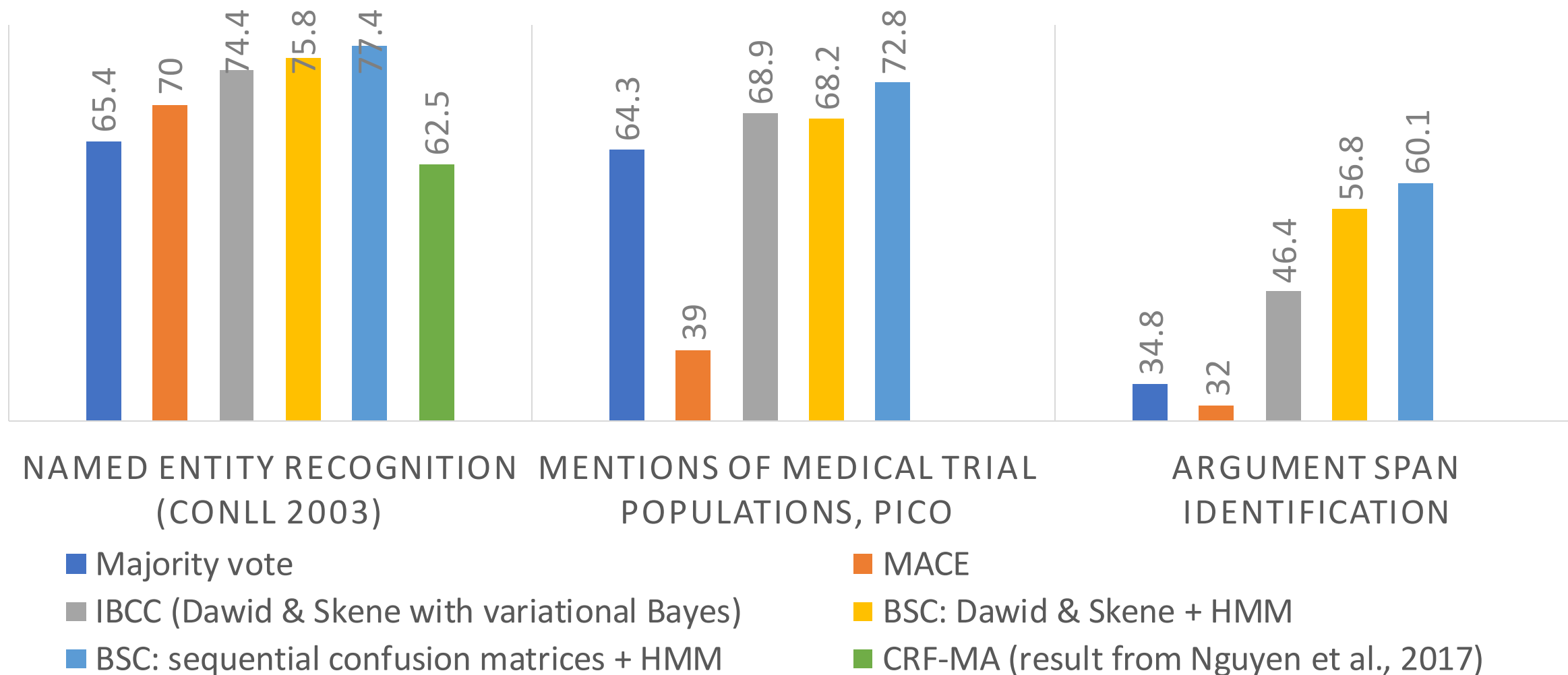
Bayesian Sequence Combination (BSC)

- Simpson & Gurevych (2019) also set priors over the transition matrix to prevent the model from predicting any illegal transitions.

$$p(p(c_{i,n} | c_{i-1,n} = O)) \sim \text{Dirichlet}(\boldsymbol{\alpha}_{n,O}),$$

- E.g., for BIO-encoding, set the prior hyperparameter for transitions from 'O' (outside) to 'I' (inside) labels close to zero: $\alpha_{n,O,I} = 1e^{-6}$
- **Variational Bayesian inference:** compared to maximum likelihood or maximum a posteriori methods, approximate Bayesian inference reduces the influence of annotators who provide fewer labels, as their confusion matrix is known with less confidence.

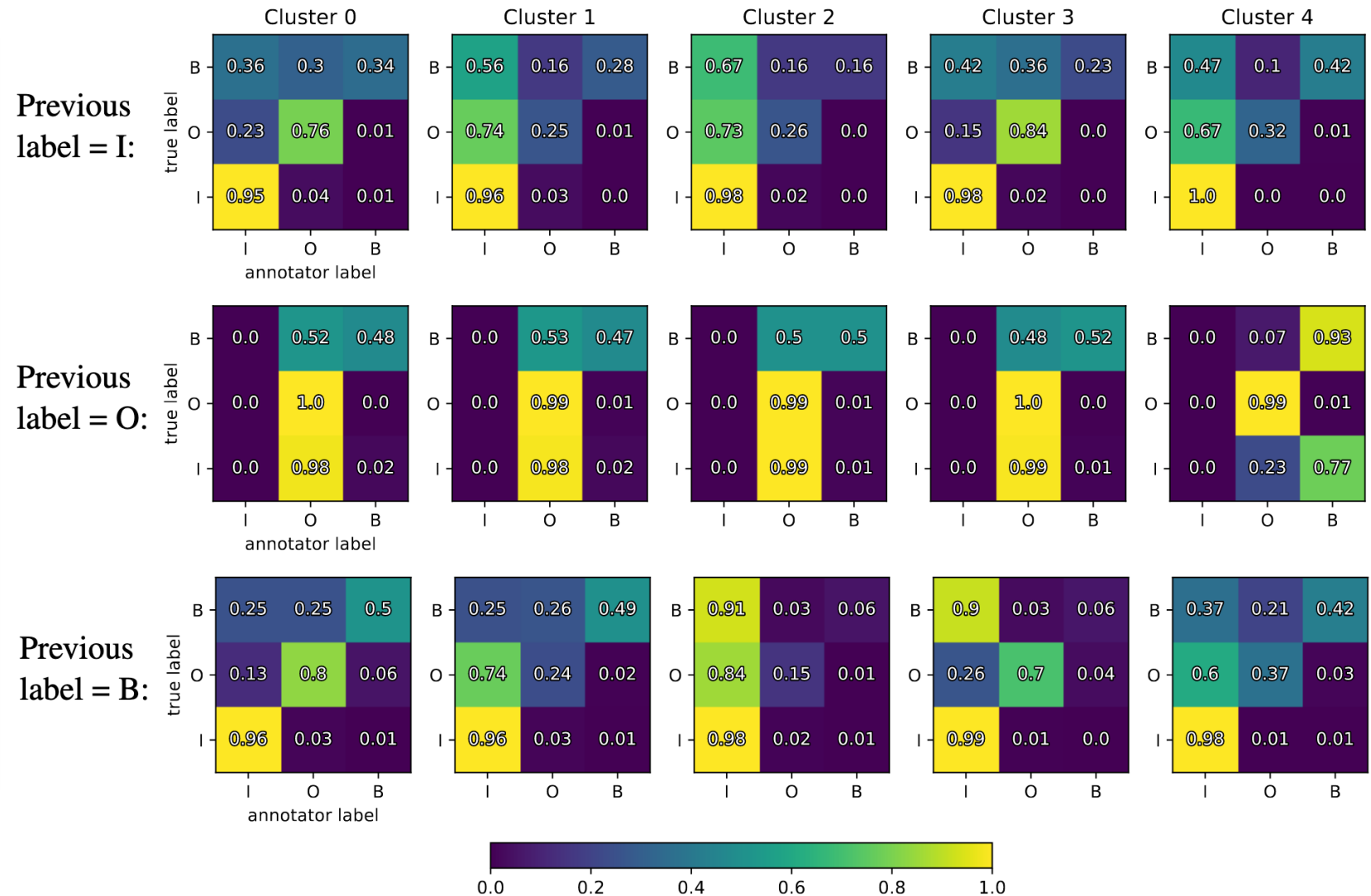
Benefits of BSC (Simpson & Gurevych 2019)



Sequential Confusion Matrices

Workers clustered according to ζ .

Plots show differences between ζ depending on previous label from the PICO dataset.



Modelling anaphoric annotations

- The task of identifying and resolving anaphoric reference to discourse entities is known in NLP as coreference resolution
- In standard annotation schemes for anaphora/coreference the annotators may mark a mention as discourse new if it introduces a new entity into the discourse or as discourse old if it refers to an already introduced entity
- In the latter case the annotators also have to specify the entity in question
 - One way to do that is to link the mentions with their most recent antecedents
 - The entities can then be determined by following the link path indicated by the most recent antecedent of each mention
- Richer annotation schemes allow annotators to also mark, e.g., expletives and predicative noun phrases

Let's look at an example

John, a colleague from work, said it will rain later today. He was right.

- The annotators should mark:
 - “John” as discourse new
 - “a colleague from work” as a predicative noun phrase
 - “it” as an expletive
 - and the pronoun “he” as a discourse old mention further selecting “John” as the most recent antecedent that refers to the same entity
 - Mentions “John” and “he” are said to form a coreference chain

The anaphoric annotation task

- Apparently, this is a classification task
- But unlike standard classification tasks, the set of classes the annotators can choose from changes depending on the mentions they annotate
- For this reason, standard models of annotation are not immediately applicable to aggregate anaphoric judgements
- Paun et al. (2018b) addressed this gap with their mention pair model of annotation (MPA)

Introducing the MPA model

- MPA assumes a pre-processing step where the annotations of each mention are transformed into binary agreement decisions w.r.t. each distinct collected interpretation
- For example, for the pronoun “he” in our example, let us assume we collect 3 judgements
 - Say 2 annotators provide the correct answer, i.e., a discourse old interpretation with “John” as the most recent antecedent of the entity
 - And the 3rd one, a spammer, chooses the discourse new interpretation
- The collected judgements are processed as follows:
 - Interpretation DO(“John”) belonging to the DO class with annotations 1, 1, 0
 - Interpretation DN() belonging to the DN class with annotations 0, 0, 1

The generative process of MPA

- For every mention and distinct interpretation, the model first decides whether the **mention pair** is a true mention pair or not:

$$c_{i,m} \sim \text{Bernoulli}(\pi_{z_{i,m}})$$

- If the mention pair is believed to be correct ($c_{i,m} = 1$), then the associated binary decisions are assumed to be the result of the annotators' class sensitivity:

$$y_{i,m,n} \sim \text{Bernoulli}(\alpha_{jj[i,m,n],z_{i,m}})$$

- When the mention pair is considered incorrect ($c_{i,m} = 0$) the binary decisions are modelled according to the class specificity of the coders:

$$y_{i,m,n} \sim \text{Bernoulli}(1 - \beta_{jj[i,m,n],z_{i,m}})$$

Some final comments on MPA

- The annotators have a sensitivity and a specificity associated with each class
 - This allows to capture, e.g., that discourse old interpretations are generally harder compared to discourse new ones
- After the parameters have been estimated each mention is assigned the most likely interpretation based on the posterior of the mention pair indicators
- The coreference chains (entities) can then be built by simply following the link path from the adjudicated mention pairs
- The model can also be used in an analysis of anaphoric ambiguity
 - Identify those mentions with no or more than one likely interpretation

How well does MPA do

- We used MPA to adjudicate the anaphoric interpretations collected over many years by the **Phrase Detectives** game with a purpose
- The latest version of the released corpus (Poesio et al., 2019) contains at least 8 anaphoric judgements for over 100 thousand mentions from about 540 documents covering 2 main genres, Wikipedia articles and fiction from the Gutenberg collection
- 45 of those documents, containing over 6 thousand mentions, were annotated by linguists to provide a reliable gold standard for evaluation

Evaluating mention pairs

| | Majority Vote | | | MPA (Paun et al., 2018b) | | |
|-----------------|---------------|-------------|------|--------------------------|-------------|-------------|
| | Precision | Recall | F1 | Precision | Recall | F1 |
| discourse old | 0.94 | 0.63 | 0.75 | 0.90 | 0.87 | 0.89 |
| discourse new | 0.79 | 0.99 | 0.88 | 0.95 | 0.96 | 0.95 |
| predicative NPs | 0.54 | 0.10 | 0.16 | 0.64 | 0.72 | 0.68 |
| expletives | 0.97 | 0.71 | 0.82 | 0.94 | 0.98 | 0.96 |
| Accuracy | 0.83 | | | 0.92 | | |
| Avg. F1 | 0.66 | | | 0.87 | | |

The quality of the aggregated mention pairs evaluated against a gold standard built by linguists

Evaluating coreference chains

| | Method | MUC | | | BCUB | | | CEAFE | | | Avg. F1 |
|------------------------|----------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | | P | R | F1 | P | R | F1 | P | R | F1 | |
| Singletons included | MV | 96.0 | 63.9 | 76.7 | 95.7 | 78.7 | 86.4 | 77.1 | 94.9 | 85.1 | 82.7 |
| | MPA | 91.6 | 82.4 | 86.8 | 94.8 | 87.8 | 91.2 | 92.4 | 93.8 | 93.1 | 90.3 |
| | Stanford | 65.4 | 62.4 | 63.8 | 78.9 | 76.1 | 77.5 | 78.4 | 85.2 | 81.7 | 74.3 |
| Singletons excluded | MV | 96.1 | 64.8 | 77.4 | 93.8 | 45.0 | 60.8 | 66.3 | 48.5 | 56.1 | 64.8 |
| | MPA | 92.2 | 89.2 | 90.7 | 88.1 | 77.8 | 82.6 | 79.5 | 80.2 | 79.8 | 84.4 |
| | Stanford | 65.7 | 62.1 | 63.9 | 50.3 | 42.5 | 46.1 | 42.7 | 49.8 | 46.0 | 52.0 |

The quality of the coreference chains inferred using different methods evaluated using standard coreference metrics against expert-annotated chains. MV stands for chains determined from mention pairs inferred using majority voting, MPA for chains inferred from MPA-deduced mention pairs, and Stanford for chains produced by the (Lee et al., 2011) deterministic coreference system

Ambiguous and Continuous Labelling Tasks

- Motivating tasks:
 - How good is translation A?
 - How complete is summary B?
 - How funny is joke C?
 - How metaphorical is metaphor D?
 - How convincing is argument E?
- Let's consider rating two example arguments:
- Score out of ten? Likert scale?

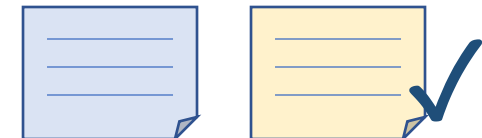
Who said anything about FF/mozilla having anything to do with steve jobs/apple? Seperate enititys, not in the least bit attached! BTW, IE blows when making pages and ...

Firefox takes the best of all previous browsers and sticks it all in one neat package. Security and extendibility are some of its top features. And those times when ...

Comparisons and Preferences



- Disadvantages of numerical ratings:
 - Discrete categories such as star ratings do not provide fine-grained differentiation.
 - Annotators label inconsistently over time as they adjust to what they have seen.
 - Calibration: annotators interpret scores differently, e.g., some people consistently give extreme scores, others choose middling values.
 - Inconsistencies may be worse for continuous ratings.
- With comparative labelling, annotators select preferred and dis-preferred items from pairs or lists:
 - Allows for total sorting of items, i.e., fine-grained rankings.
 - Reduces inconsistencies over time and between annotators since annotators do not have to select a rating value.
 - Ranking requires more labels but annotators can often label preferences more quickly.



From Pairwise Labels to Rank Scores

- There are two popular *pooled* models that relate pairwise labels to the *utility* of the items, a score that can be used for ranking.

- Bradley-Terry Model:

$$p(a \succ b) = \frac{e^{u(a)}}{e^{u(a)} + e^{u(b)}}$$

- Thurstone-Mosteller Model:

$$p(a \succ b) = \Phi \left(\frac{u(a) - u(b)}{s} \right)$$

- Where Φ is the *probit* or cumulative distribution function of the standard normal distribution and s is the annotation noise variance.

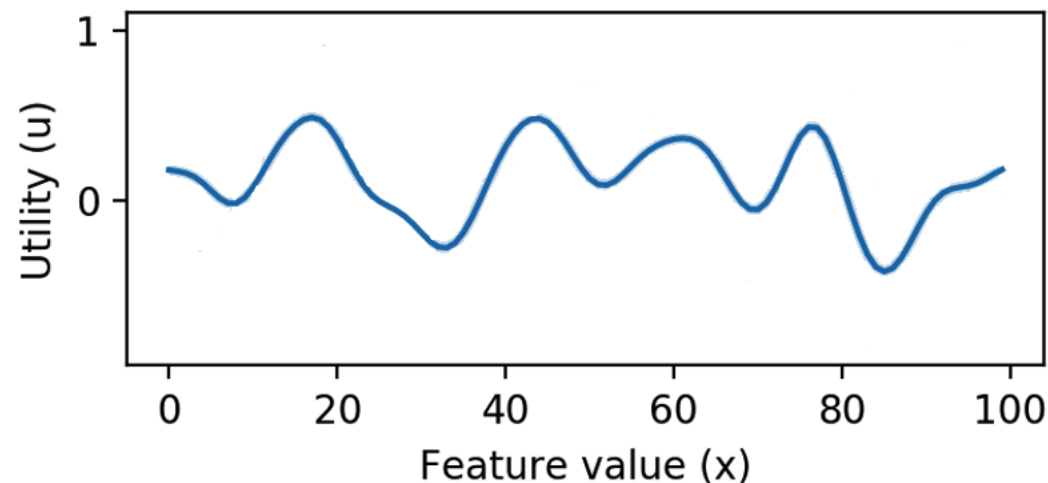
Best-Worst Scaling



- Annotators choose the best and worst items from a list (see Flynn and Marley, 2014).
- A simple scoring method approximates a maximum likelihood estimate of the BT model ranking:
 - $score(a) = count(a \text{ wins}) - count(a \text{ is worst})$
 - Danger: with a small number of comparisons, many items will have the same score even if they were compared with each other.
- Reformulate as pairwise comparisons to apply other pairwise models;
 - One pair for *best item* vs. *each* of the other items;
 - One pair for *worst item* vs. *each* of the other items;
 - Assumes that the sequence of choices is either best-then-worst or worst-then-best.

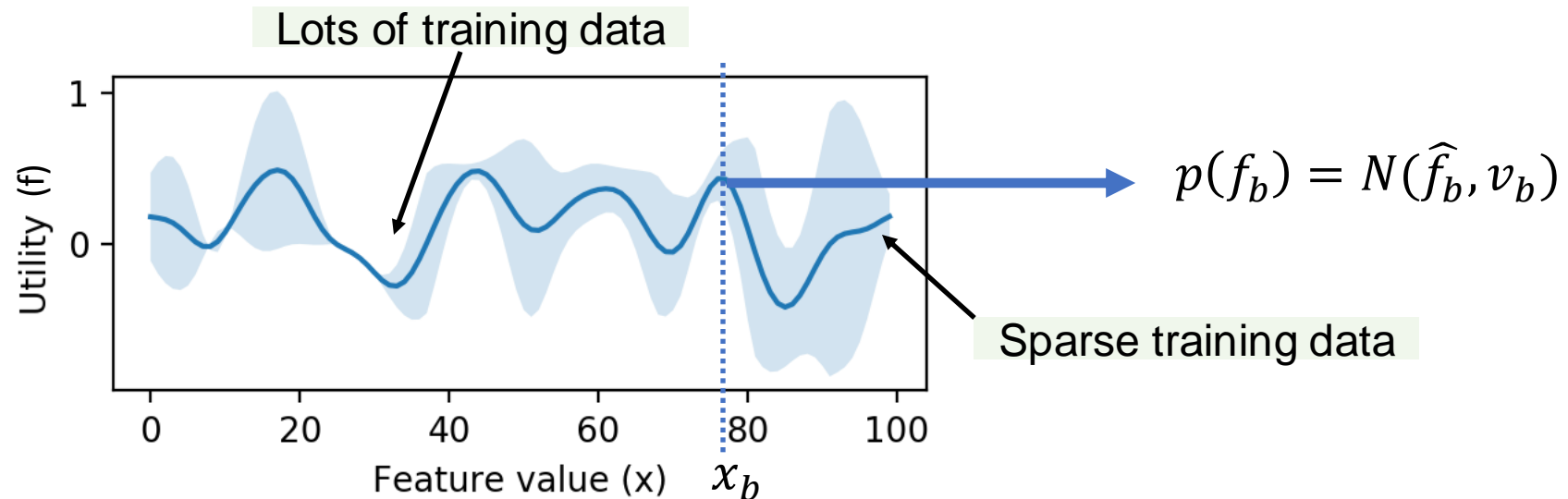
Sparsity of Pairwise Labels

- With exhaustive labelling, the number of pairs is $\mathcal{O}(n^2)$, where n is the number of data points.
- In practice, $\mathcal{O}(n)$ comparisons give good results.
- Using item features mitigates label sparsity: we can think of utility as a function of an item's features, e.g., of a document embedding.



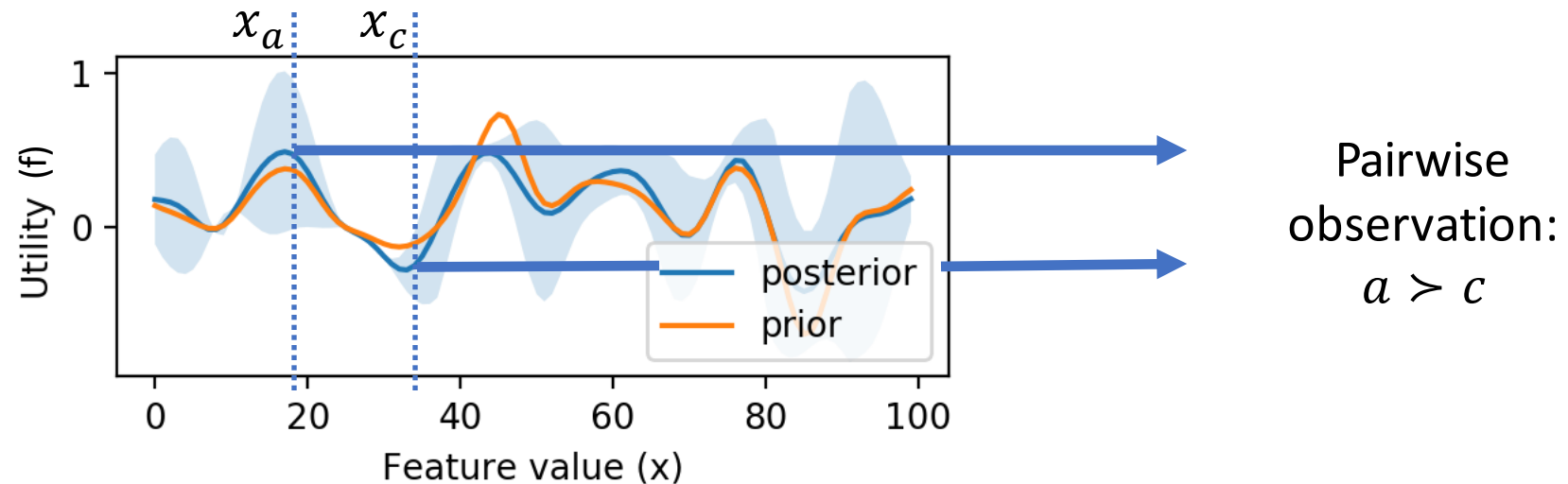
Gaussian Process Preference Learning (GPPL)

- **Gaussian processes (GPs)** are particularly well suited to modelling utility functions when learning from sparse, noisy pairwise labels.
- The Bayesian approach provides a posterior distribution over utilities which represents model confidence through variance.



Gaussian Process Preference Learning (GPPL)

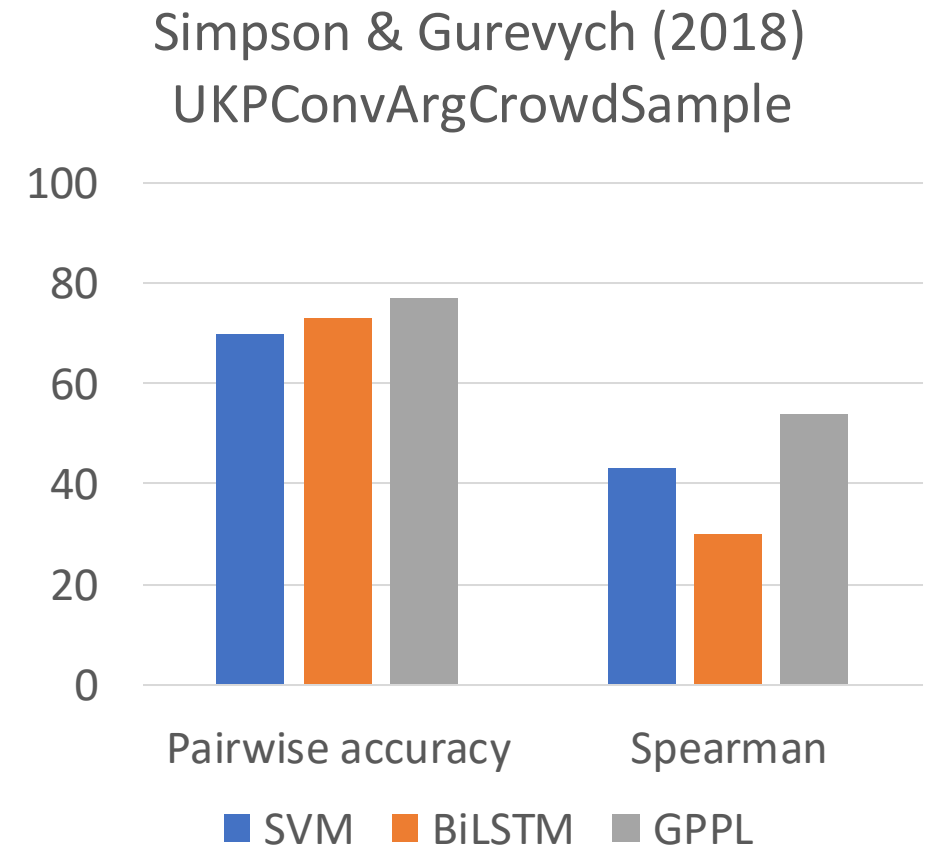
- We can think of inference as like updating the distribution for each label in the dataset in turn:



- Hence, GPs avoid making overly strong predictions where there is limited evidence, i.e., for items with few comparisons or noisy labels.

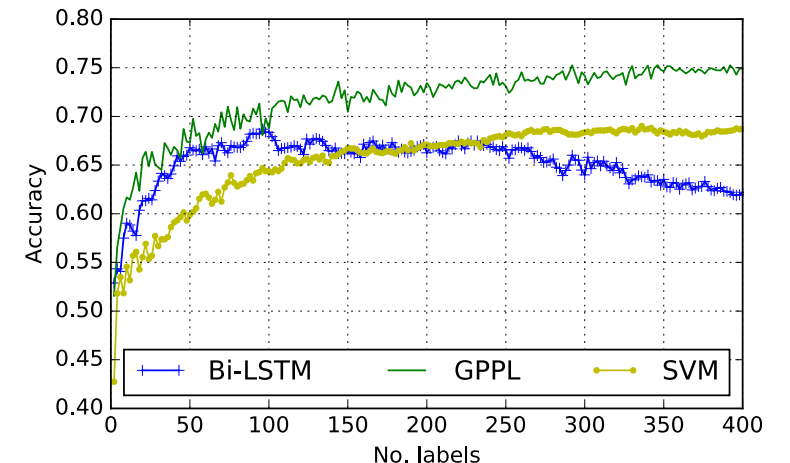
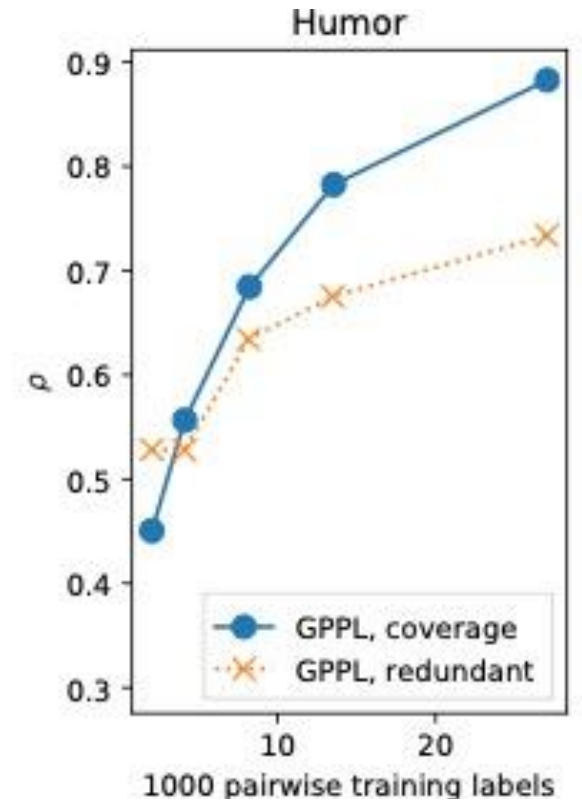
Preference Learning Comparison

- Task: ranking arguments by convincingness.
- 1 crowdsourced label per pair.
- SVM and BiLSTM trained to classify pairs of arguments.
- SVM and BiLSTM trained separately to predict rank scores.
- Right: predictive performance for unseen topics shows the benefit of Thurstone-Mosteller model + GPs.



Pairwise Labelling Strategies

- Two strategies for mitigating annotation errors when allocating pairs to annotators (Simpson et al., 2019):
 - a) **Redundant labelling**: each pair labelled by M annotators.
 - b) **Maximal coverage**: annotators label different pairs to maximise the number of different comparisons in the dataset.
- Active learning: uncertainty sampling can be very effective by leveraging the GP posterior variance (Simpson & Gurevych, 2018):



A Partially Pooled Model for Preferences

- Unpooled models capture accuracy and bias of annotators – how can we do this for preferences?
- Assume that an individual's utility function is offset from the 'true', or rather, *consensus*, utility function:

The diagram illustrates the equation $f(\mathbf{x}_a, \mathbf{u}_j) = t(\mathbf{x}_a) + v_j(\mathbf{x}_a)$. Three light green boxes are arranged horizontally: 'personal rating' on the left, 'consensus rating' in the middle, and 'user offset' on the right. A blue arrow points from the 'consensus rating' box to the 'personal rating' box, passing through the left side of the equation. Another blue arrow points from the 'consensus rating' box to the 'user offset' box, passing through the right side of the equation. The equation itself is centered between the 'personal rating' and 'user offset' boxes.


$$\text{personal rating} \leftarrow f(\mathbf{x}_a, \mathbf{u}_j) = t(\mathbf{x}_a) + v_j(\mathbf{x}_a) \rightarrow \text{user offset}$$

- The *offset*, $v_j(\mathbf{x}_a)$ represents the user j 's bias and is a function of the item's features, \mathbf{x}_a .
- This formulation is possible because the ground truth is a numerical value and observations are also (derived from) numerical values



A Partially Pooled Model for Preferences

- Both t and v_j can be modelled by Gaussian process.
- However, we rarely have enough labels per annotator to learn v_j with confidence across the feature space.
- So again, we can use a hierarchical method, where similar users are clustered:

$$f(\mathbf{x}_a, \mathbf{u}_j) = t(\mathbf{x}_a) + v_{c_j}(\mathbf{x}_a)$$


Cluster of user j

A Partially Pooled Model for Preferences

- User preferences may be more complicated than annotation behaviour for class labels: users may share similar preferences for subsets of items only.
- Following recommender systems, crowdGPPL (Simpson & Gurevych, 2020) uses matrix factorisation to identify *latent factors* common to multiple annotators:

$$f(\mathbf{x}_a, \mathbf{u}_j) = t(\mathbf{x}_a) + \sum_{c=1}^C v_c(\mathbf{x}_a) w_c(\mathbf{u}_j)$$

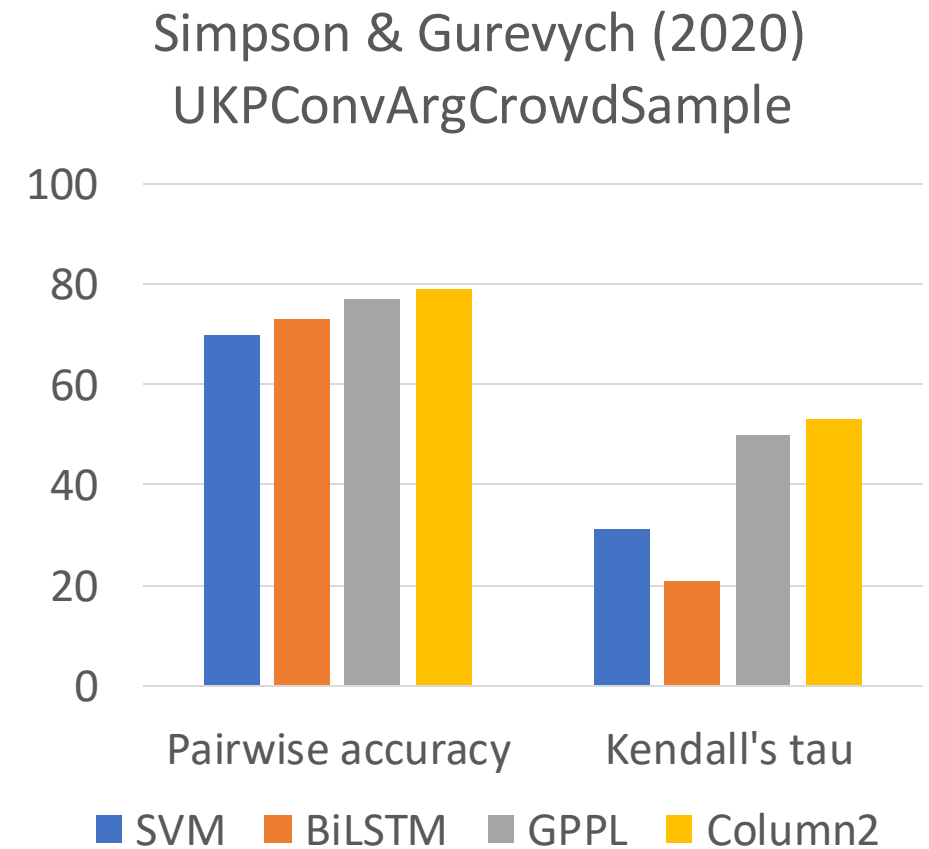
Offset for latent factor c



User j 's weight for latent factor c

Preference Learning Comparison

- Task: ranking arguments by convincingness.
- 1 crowdsourced label per pair.
- Right: predictive performance for unseen topics (predicting the consensus, t)
- More subjective or ambiguous tasks may benefit more from a partially pooled model like CrowdGPPL.



Aggregation with variational autoencoders

- Probabilistic models of the type we discussed allow us to encode our assumptions about the annotation process, but these assumptions have to be explicitly specified
 - Manually coming up with good specifications can be both cumbersome and limiting
- The alternative proposed by Yin et al. (2017) is to approach the task of aggregating the annotations with a variational auto-encoder
 - This framework allows them to use neural networks to learn more complex, non-linear relationships between the annotations and the ground truth

Aggregation with variational autoencoders

- The generative process of the annotations for an item follows a familiar set of assumptions:
 - First draw the true class c_i from a prior distribution $p_\theta(c_i)$
 - Then condition on the true class generate the annotations $p_\theta(y_i|c_i)$
- The objective is to find the parameters that maximize $p_\theta(y_{1:I})$, the marginal likelihood
- Yin et al. (2017) follow (Kingma and Welling, 2013) and introduce a variational approximation to the posterior of the true classes; let $q_\varphi(c_i|y_i)$ be this approximation, the marginal can be decomposed as follows:

$$\log p_\theta(y_{1:I}) = \sum_{i=1}^I \log p_\theta(y_i) = \sum_{i=1}^I D_{KL}(q_\varphi(c_i|y_i) || p_\theta(c_i|y_i)) + \mathcal{L}(\theta, \varphi|y_i)$$

Variational autoencoders

- It can be observed that by maximizing the marginal likelihood (the evidence) we are implicitly minimizing the KL divergence of the approximation from the true posterior of the true classes
- From a coding theory perspective $q_{\phi}(c_i|y_i)$ can be seen as a probabilistic encoder
 - Given the annotations of an item it produces a probability distribution for its true class
- Similarly, $p_{\theta}(y_i|c_i)$ can be interpreted as a probabilistic decoder
 - Given the true class of an item it species a distribution responsible for generating the annotations

Variational autoencoders

- The second term from the marginal decomposition lower bounds the evidence. Therefore, we can use this instead as our objective function to maximize:

$$\mathcal{L}(\theta, \varphi | y_i) = E_{q_{\varphi}(c_i | y_i)}[\log p_{\theta}(y_i | c_i)] - D_{KL}(q_{\varphi}(c_i | y_i) || p_{\theta}(c_i))$$

- Following the auto-encoder parlance, the first term in the bound above measures the expected reconstruction error
- The KL term can be interpreted as a regularizer for the parameters of the encoder, encouraging the approximate posterior for the true class of the items to be close to the prior

Back to aggregation

- The annotations collected for an item are arranged into a vector representation
- The vector contains J blocks, one for each coder, of K (the number of classes) elements corresponding to their one hot encoded annotation (or of zeros only when a coder did not annotate the item)
- Let us look at an example where we source 4 coders to assign an item into one of 3 classes; assuming coders 1, 3, and 4 assign the item into categories 1, 2, and 1 respectively, and coder 2 does not provide a response, we have:

$$y_i = \left[\underbrace{1 \ 0 \ 0}_{\text{coder 1}} \quad \underbrace{0 \ 0 \ 0}_{\text{coder 2}} \quad \underbrace{0 \ 1 \ 0}_{\text{coder 3}} \quad \underbrace{1 \ 0 \ 0}_{\text{coder 4}} \right]$$

The encoder part

- We can now introduce the functional form of the encoder and the decoder
- Starting with the former, Yin et al. (2017) propose to use a neural network with one fully connected layer

$$\pi_i = \textit{softmax}(y_i W_g)$$

- The output of the network specifies the encoder distribution for the true class of an item:

$$q(c_i | y_i) = \pi_{i, c_i}$$

The decoder part

- The decoder consist of another neural network and learns to reconstruct the annotations collected for an item from its (one hot encoded) true class
- As before, this network also has a single fully connected layer:

$$\beta_{i,c_i} = \textit{softmax}^*(c_i W_p)$$

- The vector of reconstructed annotations β_{i,c_i} has the same structure as the input vector, i.e., J blocks of K elements corresponding to a (reconstructed) one hot encoded annotation for each coder

*The softmax operator above is applied to each successive K -sized block of logits

A closer look at the decoder

- The matrix of weights W_p is responsible for learning the annotation behaviour of the coders for items of different classes
- The matrix consists of J successive blocks, one for each coder, of $K \times K$ class confusion matrices:

$$W_p = \left[\begin{array}{ccc|ccc} W_{p,1,1,1} & \dots & W_{p,1,1,K} & & W_{p,J,1,1} & \dots & W_{p,J,1,K} \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ W_{p,1,K,1} & \dots & W_{p,1,K,K} & & W_{p,J,K,1} & \dots & W_{p,J,K,K} \\ \hline & & \text{coder 1 class confusion weights} & & & & \text{coder J class confusion weights} \end{array} \right]$$

- According to the decoder, the probability for a coder j to annotate an item i whose true class is c_i with some class k is:

$$p\left(y_i^{(j,k)} = 1 \mid \beta_{i,c_i}\right) = \beta_{i,c_i}^{(j,k)}$$

Now back to the objective function

- $\mathcal{L}(\theta, \varphi|y_i) = E_{q_\varphi(c_i|y_i)}[\log p_\theta(y_i|c_i)] - D_{KL}(q_\varphi(c_i|y_i) || p_\theta(c_i))$
- The expected reconstruction error is approximated via sampling (Kingma and Welling, 2013)
- The practice is that at each training step you first sample a true class from the distribution described by the encoder:

$$c_i \sim \text{Categorical}(\pi_i)$$

- And then use that sample to evaluate the reconstruction term, which turns out to be the negative cross entropy of the reconstructed annotations relative to the actual input:

$$\log p_\theta(y_i|c_i) = \sum_{j,k} y_i^{(j,k)} \log \beta_{i,c_i}^{(j,k)}$$

Wrapping up the objective function

- $\mathcal{L}(\theta, \varphi|y_i) = E_{q_\varphi(c_i|y_i)}[\log p_\theta(y_i|c_i)] - D_{KL}(q_\varphi(c_i|y_i) || p_\theta(c_i))$
- In the KL term, the prior for the true class of the items is set by Yin et al. (2017) to be the distribution over the labels provided by the coders
- Additional regularization of the encoder and decoder weights can be considered to reduce the influence of those coders which labelled only a few items

Part 3. Learning with multiple annotators

Learning with human uncertainty

- The standard for training classifiers is to learn from examples labelled with the category they most likely belong to
- In doing so however any uncertainty the labellers had in their classification is ignored
- Peterson et al. (2019) suggest to collect a large number of reliable interpretations and to use the resulted distribution as a proxy for this uncertainty
- Such distributions offer a richer source of information compared to adjudicated labels – they indicate:
 - The strength of the consensus emerging from the annotations
 - The presence of ambiguity
 - And also how humans make mistakes

Training with ‘soft’ labels

- The training regime is straightforward, all that is required is to add a softmax output layer to your existing neural architecture
- The prediction for a training example is:

$$p_i = \text{softmax}(Wh_i)$$

- The loss for a training example measures the cross entropy of the prediction distribution relative to the target distribution over the annotations:

$$\mathcal{L}_i = H(t_i, p_i) = - \sum_{k=1}^K t_{i,k} \log p_{i,k}$$

Experiments: collecting the data

- Peterson et al. (2019) collected slightly above 500 thousand annotations for the test set of the popular CIFAR10 image classification corpus (Krizhevsky, 2009), for an average of about 50 annotations per image
- The distributions over the categories that result from the collected annotations offer a good representation of the coders' dissent
- Their mode (the majority vote) is on the gold (expert adjudicated) class in more than 99% of the cases

A hard and soft evaluation

- The authors train in a 10-fold cross validation setting 8 well known CNN based architectures for image classification using as target distributions either one hot encoded gold labels or the collected human labels
 - A standard accuracy based evaluation shows an average 1% increase in performance when using the human labels (83.5% vs. 84.5%)
- The collected labels have the quality to also be used to evaluate the models
 - Using a cross entropy based evaluation the models trained on the human labels register on average 29% lower values compared to those trained on one hot labels (0.5 vs. 0.7)
 - More significant gains in performance are obtained by using human labels when the evaluation is conducted on out of sample data, a result that highlights the benefits this approach to training brings to the generalization capability of the models

Humans are noisy

- The success of the approach presented previously relies on the quality of the target distributions, i.e., whether the collected annotations offer a good representation of the coders' dissent
- That may not always be the case, e.g., when their number is too low to get a good proxy for the human uncertainty, or when noise intervenes and skews the distributions
- For this purpose the ability to capture the accuracy and adjust for the bias of the coders while learning becomes essential

Aggregating the labels while training

- Learning how to distil the labels from noise is not only essential for training accurate models but also for adjudicating different interpretations
- The distilled labels leverage not only the annotation patterns produced by the coders as typically is the case in probabilistic models of annotation, but also the knowledge of the task accumulated by the model
- Jointly aggregating the labels and training a classifier leads to an improvement (over the disjoint approach) both in the performance of the classifier and in the quality of the adjudicated labels

Deep learning from crowds

- Rodrigues and Pereira (2018) proposed an approach which integrates the aggregation of the labels into the architecture of a neural network
- Simply put, the approach involves learning a mapping from the output of a neural network to the labels provided by the coders
- The mapping is intended to capture and adjust for the accuracy and bias of the coders
- The prediction distribution for an annotator on some training example is a function of the network's output layer:

$$p_{i,j} = f_j(h_i)$$

Classification mappings

- Different functions are put forward to model the classification behaviour of the coders
 - $f_j(h_i) = W_j h_i$ using a per coder (confusion) matrix of weights
 - $f_j(h_i) = w_j \odot h_i$ a per coder vector of class weights
 - $f_j(h_i) = b_j + h_i$ a per coder vector of class biases
- The loss for a training example accumulates the loss coming from each of its annotations
- The loss for an annotation is the cross entropy of the predicted label distribution relative to a one hot encoding of the actual label

Regression mappings

- When the labels are continuous such as in regression tasks the following mappings are considered:
 - $f_j(h_i) = s_j h_i$ using a per coder scale
 - $f_j(h_i) = b_j + h_i$ a per coder bias
 - $f_j(h_i) = s_j h_i + b_j$ a per coder scale and bias
- In this case the cross entropy loss is replaced by a mean squared error loss

Evaluation: classifying images

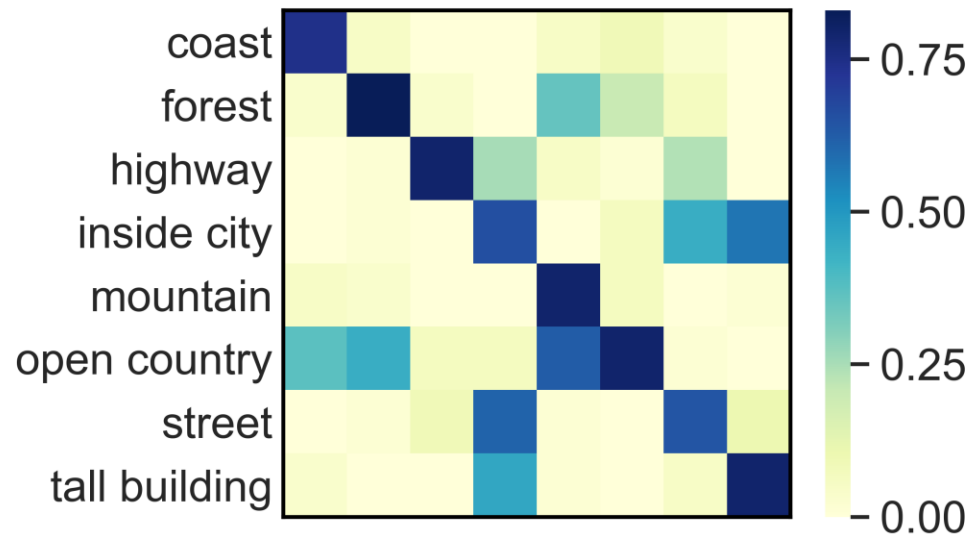
| Method | Accuracy |
|---|--------------|
| Training on the labels chosen by the majority of coders | 76.74 |
| Training on labels aggregated with (Dawid and Skene, 1979) | 80.79 |
| (Rodrigues and Pereira, 2018) with per coder class weights | 81.05 |
| (Rodrigues and Pereira, 2018) with per coder class weights and biases | 81.89 |
| (Rodrigues and Pereira, 2018) with per coder matrix of weights | 83.15 |
| Training on gold labels | 90.63 |

Evaluation: predicting movie ratings

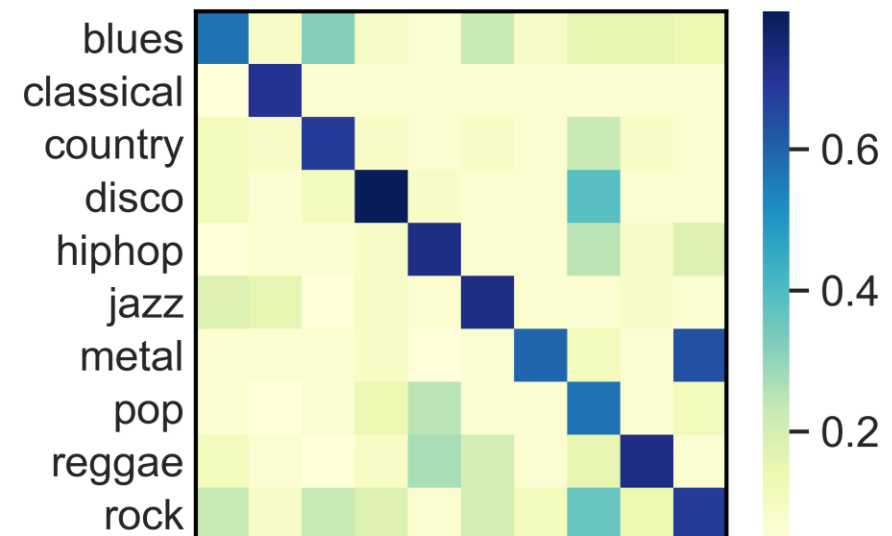
| Method | RMSE | MAE |
|--|-------------|-------------|
| Training on the mean ratings | 1.50 | 1.22 |
| (Rodrigues and Pereira, 2018) with per coder scale | 1.51 | 1.23 |
| (Rodrigues and Pereira, 2018) with per coder bias | 1.41 | 1.13 |
| (Rodrigues and Pereira, 2018) with per coder matrix of weights | 1.44 | 1.16 |
| Training on gold ratings | 1.33 | 1.05 |

Modelling common confusions

- Chu et al. (2021) argue that although coders have their own accuracy and biases when they annotate, they also share common confusions



(a) LabelMe



(b) Music

The general idea

- The annotations are assumed to be the result of either a pooled behaviour or the coders' own annotation behaviour
- Both the pooled and the individual annotation behaviours are modelled as usual, with a 'confusion' matrix of weights
- The decision on which annotation behaviour to use when predicting the annotations is made by a separate classifier on a coder and item basis based on their features

The framework, introduced more formally

- Let's start by embedding the coders:

$$u_j = f_u(e_j)$$

- Learn an embedding also for the items, based on their features:

$$v_i = f_v(x_i)$$

- Decide, for each annotation, based on the coder and item embeddings, which model of behaviour to use:

$$w_{i,j} = \text{sigmoid}(u_j^T v_i)$$

- Predict the annotation using either the coder's own behaviour f_j , or the pooled annotation behaviour f_g :

$$p_{i,j} = w_{i,j} * f_j(h_i) + (1 - w_{i,j}) * f_g(h_i)$$

Final details and results

- An optimizer minimizes the cross-entropy loss between the predicted and the observed annotations
- To encourage the pooled and the individual annotation behaviour of the coders to be different, the l2-norm on the difference between their weights is added to the loss
- The results indicate significant differences in accuracy, in particular on those classes commonly mistaken by the coders:

| Model | LabelMe | Music |
|---|---------|-------|
| Training on the labels chosen by the majority of coders | 79.83 | 72.53 |
| (Rodrigues and Pereira, 2018) with per coder scale and bias | 83.27 | 81.46 |
| (Chu et al., 2021) | 87.12 | 84.06 |

Part 4. Practical Session

Getting more acquainted with these models

- Hands-on with two of the core models presented here.
- PyIBCC: a Python implementation of the variational Bayes version of Dawid & Skene's (1979) model.
- **CrowdLayer:**
 - An additional layer that can be placed on top of a standard neural network to train the model directly from crowdsourced labels.
 - Written by Rodrigues and Pereira (2018) using Keras.
- Demonstrate how to:
 - Put your data in the right format and apply PyIBCC to obtain adjudicated labels.
 - Build and train a neural network with a crowd layer.
 - Adjust important hyperparameters of the methods.
- Link to the exercises: <https://sites.google.com/view/alma-tutorial>

A book is coming out soon on these topics, and more

- Silviu Paun, Ron Artstein, Massimo Poesio. “*Statistical Methods for Annotation Analysis*”. [Synthesis Lectures on Human Language Technologies \(morganclaypool.com\)](http://morganclaypool.com)

References

- Nguyen, A. T., Wallace, B. C., Li, J. J., Nenkova, A. & Lease, M. Aggregating and predicting sequence labels from crowd annotations. ACL 2017.
- Simpson, E. & Gurevych, I. A Bayesian Approach for Sequence Tagging with Crowds. EMNLP 2019.
- Rodrigues, F., Pereira, F., & Ribeiro, B. Sequence labeling with multiple annotators. Machine learning, 2014.
- Flynn, T. N., & Marley, A. A. Best-worst scaling: theory and methods. Handbook of choice modelling, 2014.
- Simpson, E., & Gurevych, I. Scalable Bayesian preference learning for crowds. Machine Learning, 2020.

References

- Bob Carpenter. Multilevel Bayesian models of categorical data annotation. Unpublished manuscript, 2008.
- Alexander Philip Dawid and Allan M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. Applied Statistics, 1979.
- Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. Learning whom to trust with MACE. NAACL, 2013.
- Silviu Paun, Jon Chamberlain, Udo Kruschwitz, Juntao Yu, and Massimo Poesio. A probabilistic annotation model for crowdsourcing coreference. EMNLP 2018.
- Filipe Rodrigues and Francisco C Pereira. Deep learning from crowds. AAAI 2018.

References

- Joshua C. Peterson, Ruairidh M. Battleday, Thomas L. Griths, and Olga Russakovsky. Human uncertainty makes classification more robust. ICCV 2019.
- Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast - but is it good? Evaluating non-expert annotations for natural language tasks. EMNLP 2008.
- Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. Community-based bayesian aggregation models for crowdsourcing. WWW 2014.
- Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. NIPS 2009.
- Li'ang Yin, Jianhua Han, Weinan Zhang, and Yong Yu. Aggregating crowd wisdoms with label-aware autoencoders. IJCAI 2017.

References

- Pablo G. Moreno, Antonio Artés-Rodríguez, Yee Whye Teh, and Fernando Perez-Cruz. Bayesian nonparametric crowdsourcing. JMLR, 2015.
- Massimo Poesio, Jon Chamberlain, Silviu Paun, Juntao Yu, Alexandra Uma, and Udo Kruschwitz. A crowdsourced corpus of multiple judgments and disagreement on anaphoric interpretation. NAACL 2019.
- Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. CONLL Shared Task, 2011.
- Chu, Zhendong, Jing Ma, and Hongning Wang. Learning from Crowds by Modeling Common Confusions. AAAI 2021.