Learning Whom to Trust with MACE

Dirk Hovy[1] **Taylor Berg-Kirkpatrick**[2] **Ashish Vaswani**[1] **Eduard Hovy**[3]
(1) Information Sciences Institute, University of Southern California, Marina del Rey
(2) Computer Science Division, University of California at Berkeley
(3) Language Technology Institute, Carnegie Mellon University, Pittsburgh
{dirkh,avaswani}@isi.edu, tberg@cs.berkeley.edu, hovy@cmu.edu

Abstract

Non-expert annotation services like Amazon's Mechanical Turk (AMT) are cheap and fast ways to evaluate systems and provide categorical annotations for training data. Unfortunately, some annotators choose bad labels in order to maximize their pay. Manual identification is tedious, so we experiment with an item-response model. It learns in an unsupervised fashion to a) identify which annotators are trustworthy and b) predict the correct underlying labels. We match performance of more complex state-of-the-art systems and perform well even under adversarial conditions. We show considerable improvements over standard baselines, both for predicted label accuracy and trustworthiness estimates. The latter can be further improved by introducing a prior on model parameters and using Variational Bayes inference. Additionally, we can achieve even higher accuracy by focusing on the instances our model is most confident in (trading in some recall), and by incorporating annotated control instances. Our system, MACE (Multi-Annotator Competence Estimation), is available for download[1].

1 Introduction

Amazon's MechanicalTurk (AMT) is frequently used to evaluate experiments and annotate data in NLP (Callison-Burch et al., 2010; Callison-Burch and Dredze, 2010; Jha et al., 2010; Zaidan and Callison-Burch, 2011). However, some turkers try to maximize their pay by supplying quick answers that have nothing to do with the correct label. We refer to this type of annotator as a *spammer*. In order to mitigate the effect of spammers, researchers typically collect multiple annotations of the same instance so that they can, later, use de-noising methods to infer the best label. The simplest approach is majority voting, which weights all answers equally. Unfortunately, it is easy for majority voting to go wrong. A common and simple spammer strategy for categorical labeling tasks is to always choose the same (often the first) label. When multiple spammers follow this strategy, the majority can be incorrect. While this specific scenario might seem simple to correct for (remove annotators that always produce the same label), the situation grows more tricky when spammers do not annotate consistently, but instead choose labels at random. A more sophisticated approach than simple majority voting is required.

If we knew whom to trust, *and when*, we could reconstruct the correct labels. Yet, the only way to be sure we know whom to trust is if we knew the correct labels ahead of time. To address this circular problem, we build a generative model of the annotation process that treats the correct labels as latent variables. We then use unsupervised learning to estimate parameters directly from redundant annotations. This is a common approach in the class of unsupervised models called *item-response* models (Dawid and Skene, 1979; Whitehill et al., 2009; Carpenter, 2008; Raykar and Yu, 2012). While such models have been implemented in other fields (e.g., vision), we are not aware of their availability for NLP tasks (see also Section 6).

Our model includes a binary latent variable that explicitly encodes if and *when* each annotator is spamming, as well as parameters that model the annotator's specific spamming "strategy". Impor-

[1]Available under http://www.isi.edu/publications/licensed-sw/mace/index.html

tantly, the model assumes that labels produced by an annotator when spamming are independent of the true label (though, a spammer can still produce the correct label by chance).

In experiments, our model effectively differentiates dutiful annotators from spammers (Section 4), and is able to reconstruct the correct label with high accuracy (Section 5), even under extremely adversarial conditions (Section 5.2). It does not require any annotated instances, but is capable of including varying levels of supervision via token constraints (Section 5.2). We consistently outperform majority voting, and achieve performance equal to that of more complex state-of-the-art models. Additionally, we find that thresholding based on the posterior label entropy can be used to trade off coverage for accuracy in label reconstruction, giving considerable gains (Section 5.1). In tasks where correct answers are more important than answering every instance, e.g., when constructing a new annotated corpus, this feature is extremely valuable. Our contributions are:

- We demonstrate the effectiveness of our model on real world AMT datasets, matching the accuracy of more complex state-of-the-art systems
- We show how posterior entropy can be used to trade some coverage for considerable gains in accuracy
- We study how various factors affect performance, including number of annotators, annotator strategy, and available supervision
- We provide MACE (Multi-Annotator Competence Estimation), a Java-based implementation of a simple and scalable unsupervised model that identifies malicious annotators and predicts labels with high accuracy

2 Model

We keep our model as simple as possible so that it can be effectively trained from data where annotator quality is unknown. If the model has too many parameters, unsupervised learning can easily pick up on and exploit coincidental correlations in the data. Thus, we make a modeling assumption that keeps our parameterization simple. We assume that an annotator always produces the correct label when

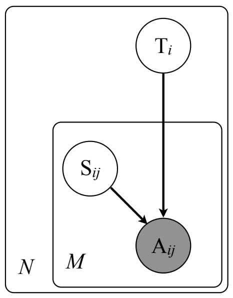


Figure 1: **Graphical model**: Annotator j produces label A_{ij} on instance i. Label choice depends on instance's true label T_i, and whether j is spamming on i, modeled by binary variable S_{ij}. $N = |instances|$, $M = |annotators|$.

```
for $i = 1 \dots N$ :
    $T_i \sim$ Uniform
    for $j = 1 \dots M$ :
        $S_{ij} \sim$ Bernoulli$(1 - \theta_j)$
        if $S_{ij} = 0$ :
            $A_{ij} = T_i$
        else :
            $A_{ij} \sim$ Multinomial$(\xi_j)$
```

Figure 2: **Generative process**: see text for description.

he tries to. While this assumption does not reflect the reality of AMT, it allows us to focus the model's power where it's important: explaining away labels that are not correlated with the correct label.

Our model generates the observed annotations as follows: First, for each instance i, we sample the true label T_i from a uniform prior. Then, for each annotator j we draw a binary variable S_{ij} from a Bernoulli distribution with parameter $1 - \theta_j$. S_{ij} represents whether or not annotator j is spamming on instance i. We assume that when an annotator is not spamming on an instance, i.e. $S_{ij} = 0$, he just copies the true label to produce annotation A_{ij}. If $S_{ij} = 1$, we say that the annotator is spamming on the current instance, and A_{ij} is sampled from a multinomial with parameter vector ξ_j. Note that in this case the annotation A_{ij} *does not* depend on the true label T_i. The annotations A_{ij} are observed, while the true labels T_i and the spamming indicators S_{ij} are unobserved. The graphical model is shown in Figure 1 and the generative process is described in Figure 2.

The model parameters are θ_j and ξ_j. θ_j specifies the probability of trustworthiness for annotator j (i.e. the probability that he is not spamming on any given instance). The learned value of θ_j will prove useful later when we try to identify reliable annotators (see Section 4). The vector ξ_j determines how annotator j behaves when he is spamming. An annotator can produce the correct answer even while spamming, but this can happen only by chance since the annotator must use the same multinomial parameters ξ_j across all instances. This means that we only learn annotator biases *that are <u>not</u> correlated with the correct label*, e.g., the strategy of the spammer who always chooses a certain label. This contrasts with previous work where additional parameters are used to model the biases that even dutiful annotators exhibit. Note that an annotator can also choose not to answer, which we can naturally accommodate because the model is generative. We enhance our generative model by adding Beta and Dirichlet priors on θ_j and ξ_j respectively which allows us to incorporate prior beliefs about our annotators (section 2.1).

2.1 Learning

We would like to set our model parameters to maximize the probability of the observed data, i.e., the marginal data likelihood:

$$P(A;\theta,\xi) = \sum_{T,S}\Big[\prod_{i=1}^{N} P(T_i)\cdot\prod_{j=1}^{M} P(S_{ij};\theta_j)\cdot P(A_{ij}|S_{ij},T_i;\xi_j)\Big]$$

where A is the matrix of annotations, S is the matrix of competence indicators, and T is the vector of true labels.

We maximize the marginal data likelihood using Expectation Maximization (EM) (Dempster et al., 1977), which has successfully been applied to similar problems (Dawid and Skene, 1979). We initialize EM randomly and run for 50 iterations. We perform 100 random restarts, and keep the model with the best marginal data likelihood. We smooth the M-step by adding a fixed value δ to the fractional counts before normalizing (Eisner, 2002). We find that smoothing improves accuracy, but, overall, learning is robust to varying δ, and set $\delta = \frac{0.1}{\text{num labels}}$.

We observe, however, that the average annotator proficiency is usually high, i.e., most annotators answer correctly. The distribution learned by EM, however, is fairly linear. To improve the correlation between model estimates and true annotator proficiency, we would like to add priors about the annotator behavior into the model. A straightforward approach is to employ Bayesian inference with Beta priors on the proficiency parameters, θ_j. We thus also implement Variational-Bayes (VB) training with symmetric Beta priors on θ_j and symmetric Dirichlet priors on the strategy parameters, ξ_j. Setting the shape parameters of the Beta distribution to 0.5 favors the extremes of the distribution, i.e., either an annotator tried to get the right answer, or simply did not care, but (almost) nobody tried "a little". With VB training, we observe improved correlations over all test sets with no loss in accuracy. The hyper-parameters of the Dirichlet distribution on ξ_j were clamped to 10.0 for all our experiments with VB training. Our implementation is similar to Johnson (2007), which the reader can refer to for details.

3 Experiments

We evaluate our method on existing annotated datasets from various AMT tasks. However, we also want to ensure that our model can handle adversarial conditions. Since we have no control over the factors in existing datasets, we create synthetic data for this purpose.

3.1 Natural Data

In order to evaluate our model, we use the datasets from (Snow et al., 2008) that use discrete label values (some tasks used continuous values, which we currently do not model). Since they compared AMT annotations to experts, gold annotations exist for these sets. We can thus evaluate the accuracy of the model as well as the proficiency of each annotator. We show results for word sense disambiguation (WSD: 177 items, 34 annotators), recognizing textual entailment (RTE: 800 items, 164 annotators), and recognizing temporal relation (Temporal: 462 items, 76 annotators).

3.2 Synthetic Data

In addition to the datasets above, we generate synthetic data in order to control for different factors. This also allows us to create a gold standard to which we can compare. We generate data sets with 100 items, using two or four possible labels.

For each item, we generate answers from 20 different annotators. The “annotators” are functions that return one of the available labels according to some strategy. Better annotators have a smaller chance of guessing at random.

For various reasons, usually not all annotators see or answer all items. We thus remove a randomly selected subset of answers such that each item is only answered by 10 of the annotators. See Figure 3 for an example annotation of three items.

	annotators									
items	–	0	0	1	–	0	–	–	0	–
	1	–	1	0	1	1	0	–	–	0
	–	–	0	–	0	1	–	0	–	0

Figure 3: Annotations: 10 annotators on three items, labels 1/0. Missing annotations marked ‘–’

3.3 Evaluations

First, we want to know which annotators to trust. We evaluate whether our model’s learned trustworthiness parameters θ_j can be used to identify these individuals (Section 4).

We then compare the label predicted by our model and by majority voting to the correct label. The results are reported as accuracy (Section 5). Since our model computes posterior entropies for each instance, we can use this as an approximation for the model’s confidence in the prediction. If we focus on predictions with high confidence (i.e., low entropy), we hope to see better accuracy, even at the price of leaving some items unanswered. We evaluate this trade-off in Section 5.1. In addition, we investigate the influence of the number of spammers and their strategy on the accuracy of our model (Section 5.2).

4 Identifying Reliable Annotators

One of the distinguishing features of the model is that it uses a parameter for each annotator to estimate whether or not they are spamming. Can we use this parameter to identify trustworthy individuals, to invite them for future tasks? Using majority voting often fails to find the correct label. This problem worsens when there are more than two labels. We need to take relative majorities into account or

	RTE	**Temporal**	**WSD**
raw agreement	0.78	0.73	0.81
Cohen’s κ	0.70	0.80	0.13
G-index	0.76	0.73	0.81
MACE-EM	0.87	0.88	0.44
MACE-VB $_{(0.5,0.5)}$	**0.91**	**0.90**	**0.90**

Table 1: Correlation with annotator proficiency: Pearson ρ of different methods for various data sets. MACE-VB’s trustworthiness parameter (trained with Variational Bayes with $\alpha = \beta = 0.5$) correlates best with true annotator proficiency.

break ties when two or more labels receive the same number of votes. This is deeply unsatisfying.

Instead, it is natural to apply some form of weighting. One approach is to assume that reliable annotators agree more with others than random annotators. Inter-annotator agreement is thus a good candidate to weigh the answers. There are various measures for inter-annotator agreement.

Tratz and Hovy (2010) compute the average agreement of each annotator and use it as a weight to identify reliable ones. Raw agreement can be directly computed from the data. It is related to majority voting, since it will produce high scores for all members of the majority class. Raw agreement is thus a very simple measure.

In contrast, Cohen’s κ corrects the agreement between two annotators for chance agreement. It is widely used for inter-annotator agreement in annotation tasks. We also compute the κ values for each pair of annotators, and average them for each annotator (similar to the approach in Tratz and Hovy (2010)). However, whenever one label is more prevalent (a common case in NLP tasks), κ overestimates the effect of chance agreement (Feinstein and Cicchetti, 1990) and penalizes disproportionately. The G-index (Gwet, 2008) corrects for the number of labels rather than chance agreement.

We compare these measures to our learned trustworthiness parameters θ_j in terms of their ability to select reliable annotators. A better measure should lend higher score to annotators who answer correctly more often than others. We thus compare the ratings of each measure to the true proficiency of each annotator. This is the percentage of annotated items the annotator answered correctly. Methods that can identify reliable annotators should highly correlate to the annotator's proficiency. Since the methods use different scales, we compute Pearson's ρ for the correlation coefficient, which is scale-invariant. The correlation results are shown in Table 1.

For all datasets, the learned θ_j correlates much more strongly with annotator proficiency than either κ or raw agreement. The variant trained with VB performs consistently better than standard EM training. The results show that our model not only finds the correct answer, but also that this method detects reliable annotators much better than any of the other measures, which are only loosely correlated to annotator proficiency.

The numbers for WSD also illustrate the low κ score resulting when all annotators (correctly) agree on a small number of labels. However, all inter-annotator agreement measures suffer from an even more fundamental problem: removing/ignoring annotators with low agreement will always improve the overall score, irrespective of the quality of their annotations. Worse, there is no natural stopping point: deleting the most egregious outlier always improves agreement, until we have only one annotator with perfect agreement left (Hovy, 2010). In contrast, MACE does not discard any annotators, but weighs their contributions differently. We are thus not losing information. This works well even under adversarial conditions (see Section 5.2).

5 Recovering the Correct Answer

The previous sections showed that our model reliably identifies trustworthy annotators. However, we also want to find the most likely correct answer. Figure 2 shows the accuracy of our model on various data sets from Snow et al. (2008). The model outperforms majority voting on both RTE and Temporal recognition sets. It performs as well as majority voting for the WSD task. This last set is somewhat of an exception, though, since almost all annotators are correct all the time, so majority voting is trivially correct. Still, we need to ensure that the model does not perform worse under such conditions. The results for RTE and Temporal data also rival those reported in Raykar and Yu (2012) and Carpenter (2008), yet were achieved with a much simpler model.

	RTE	Temporal	WSD
majority	0.90	0.93	0.99
Raykar/Yu 2012	0.93	0.94	—
Carpenter 2008	0.93	—	—
MACE-EM/VB	0.93	0.94	0.99
MACE-EM@90	0.95	0.97	0.99
MACE-EM@75	0.95	0.97	1.0
MACE-VB@90	0.96	0.97	1.0
MACE-VB@75	0.98	0.98	1.0

Table 2: Accuracy of different methods on data sets from (Snow et al., 2008). MACE-VB uses Variational Bayes training. Results @n use the $n\%$ items the model is most confident in (Section 5.1). Results below double line trade coverage for accuracy and are thus not comparable to upper half.

Carpenter (2008) models instance difficulty as a parameter. While it seems intuitively useful to model which items are harder than other, it increases the parameter space more than our trustworthiness variable. We achieve comparable performance without modeling difficulty, which greatly simplifies inference. The model of Raykar and Yu (2012) is more similar to our approach, in that it does not model item difficulty. However, it adds an extra step that learns priors from the estimated parameters. In our model, this is part of the training process. For more details on both models, see Section 6.

5.1 Trading Coverage for Accuracy

Sometimes, we want to produce an answer for every item (e.g., when evaluating a data set), and sometimes, we value good answers more than answering all items (e.g., when developing an annotated corpus). Jha et al. (2010) have demonstrated how to achieve better coverage (i.e., answer more items) by relaxing the majority voting constraints. Similarly, we can improve accuracy if we only select high quality annotations, even if this incurs lower coverage.

We provide a parameter in MACE that allows users to set a threshold for this trade-off: the model only returns a label for an instance if it is sufficiently confident in its answer. We approximate the model's confidence by the posterior entropy of each instance. However, entropy depends strongly on the specific makeup of the dataset (number of

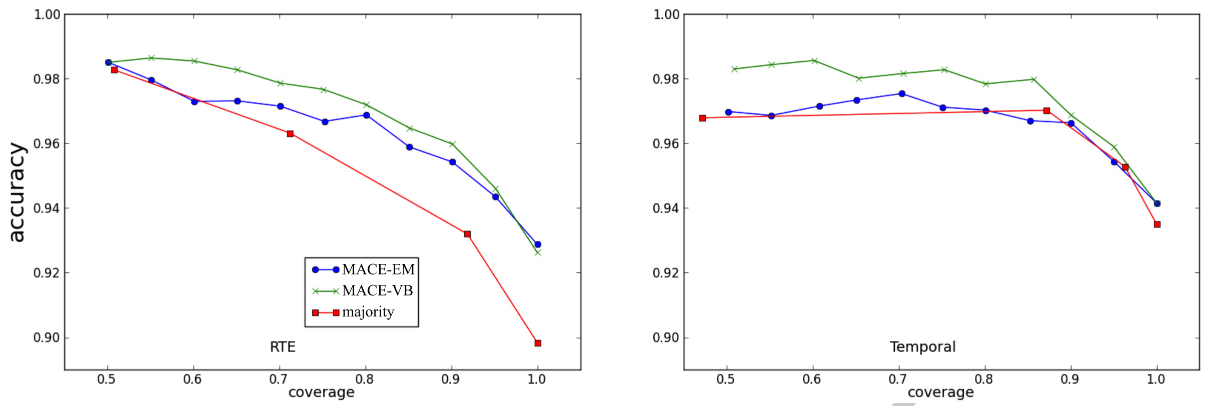


Figure 4: Tradeoff between coverage and accuracy for RTE (left) and temporal (right). Lower thresholds lead to less coverage, but result in higher accuracy.

annotators and labels, etc.), so it is hard for the user to set a specific threshold.

Instead of requiring an exact entropy value, we provide a simple thresholding between 0.0 and 1.0 (setting the threshold to 1.0 will include all items). After training, MACE orders the posterior entropies for all instances and selects the value that covers the selected fraction of the instances. The threshold thus roughly corresponds to coverage. It then only returns answers for instances whose entropy is below the threshold. This procedure is similar to precision/recall curves.

Jha et al. (2010) showed the effect of varying the relative majority required, i.e., requiring that at least n out of 10 annotators have to agree to count an item. We use that method as baseline comparison, evaluating the effect on coverage and accuracy when we vary n from 5 to 10.

Figure 4 shows the tradeoff between coverage and accuracy for two data sets. Lower thresholds produce more accurate answers, but result in lower coverage, as some items are left blank. If we produce answers for all items, we achieve accuracies of 0.93 for RTE and 0.94 for Temporal, but by excluding just the 10% of items in which the model is least confident, we achieve accuracies as high as 0.95 for RTE and 0.97 for Temporal. We omit the results for WSD here, since there is little headroom and they are thus not very informative. Using Variational Bayes inference consistently achieves higher results for the same coverage than the standard implementation. Increasing the required majority also improves accuracy, although not as much, and the loss in coverage is larger and cannot be controlled. In contrast, our method allows us to achieve better accuracy at a smaller, controlled loss in coverage.

5.2 Influence of Strategy, Number of Annotators, and Supervision

Adverse Strategy We showed that our model recovers the correct answer with high accuracy. However, to test whether this is just a function of the annotator pool, we experiment with varying the trustworthiness of the pool. If most annotators answer correctly, majority voting is trivially correct, as is our model. What happens, however, if more and more annotators are unreliable? While some agreement can arise from randomness, majority voting is bound to become worse—can our model overcome this problem? We set up a second set of experiments to test this, using synthetic data. We choose 20 annotators and vary the amount of good annotators among them from 0 to 10 (after which the trivial case sets in). We define a *good* annotator as one who answers correctly 95% of the time.[2] Adverse annotators select their answers randomly or always choose a certain value (minimal annotators). These are two frequent strategies of spammers.

For different numbers of labels and varying percentage of spammers, we measure the accuracy of our model and majority voting on 100 items, averaged over 10 runs for each condition. Figure

[2]The best annotators on the Snow data sets actually found the correct answer 100% of the time.

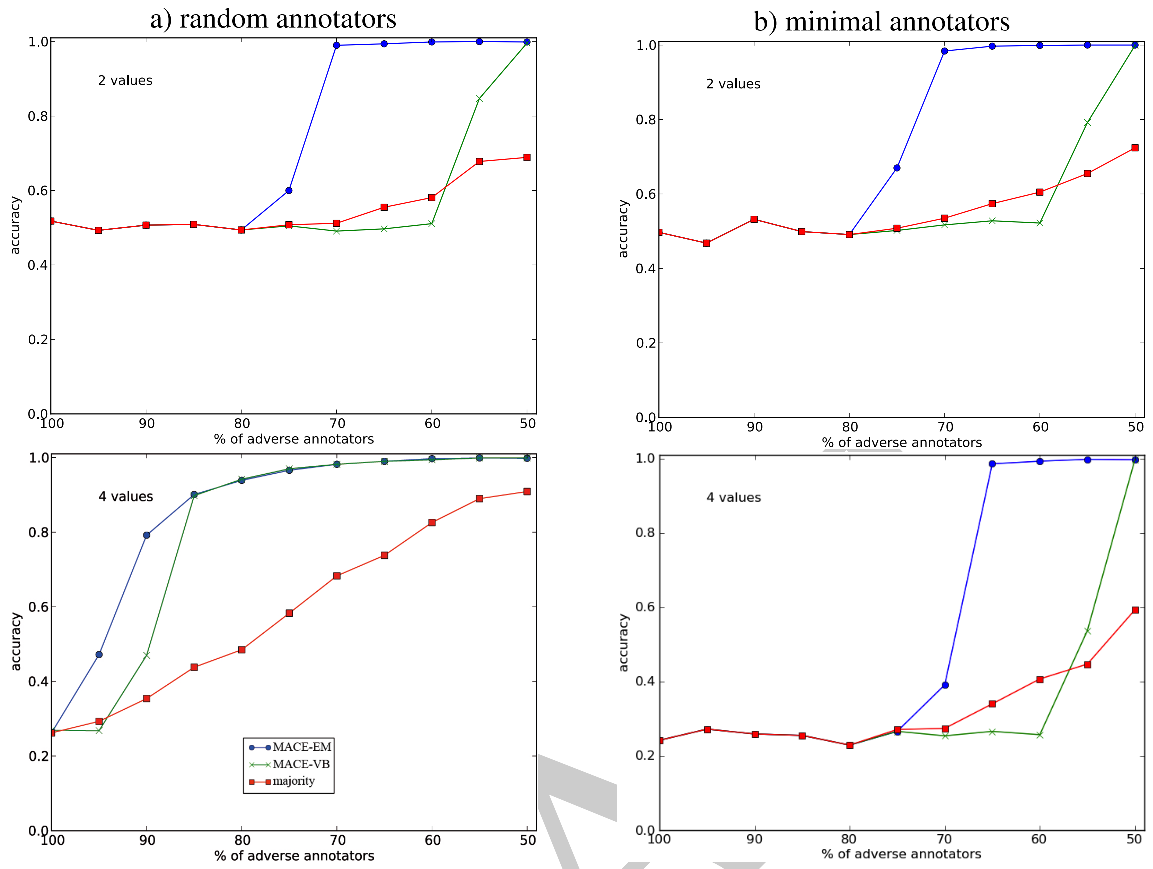


Figure 5: Influence of adverse annotator strategy on label accuracy (y-axis). Number of possible labels varied between 2 (top row) and 4 (bottom row). Adverse annotators either choose at random (a) or always select the first label (b). MACE needs fewer good annotators to recover the correct answer.

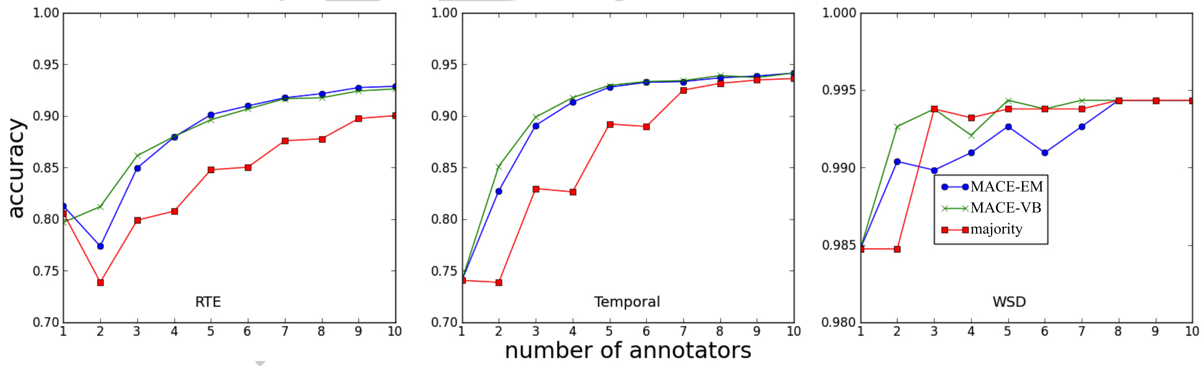


Figure 6: Varying number of annotators: effect on prediction accuracy. Each point averaged over 10 runs. Note different scale for WSD.

5 shows the effect of annotator proficiency on both majority voting and our method for both kinds of spammers. Annotator pool strategy affects majority voting more than our model. Even with few good annotators, our model learns to dismiss the spammers as noise. There is a noticeable point on each graph where MACE diverges from the majority voting line. It thus reaches good accuracy much faster than majority voting, i.e., with fewer good annotators. This divergence point happens earlier with

more label values when adverse annotators label randomly. In general, random annotators are easier to deal with than the ones always choosing the first label. Note that in cases where we have a majority of adversarial annotators, VB performs worse than EM, since this condition violates the implicit assumptions we encoded with the priors in VB. Under these conditions, setting different priors to reflect the annotator pool should improve performance.

Obviously, both of these pools are extremes: it is unlikely to have so few good or so many malicious annotators. Most pools will be somewhere in between. It does show, however, that our model can pick up on reliable annotators even under very unfavorable conditions. The result has a practical upshot: AMT allows us to require a minimum rating for annotators to work on a task. Higher ratings improve annotation quality, but delay completion, since there are fewer annotators with high ratings. The results in this section suggest that we can find the correct answer even in annotator pools with low overall proficiency. We can thus waive the rating requirement and allow more annotators to view and complete the task. This considerably speeds up completion.

Number of Annotators Figure 6 shows the effect different numbers of annotators have on accuracy. As we increase the number of annotators, MACE and majority voting achieve better accuracy results. We note that majority voting results level or even drop when going from an odd to an even number. In these cases, the new annotator does not improve accuracy if it goes with the previous majority (i.e., going from 3:2 to 4:2), but can force an error when going against the previous majority (i.e., from 3:2 to 3:3), by creating a tie. MACE-EM and MACE-VB dominate majority voting for RTE and Temporal. For WSD, the picture is less clear, where majority voting dominates when there are fewer annotators. Note that the differences are minute, though (within 1 percentage point). For very small pool sizes (< 3), MACE-VB outperforms both other methods.

Amount of Supervision So far, we have treated the task as completely unsupervised. MACE does not require any expert annotations in order to achieve high accuracy. However, we often have annotations for some of the items. These annotated data points are usually used as control items (by removing annotators that answer them incorrectly). If such annotated data is available, we would like to make use of it. We include an option that lets users supply annotations for some of the items, and use this information as token constraints in the E-step of training. In those cases, the model does not need to estimate the correct value, but only has to adjust the trust parameter. This leads to improved performance.[3]

We explore for RTE and Temporal how performance changes when we vary the amount of supervision in increments of 5%.[4] We average over 10 runs for each value of n, each time supplying annotations for a random set of n items. The baseline uses the annotated label whenever supplied, otherwise the majority vote, with ties split at random.

Figure 7 shows that, unsurprisingly, all methods improve with additional supervision, ultimately reaching perfect accuracy. However, MACE uses the information more effectively, resulting in higher accuracy for a given amount of supervision. This gain is more pronounced when only little supervision is available.

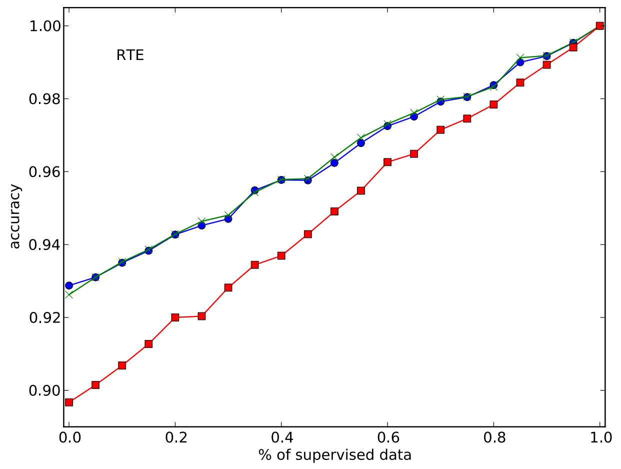
6 Related Research

Snow et al. (2008) and Sorokin and Forsyth (2008) showed that Amazon's MechanicalTurk use in providing non-expert annotations for NLP tasks. Various models have been proposed for predicting correct annotations from noisy non-expert annotations and for estimating annotator trustworthiness. These models divide naturally into two categories: those that use expert annotations for supervised learning (Snow et al., 2008; Bian et al., 2009), and completely unsupervised ones. Our method falls into the latter category because it learns from the redundant non-expert annotations themselves, and makes no use of expertly annotated data.

Most previous work on unsupervised models belongs to a class called "Item-response models", used in psychometrics. The approaches differ with respect to which aspect of the annotation process they choose to focus on, and the type of annotation

[3]If we had annotations for all items, accuracy would be perfect and require no training.

[4]Given the high accuracy for the WSD data set even in the fully unsupervised case, we omit the results here.



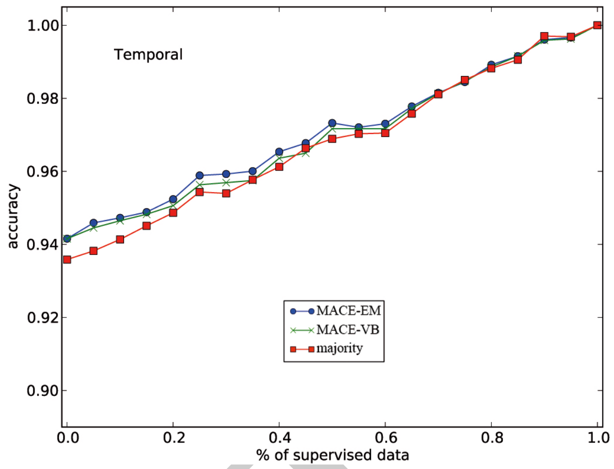


Figure 7: Varying the amount of supervision: effect on prediction accuracy. Each point averaged over 10 runs. MACE uses supervision more efficiently.

task they model. For example, many methods explicitly model annotator bias in addition to annotator competence (Dawid and Skene, 1979; Smyth et al., 1995). Our work models annotator bias, but only when the annotator is suspected to be spamming.

Other methods focus modeling power on instance difficulty to learn not only which annotators are good, but which instances are hard (Carpenter, 2008; Whitehill et al., 2009). In machine vision, several models have taken this further by parameterizing difficulty in terms of complex features defined on each pairing of annotator and annotation instance (Welinder et al., 2010; Yan et al., 2010). While such features prove very useful in vision, they are more difficult to define for the categorical problems common to NLP. In addition, several methods are specifically tailored to annotation tasks that involve ranking (Steyvers et al., 2009; Lee et al., 2011), which limits their applicability in NLP.

The method of Raykar and Yu (2012) is most similar to ours. Their goal is to identify and filter out annotators whose annotations are not correlated with the gold label (dubbed *spammers*). They define a function of the learned parameters that is useful for identifying spammers, and then use this function to build a prior. In contrast, we use simple priors, but incorporate a model parameter that explicitly represents the probability that an annotator is spamming. Our simple model achieves the same accuracy on gold label predictions as theirs.

7 Conclusion

We provide a Java-based implementation, MACE, that recovers correct labels with high accuracy, and reliably identifies trustworthy annotators. In addition, it provides a threshold to control the accuracy/coverage trade-off and can be trained with standard EM or Variational Bayes EM. MACE works fully unsupervised, but can incorporate token constraints via annotated control items. We show that even small amounts help improve accuracy.

Our model focuses most of its modeling power on learning trustworthiness parameters, which are highly correlated with true annotator reliability (Pearson ρ 0.9). We show on real-world and synthetic data sets that our method is more accurate than majority voting, even under adversarial conditions, and as accurate as more complex state-of-the-art systems. Focusing on high-confidence instances improves accuracy considerably. MACE is freely available for download under `http://www.isi.edu/publications/licensed-sw/mace/index.html`.

Acknowledgements

The authors would like to thank Chris Callison-Burch, Victoria Fossum, Stephan Gouws, Marc Schulder, Nathan Schneider, and Noah Smith for invaluable discussions, as well as the reviewers for their constructive feedback.

References

Jiang Bian, Yandong Liu, Ding Zhou, Eugene Agichtein, and Hongyuan Zha. 2009. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *Proceedings of the 18th international conference on World wide web*, pages 51–60. ACM.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12, Los Angeles, June. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, Uppsala, Sweden, July. Association for Computational Linguistics.

Bob Carpenter. 2008. Multilevel Bayesian models of categorical data annotation. *Unpublished manuscript*.

A. Philip Dawid and Allan M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, pages 20–28.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Jason Eisner. 2002. An interactive spreadsheet for teaching the forward-backward algorithm. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 10–18. Association for Computational Linguistics.

Alvan R. Feinstein and Domenic V. Cicchetti. 1990. High agreement but low kappa: I. the problems of two paradoxes. *Journal of Clinical Epidemiology*, 43(6):543–549.

Kilem Li Gwet. 2008. Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48.

Eduard Hovy. 2010. Annotation. A Tutorial. In *48th Annual Meeting of the Association for Computational Linguistics*.

Mukund Jha, Jacob Andreas, Kapil Thadani, Sara Rosenthal, and Kathleen McKeown. 2010. Corpus creation for new genres: A crowdsourced approach to pp attachment. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 13–20. Association for Computational Linguistics.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305.

Michael D. Lee, Mark Steyvers, Mindy de Young, and Brent J. Miller. 2011. A model-based approach to measuring expertise in ranking tasks. In L. Carlson, C. Hölscher, and T.F. Shipley, editors, *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, Austin, TX. Cognitive Science Society.

Vikas C. Raykar and Shipeng Yu. 2012. Eliminating Spammers and Ranking Annotators for Crowdsourced Labeling Tasks. *Journal of Machine Learning Research*, 13:491–518.

Padhraic Smyth, Usama Fayyad, Mike Burl, Pietro Perona, and Pierre Baldi. 1995. Inferring ground truth from subjective labelling of Venus images. *Advances in neural information processing systems*, pages 1085–1092.

Rion Snow, Brendan O'Connor, Dan Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics.

Alexander Sorokin and David Forsyth. 2008. Utility data annotation with Amazon Mechanical Turk. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPRW '08*, pages 1–8. IEEE.

Mark Steyvers, Michael D. Lee, Brent Miller, and Pernille Hemmer. 2009. The wisdom of crowds in the recollection of order information. *Advances in neural information processing systems*, 23.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 678–687. Association for Computational Linguistics.

Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. 2010. The multidimensional wisdom of crowds. In *Neural Information Processing Systems Conference (NIPS)*, volume 6.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in Neural Information Processing Systems*, 22:2035–2043.

Yan Yan, Rómer Rosales, Glenn Fung, Mark Schmidt, Gerardo Hermosillo, Luca Bogoni, Linda Moy, and

Jennifer Dy. 2010. Modeling annotator expertise: Learning when everybody knows a bit of something. In *International Conference on Artificial Intelligence and Statistics*.

Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1220–1229, Portland, Oregon, USA, June. Association for Computational Linguistics.