Detecting annotation noise in automatically labelled data

Ines Rehbein **Josef Ruppenhofer**
IDS Mannheim/University of Heidelberg, Germany
Leibniz Science Campus "Empirical Linguistics and Computational Language Modeling"
rehbein@cl.uni-heidelberg.de, ruppenhofer@ids-mannheim.de

Abstract

We introduce a method for error detection in automatically annotated text, aimed at supporting the creation of high-quality language resources at affordable cost. Our method combines an unsupervised generative model with human supervision from active learning. We test our approach on in-domain and out-of-domain data in two languages, in AL simulations and in a real world setting. For all settings, the results show that our method is able to detect annotation errors with high precision and high recall.

1 Introduction

Until recently, most of the work in Computational Linguistics has been focussed on standard written text, often from newswire. The emergence of two new research areas, Digital Humanities and Computational Sociolinguistics, have however shifted the interest towards large, noisy text collections from various sources. More and more researchers are working with social media text, historical data, or spoken language transcripts, to name but a few. Thus the need for NLP tools that are able to process this data has become more and more apparent, and has triggered a lot of work on domain adaptation and on developing more robust preprocessing tools. Studies are usually carried out on large amounts of data, and thus fully manual annotation or even error correction of automatically prelabelled text is not feasible. Given the importance of identifying noisy annotations in automatically annotated data, it is all the more surprising that up to now this area of research has been severely understudied.

This paper addresses this gap and presents a method for error detection in *automatically* labelled text. As test cases, we use POS tagging and Named Entity Recognition, both standard preprocessing steps for many NLP applications. However, our approach is general and can also be applied to other classification tasks.

Our approach is based on the work of Hovy et al. (2013) who develop a generative model for estimating the reliability of multiple annotators in a crowdsourcing setting. We adapt the generative model to the task of finding errors in *automatically* labelled data by integrating it in an active learning (AL) framework. We first show that the approach of Hovy et al. (2013) on its own is not able to beat a strong baseline. We then present our integrated model, in which we impose human supervision on the generative model through AL, and show that we are able to achieve substantial improvements in two different tasks and for two languages.

Our contributions are the following. We provide a novel approach to error detection that is able to identify errors in *automatically* labelled text with high precision *and* high recall. To the best of our knowledge, our method is the first that addresses this task in an AL framework. We show how AL can be used to guide an unsupervised generative model, and we will make our code available to the research community.[1] Our approach works particularly well in out-of-domain settings where no annotated training data is yet available.

2 Related work

Quite a bit of work has been devoted to the identifcation of errors in *manually* annotated corpora (Eskin, 2000; van Halteren, 2000; Kveton and Oliva, 2002; Dickinson and Meurers, 2003; Loftsson, 2009; Ambati et al., 2011).

[1] Our code is available at http://www.cl.uni-heidelberg.de/~rehbein/resources.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1160–1170
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1107

Several studies have tried to identify trustworthy annotators in crowdsourcing settings (Snow et al., 2008; Bian et al., 2009), amongst them the work of Hovy et al. (2013) described in Section 3. Others have proposed selective relabelling strategies when working with non-expert annotators (Sheng et al., 2008; Zhao et al., 2011).

Manual annotations are often inconsistent and annotation errors can thus be identified by looking at the variance in the data. In contrast to this, we focus on detecting errors in *automatically* labelled data. This is a much harder problem as the annotation errors are systematic and consistent and therefore hard to detect. Only a few studies have addressed this problem. One of them is Rocio et al. (2007) who adapt a multiword unit extraction algorithm to detect automatic annotation errors in POS tagged corpora. Their semi-automatic method is geared towards finding (a small number of) high frequency errors in large datasets, often caused by tokenisation errors. Their algorithm extracts sequences that have to be manually sorted into linguistically sound patterns and erroneous patterns.

Loftsson (2009) tests several methods for error detection in POS tagged data, one of them based on the predictions of an ensemble of 5 POS taggers. Error candidates are those tokens for which the predictions of all ensemble taggers agree but that diverge from the manual annotation. This simple method yields a precision of around 16% (no. of true positives amongst the error candidates), but no information is given about the recall of the method, i.e. how many of the errors in the corpus have been identified. Rehbein (2014) extends the work of Loftsson (2009) by training a CRF classifier on the output of ensemble POS taggers. This results in a much higher precision, but with low recall (for a precision in the range of 50-60% they report a recall between 10-20%).

Also related is work that addresses the issue of learning in the presence of annotation noise (Reidsma and Carletta, 2008; Beigman and Klebanov, 2009; Bekker and Goldberger, 2016). The main difference to our work lies in its different focus. While our focus is on identifying errors with the goal of improving the quality of an existing *language resource*, their main objective is to improve the accuracy of a *machine learning system*.

In the next section we describe the approach of Hovy et al. (2013) and present our adaptation for semi-supervised error detection that combines Bayesian inference with active learning.

Algorithm 1 AL with variational inference

```
Input: classifier predictions A
 1: for 1 ... n iterations do
 2:     procedure GENERATE(A)
 3:         for i = 1 ... n classifiers do
 4:             T_i ~ Uniform
 5:             for j = 1 ... n instances do
 6:                 S_ij ~ Bernoulli(1 − θ_j)
 7:                 if S_ij = 0 then
 8:                     A_ij = T_i
 9:                 else
10:                     A_ij ~ Multinomial(ξ_j)
11:                 end if
12:             end for
13:         end for
14:         return posterior entropies E
15:     end procedure
16:     procedure ACTIVELEARNING(A)
17:         rank J → max(E)
18:         for j = 1 ... n instances do
19:             Oracle → label(j);
20:             select random classifier i;
21:             update model prediction for i(j);
22:         end for
23:     end procedure
24: end for
```

3 Method

3.1 Modelling human annotators

Hovy et al. (2013) develop a generative model for Multi-Annotator Competence Estimation (MACE) to determine which annotators to trust in a crowdsourcing setting (Algorithm 1, lines 2-15). MACE implements a simple graphical model where the input consists of all annotated instances I by a set of J annotators. The model generates the observed annotations A as follows. The (unobserved) "true" label T_i is sampled from a uniform prior, based on the assumption that the annotators always try to predict the correct label and thus the majority of the annotations should, more often than not, be correct. The model is unsupervised, meaning that no information on the real gold labels is available.

To model each annotator's behaviour, a binary variable S_{ij} (also unobserved) is drawn from a Bernoulli distribution that describes whether annotator j is trying to predict the correct label for instance i or whether s/he is just spamming (a behaviour not uncommon in a crowdsourcing setting). If S_{ij} is 0, the "true" label T_i is used to generate the annotation A_{ij}. If S_{ij} is 1, the predicted label A_{ij} for instance i comes from a multinomial distribution with parameter vector ξ_j.

The model parameter θ_j can be interpreted as a "trustworthiness" parameter that describes the probability that annotator j predicts the correct label. ξ_j, on the other hand, contains information about the actual behaviour of annotator j in the case that the annotator is not trying to predict the correct label.

The model parameters are learned by maximizing the marginal likelihood of the observed data, using Expectation Maximization (EM) (Dempster et al., 1977) and Bayesian variational inference. Bayesian inference is used to provide the model with priors on the annotators' behaviour and yields improved correlations over EM between the model estimates and the annotators' proficiency while keeping accuracy high. For details on the implementation and parameter settings refer to Hovy et al. (2013) and Johnson (2007).

We adapt the model of Hovy et al. (2013) and apply it to the task of error detection in *automatically* labelled text. To that end, we integrate the variational model in an active learning (AL) setting, with the goal of identifying as many errors as possible while keeping the number of instances to be checked as small as possible. The tasks we chose in our experiments are POS tagging and NER, but our approach is general and can easily be applied to other classification tasks.

3.2 Active learning

Active learning (Cohn et al., 1996) is a semi-supervised framework where a machine learner is trained on a small set of carefully selected instances that are informative for the learning process, and thus yield the same accuracy as when training the learner on a larger set of randomly chosen examples. The main objective is to save time and money by minimising the need for manual annotation. Many different measures of informativeness as well as selection strategies for AL have been proposed in the literature, amongst them *query-by-committee* learning (Seung et al., 1992).

The *query-by-committee* (QBC) approach uses a classifier ensemble (or committee) and selects the instances that show maximal disagreement between the predictions of the committee members. These instances are assumed to provide new information for the learning process, as the classifiers are most unsure about how to label them. The selected instances are then presented to the *oracle* (the human annotator), to be manually disambiguated and added to the training data. Then the classifier committee is retrained on the extended training set and the next AL iteration starts.

The query-by-committee strategy calls to mind previous work on error detection in *manually* labelled text that made use of disagreements between the predictions of a classifier ensemble and the manually assigned tag, to identify potential annotation errors in the data (Loftsson, 2009). This approach works surprisingly well, and the trade-off between precision and recall can be balanced by adding a threshold (i.e. by considering all instances where at least N of the ensemble classifiers disagree with the manually assigned label). Loftsson (2009) reports a precision of around 16% for using a committee of five POS taggers to identify annotation errors (see section 2).

Let us assume we follow this approach and apply a tagger with an average accuracy of 97% to a corpus with 100,000 tokens. We can then expect around 3,000 incorrectly tagged instances in the data. Trying to identify these with a precision of 16% means that when looking at 1,000 instances of potential errors, we can only expect to see around 160 true positive cases, and we would have to check a large amount of data in order to correct a substantial part of the annotation noise. This means that this approach is not feasible for correcting large automatically annotated data.

It is thus essential to improve precision *and* recall for error detection, and our goal is to minimise the number of instances that have to be manually checked while maximizing the number of true errors in the candidate set. In what follows we show how we can achieve this by using active learning to guide variational inference for error detection.

3.3 Guiding variational inference with AL

Variational inference is a method from calculus where the posterior distribution over a set of unobserved random variables Y is approximated by a variational distribution $Q(Y)$. We start with some observed data X (a set of predictions made by our committee of classifiers) The distribution of the true labels $Y = \{y_1, y_2, ..., y_n\}$ is unknown.

As it is too difficult to work with the posterior $p(y|x)$, we try to approximate it with a much simpler distribution $q(y)$ which models y for each observed x. To that end, we define a family Q of distributions that are computationally easy to work with, and pick the q in Q that best approximates the posterior, where $q(y)$ is called the variational approximation to the posterior $p(y|x)$.

For computing variational inference, we use the implementation of Hovy et al. (2013)[2] who jointly optimise p and q using variational EM. They alternate between adjusting q given the current p (E-step) and adjusting p given the current q (M-step). In the E-step, the objective is to find the q that minimises the divergence between the two distributions, $D(q||p)$. In the M-step, we keep q fixed and try to adjust p. The two steps are repeated until convergence.

We extend the model for use in AL as follows (Algorithm 1). We start with the predictions from a classifier ensemble and learn a variational inference model on the data (lines 2-15). We then use the posterior entropies according to the current model, and select the c instances with the highest entropies for manual validation. These instances are presented to the oracle who assigns the true label. We save the predictions made by the human annotator and, in the next iteration, use them in the variational E-step as a prior to guide the learning process. In addition, we randomly pick one of the classifiers and update its prediction by replacing the classifier's prediction with the label we obtained from the oracle.[3] In the next iteration, we train the variational model on the updated predictions. By doing this, we also gradually improve the quality of the input to the variational model.

In a typical AL approach, the main goal is to improve the *classifiers' accuracy* on new data. In contrast to that, our approach aims at increasing *precision and recall* for error detection in *automatically* labelled data, and thus at minimising the time needed for manual correction. Please note that in our model we do *not* need to retrain the classifiers used for predicting the labels but only retrain the model that determines which of the classifiers' predictions we can trust. This is crucial as it saves time and makes it easy to integrate the approach in a realistic scenario with a real human annotator in the loop.

4 Data and setup

In our first experiment (§5.1) we want to assess the benefits of our approach for finding POS errors in standard newspaper text (in-domain setting) where we have plenty of training data. For this setting, we use the English Penn Treebank, annotated with parts-of-speech, for training and testing.

In the second experiment (§5.2) we apply our method in an out-of-domain setting where we want to detect POS errors in text from new domains where no training data is yet available (out-of-domain setting). For this we use the Penn Treebank as training data, and test our models on data from the English Web treebank (Bies et al., 2012).

To test our method on a different task and a new language, we apply it to Named Entity Recognition (NER) (experiment 3, §5.3), using out-of-domain data from the Europarl corpus.[4] The data was created by Faruqui and Pado (2010) and includes the first two German Europarl session transcripts, manually annotated with NER labels according to the CoNLL 2003 annotation guidelines (Tjong Kim Sang and De Meulder, 2003).

The first three experiments are simulation studies. In our last experiment (§5.4), we show that our method also works well in a real AL scenario with a human annotator in the loop. For this we use the out-of-domain setting from the second experiment and let the annotators correct POS errors in two web genres (*answers, weblogs*) from the English Web treebank.

4.1 Tools for preprocessing

For the POS tagging experiments, we use the following taggers to predict the labels:

- bi-LSTM-aux (Plank et al., 2016)
- HunPos (Halácsy et al., 2007)
- Stanford postagger (Toutanova et al., 2003)
- SVMTool (Giménez and Màrquez, 2004)
- TreeTagger (Schmid, 1999)
- TWeb (Ma et al., 2014)
- Wapiti (Lavergne et al., 2010)

The taggers implement a range of different algorithms, including HMMs, decision trees, SVMs, maximum entropy and neural networks. We train the taggers on subsets of 20,000 sentences extracted from the standard training set of the PTB (sections 00-18)[5] and use the development and test set (sections 19-21 and 22-24) for testing. The training times of the taggers vary considerably, ranging from a few seconds (HunPos) to several

[2]MACE is available for download from http://www.isi.edu/publications/licensed-sw/mace

[3]We also experimented with updating more than one classifier, which resulted in lower precision and recall. We take this as evidence for the importance of keeping the variance in the predictions high.

[4]The NER taggers have been trained on written German data from the HGC and DeWaC corpora (see §4.1).

[5]For taggers that use a development set during training, we also extract the dev data from sections 00-18 of the PTB.

hours. This is a problem for the typical AL setting where it is crucial not to keep the human annotators waiting for the next instance while the system retrains. A major advantage of our setup is that we do not need to retrain the baseline classifiers as we only use them once, for preprocessing, before the actual error detection starts.

For the NER experiment, we use tools for which pretrained models for German are available, namely GermaNER (Benikova et al., 2015), and the StanfordNER system (Finkel and Manning, 2009) with models trained on the HGC and the DeWaC corpus (Baroni et al., 2009; Faruqui and Padó, 2010).[6]

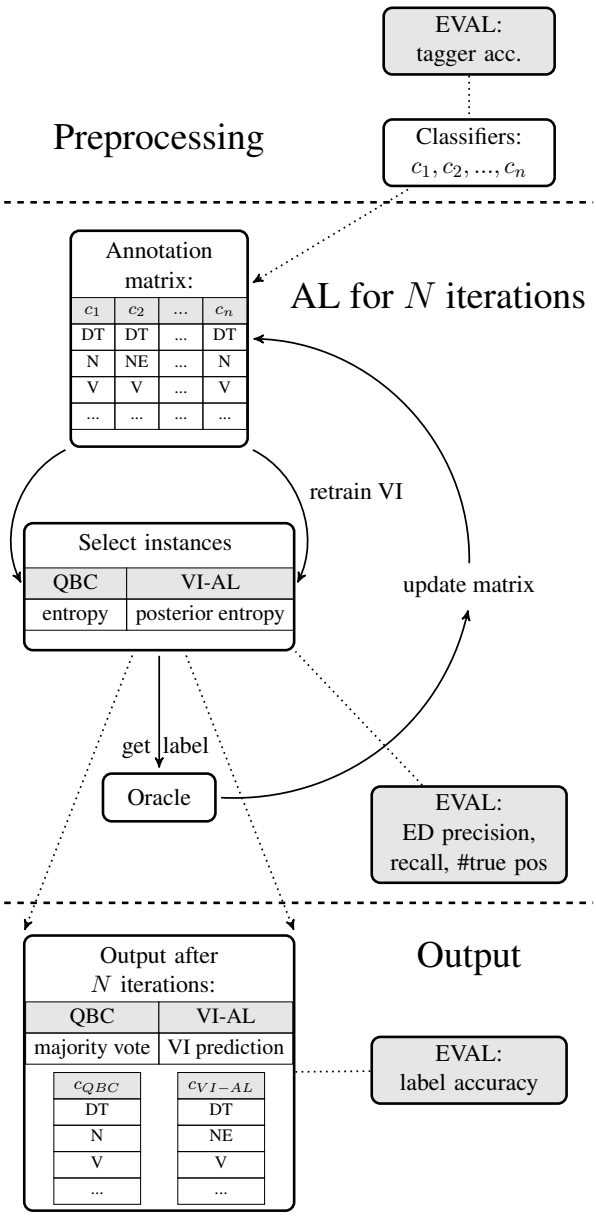
4.2 Evaluation measures

We report results for different evaluation measures to asses the usefulness of our method. First, we report **tagger accuracy** on the data, obtained during preprocessing (figure 1). This corresponds to the accuracy of the labels in the corpus *before* error correction (baseline accuracy). **Label accuracy** measures the accuracy of the labels in the corpus *after* N iterations of error correction. Please note that we do not retrain the tools used for preprocessing, but assess the quality of the data after N iterations of manual inspection and correction.

We also report precision and recall for the error detection itself. **True positives (tp)** refers to the number of instances selected for correction during AL that were actual annotation errors. We compute **Error detection (ED) precision** as the number of true positives divided by the number of all instances selected for error correction during N iterations of AL, and **recall** as the ratio of correctly identified errors to all errors in the data.

4.3 Baseline accuracies

Table 1 shows the accuracies for the individual POS taggers used in experiments 1, 2 and 4. Please note that this is not a fair comparison as each tagger was trained on a *different* randomly sampled subset of the data and, crucially, we did not optimise any of the taggers but used default settings in all experiments.[7] The accuracies of the



EVAL	*Evaluation measures used in the experiments*
tagger acc	Accuracy of preprocessing classifiers on the data.
label acc	Label accuracy in the corpus after N iterations of AL.
true pos	No. of instances selected for correction that are true errors.
ED prec	No. of true pos. / all instances selected for error correction.
recall	Correctly identified errors / all errors in the corpus.

Figure 1: Error detection procedure and overview over different evaluation measures for assessing the quality of error identification.

baseline taggers vary between 94-97%, with an average accuracy of 95.8%. The majority baseline yields better results than the best individual tagger, with an accuracy of 97.3%. Importantly, the predictions made by the variational inference model (MACE) are in the same range as the majority baseline and thus *do not improve over the*

[6]To increase the number of annotators we use an older version of the StanfordNER (2009-01-16) and a newer version (2015-12-09), with both the DeWaC and HGC models, resulting in a total of 5 annotators for the NER task.

[7]Please note that the success of our method relies on the variation in the ensemble predictions, and thus improving the accuracies for preprocessing is not guaranteed to improve precision for the error detection task.

Tagger	Acc.
bilstm	97.00
hunpos	96.18
stanford	96.93
svmtool	95.86
treetagger	94.35
tweb	95.99
wapiti	94.52
avg.	95.83
majority vote	**97.28**
MACE	97.27

Table 1: **Tagger accuracies** for POS taggers trained on subsamples of the WSJ with 20,000 tokens (for the majority vote, ties were broken randomly).

majority vote on the *automatically* labelled data.

To be able to run the variational inference model in an AL setting, we limit the size of the test data (the size of the pre-annotated data to be corrected) to batches of 5,000 tokens. This allows us to reduce the training time of the variational model and avoid unnecessary waiting times for the oracle.

For NER (experiment 3), in contrast to POS tagging, we have a much smaller label set with only 5 labels (PER, ORG, LOC, MISC, O), and a highly skewed distribution where most of the instances belong to the negative class (O). To ensure a sufficient number of NEs in the data, we increase the batch size and use the whole out-of-domain testset with 4,395 sentences in the experiment.[8] The overall accuracies of the different NER models are all in the range of 97.7-98.6%. Results for individual classes, however, vary considerably between the different models.

5 Results

5.1 Experiment 1: In-domain setting

In our first experiment, we explore the benefits of our AL approach to error detection in a setting where we have a reasonably large amount of training data, and where training and test data come from the same domain (in-domain setting).

We implement two selection strategies. The first one is a *Query-by-Committee* approach (QBC) where we use the disagreements in the predictions of our tagger ensemble to identify potential errors. For each instance i, we compute the entropy over the predicted labels M by the 7 taggers and select the N instances with the highest entropy (Equation 1).

$$H = -\sum_{m=1}^{M} P(y_i = m) \log P(y_i = m) \quad (1)$$

For each selected instance, we then replace the label predicted by majority vote with the gold label. Please note that the selected instances might already have the correct label, and thus the replacement does not necessarily increase accuracy but only does so when the algorithm selects a true error. We then evaluate the accuracy of the majority predictions after updating the N instances ranked highest for entropy[9] (figure 1).

We compare the QBC setting to our integrated approach where we guide the generative model with human supervision. Here the instances are selected according to their posterior entropy as assigned by the variational model, and after being disambiguated by the oracle, the predictions of a randomly selected classifier are updated with the oracle tags. We run the AL simulation for 500 iterations[10] and select one new instance in each iteration. After replacing the predicted label for this instance by the gold label, we retrain the variational model and select the next instance, based on the new posterior probabilities learned on the modified dataset. We refer to this setting as VI-AL.

	QBC		VI-AL	
N	**label acc**	**ED prec**	**label acc**	**ED prec**
0	97.58	-	97.56	-
100	97.84	13.0	98.42	**41.0**
200	97.86	7.0	98.90	**33.0**
300	97.90	5.3	99.16	**26.3**
400	97.82	3.0	99.26	**21.0**
500	97.92	3.4	99.34	**17.6**

Table 2: **Label accuracies** on 5,000 tokens of WSJ text after N iterations, and precision for error detection (ED prec).

Table 2 shows POS tag accuracies (lab-acc) *after* N iterations of active learning. For the QBC setting, we see a slight increase in label accuracy of 0.3% (from 97.6 to 97.9) after manually validating 10% of the instances in the data. For the first 100 instances, we see a precision of 13% for error

[8] This is possible because, given the lower number of class labels, the training time for the VI-AL model for NER is much shorter than for the POS data.

[9] Please recall that, in contrast to a traditional QBC active learning approach, we do not retrain the classifiers but only update the labels predicted by the classifiers.

[10] We stopped after 500 iterations as this was enough to detect nearly all errors in the WSJ data.

	answer	email	newsg.	review	weblog
bilstm	85.5	84.2	86.5	86.9	89.6
hun	88.5	87.4	89.2	89.7	92.2
stan	**89.0**	88.1	**89.9**	90.7	93.0
svm	87.4	86.1	88.2	88.8	91.3
tree	86.8	85.6	87.1	88.7	87.4
tweb	88.2	87.1	88.5	89.3	92.0
wapiti	85.2	82.4	84.6	86.5	87.3
avg.	87.2	85.8	87.7	88.7	90.4
major.	87.4	**88.8**	89.1	90.9	93.8
MACE	87.4	88.6	89.1	**91.0**	**93.9**

Table 3: **Tagger accuracies** on different web genres (trained on the WSJ); avg. accuracy, accuracy for majority vote (major.), and accuracy for MACE.

N	answer	email	newsg	review	weblog
0	87.4	88.6	89.1	91.0	93.9
100	88.9	90.0	90.4	92.2	95.2
200	90.3	91.1	91.3	93.4	96.2
300	91.6	92.2	92.0	94.4	97.2
400	92.9	93.3	92.8	95.4	97.5
500	93.9	94.0	93.5	96.0	97.8
600	94.8	94.9	93.9	96.5	97.9
700	95.6	95.6	94.1	96.9	98.0
800	96.2	95.9	94.7	97.3	98.4
900	96.7	96.2	94.9	97.7	98.6
1000	97.0	96.8	95.1	97.9	98.6

Table 4: Increase in POS **label accuracy** on the web genres (5,000 tokens) after N iterations of error correction with VI-AL.

detection. In the succeeding iterations, the precision slowly decreases as it gets harder to identify new errors. We even observe a slight decrease in label accuracy after 400 iterations that is due to the fact that ties are broken randomly and thus the vote for the same instance can vary between iterations.

Looking at the AL setting with variational inference, we also see the highest precision for identifying errors during the first 100 iterations. However, the precision for error dection is more than 3 times as high as for QBC (41% vs. 13%), and we are still able to detect new errors during the last 100 iterations. This results in an increase in POS label accuracy in the corpus from 97.56% to 99.34%, a near perfect result.

To find out what error types we were not able to identify, we manually checked the remaining 33 errors that we failed to detect in the first 500 iterations. Most of those are cases where an adjective (JJ) was mistaken for a past participle (VBN).

(2) Companies were **closed**$_{JJ/VBN}$ yesterday

Manning (2011), who presents a categorization of the type of errors made by a state-of-the-art POS tagger on the PTB, refers to the error type in example (2) as *underspecified/unclear*, a category that he applies to instances where "the tag is underspecified, ambiguous, or unclear in the context". These cases are also hard to disambiguate for human annotators, so it is not surprising that our system failed to detect them.

5.2 Experiment 2: Out-of-domain setting

In the second experiment, we test how our approach performs in an out-of-domain setting. For this, we use the English Web treebank (Bies et al., 2012), a corpus of over 250,000 words of English weblogs, newsgroups, email, reviews and question-answers manually annotated for parts-of-speech and syntax. Our objective is to develop and test a method for error detection that can also be applied to out-of-domain scenarios for creating and improving language resources when *no in-domain training data is available*. We thus abstain from retraining the taggers on the web data and use the tools and models from experiment 1 (§5.1) as is, trained on the WSJ. As the English Web treebank uses an extended tagset with additional tags for URLs and email addresses etc., we allow the oracle to assign new tags unknown to the preprocessing classifiers. In a traditional AL setting, this would not be possible, as all class labels have to be known from the start. In our setting, however, this can be easily implemented.

For each web genre, we extract samples of 5,000 tokens and run an active learning simulation with 500 iterations, where in each iteration one new instance is selected and disambiguated. After each iteration, we update the variational model and the predictions of a randomly selected classifier, as described in Section 5.1.

Table 3 shows the performance of the WSJ-trained taggers on the web data. As expected, the results are much lower than the ones from the in-domain setting. This allows us to explore the behaviour of our error detection approach under different conditions, in particular to test our approach on tag predictions of a lower quality. The last three rows in Table 3 give the average tagger accuracy, the accuracy for the majority vote for the ensemble (not to be confused with QBC), and the accuracy we get when using the predictions from the variational model *without AL* (MACE).

	QBC			VI-AL		
N	**# tp**	**ED prec**	**rec**	**# tp**	**ED prec**	**rec**
100	85	**85.0**	**13.5**	75	75.0	11.9
200	148	**74.0**	**23.5**	146	73.0	23.2
300	198	66.0	31.4	212	**70.7**	**33.6**
400	239	59.7	37.9	278	**69.5**	**44.1**
500	282	56.4	44.8	323	**64.6**	**51.3**
600	313	52.2	49.7	374	**62.3**	**59.4**
700	331	47.3	52.5	412	**58.9**	**65.4**
800	355	44.4	56.3	441	**55.1**	**70.0**
900	365	40.6	57.9	465	**51.7**	**73.8**
1000	371	37.1	58.9	484	**48.4**	**76.8**

Table 5: No. of true positives (# tp), precision (ED prec) and recall for error detection on 5,000 tokens from the *answers* set after N iterations.

We can see that the majority baseline often, but not always succeeds in beating the best individual tagger. Results for MACE are more or less in the same range as the majority vote, same as in experiment 1, but *do not improve over the baseline*.

Next, we employ AL in the out-of-domain setting (Tables 4, 5 and 6). Table 4 shows the increase in POS label accuracy for the five web genres after running N iterations of AL with variational inference (VI-AL). Table 5 compares the results of the two selection strategies, QBC and VI-AL, on the *answers* subcorpus after an increasing number of AL iterations.[11] Table 6 completes the picture by showing results for error detection for all web genres, for QBC and VI-AL, after inspecting 10% of the data (500 iterations).

Table 4 shows that using VI-AL for error detection results in a substantial increase in POS label accuracy for all genres. VI-AL still detects new errors after a high number of iterations, without retraining the ensemble taggers. This is especially useful in a setting where no labelled target domain data is yet available.

Table 5 shows the number of true positives amongst the selected error candidates as well as precision and recall for error detection for different stages of AL on the *answers* genre. We can see that during the early learning stages, both selection strategies have a high precision and QBC beats VI-AL. After 200 iterations it becomes more difficult to detect new errors, and the precision for both methods decreases. The decrease, however, is much slower for VI-AL, leading to higher precision after the initial rounds of training, and the gap in results becomes more and more pronounced.

[11] Due to space restrictions, we can only report detailed results for one web genre. Results for the other web genres follow the same trend (see Tables 4 and 6).

	QBC			VI-AL		
	# tp	**ED prec**	**rec**	**# tp**	**ED prec**	**rec**
answer	282	56.4	44.8	323	**64.6**	**51.3**
email	264	**52.8**	**47.1**	261	52.2	46.6
newsg.	195	39.0	36.0	214	**42.8**	**39.6**
review	227	45.4	49.7	255	**51.0**	**55.8**
weblog	166	33.2	54.6	196	**39.2**	**64.5**

Table 6: No. of true positives (# tp), precision (ED prec) and recall for error detection on 5,000 tokens after 500 iterations on all web genres.

After 600 iterations, VI-AL beats QBC by more than 10%, thus resulting in a lower number of instances that have to be checked to obtain the same POS accuracy in the final dataset. Looking at recall, we see that by manually inspecting 10% of the data VI-AL manages to detect more than 50% of all errors, and after validating 20% of the data, we are able to eliminate 75% of all errors in the corpus. In contrast, QBC detects less than 60% of the annotation errors in the dataset.

In the out-of-domain setting where we start with low-quality POS predictions, we are able to detect errors in the data with a much higher precision than in the in-domain setting, where the number of errors in the dataset is much lower. Even after 1,000 iterations, the precision for error detection is close to 50% in the answers data.

Table 6 shows that the same trend appears for the other web genres, where we observe a substantially higher precision and recall when guiding AL with variational inference (VI-AL). Only on the email data are the results below the ones for QBC, but the gap is small.

5.3 Experiment 3: A new task (and language)

We now want to test if the approach generalises well to other classification tasks, and also to new languages. To that end, we apply our approach to the task of Named Entity Recognition (NER) on German data (§4).

Table 7 shows results for error detection for NER. In comparison to the POS experiments, we observe a much lower recall, for both QBC and VI-AL. This is due to the larger size of the NER testset which results in a higher absolute number of errors in the data. Please bear in mind that recall is computed as the ratio of correctly identified errors to all errors in the testset (here we have a total of 110,405 tokens in the test set which means that we identified >35% of all errors by querying *less than 1% of the data*). Also note that the overall number of errors is higher in the QBC setting (1,756

	QBC			VI-AL		
N	**# tp**	**ED prec**	**rec**	**# tp**	**ED prec**	**rec**
100	54	54.0	3.1	76	**76.0**	**4.7**
200	113	56.5	6.4	155	**77.5**	**9.6**
300	162	54.0	9.2	217	**72.3**	**13.4**
400	209	52.2	11.9	297	**74.2**	**18.2**
500	274	54.8	15.6	352	**70.4**	**22.3**
600	341	56.8	19.4	409	**68.2**	**25.5**
700	406	58.0	23.1	452	**64.6**	**27.8**
800	480	60.0	27.3	483	**60.4**	**29.8**
900	551	**61.2**	31.4	512	56.9	**31.9**
1000	617	**61.7**	35.1	585	58.5	**35.8**
1000	remaining errors:1,139			remaining errors:1,043		

Table 7: Error detection results on the GermEval 2014 NER testset after N iterations (true positives, ED precision and recall).

errors) than in the VI-AL setting (1,628 errors), as in the first setting we used a majority vote for generating the data pool while in the second setting we relied on the predictions of MACE. For POS tagging, we did not observe a difference between the initial data pools (Table 3). For NER, however, the initial predictions of MACE are better than the majority vote.

During the first 800 iterations, precision for VI-AL is much higher than for QBC, but then slowly decreases. For QBC, however, we see the opposite trend. Here precision stays in the range of 52-56% for the first 600 iterations. After that, it slowly increases, and during the last iterations QBC precision outperforms VI-AL.

Recall, however, is higher for the VI-AL model, for *all* iterations. This means that even if precision is slightly lower than in the QBC setting after 800 iterations, it is still better to use the VI-AL model. For comparison, in the QBC setting we still have 1,139 errors left in the corpus after 1,000 iterations, while for VI-AL the number of errors remaining in the data is much lower (1,043).

5.4 Experiment 4: A real-world scenario

In our final experiment, we test our approach in a real-world scenario with a human annotator in the loop. To that end, we let two linguistically trained human annotators correct POS errors identified by AL. We use the out-of-domain data from experiment 2 (§5.2), specifically the *answers* and *weblog* subcorpora.

We run two VI-AL experiments where the oracle is presented with new error candidates for 500 iterations. The time needed for correction was 135 minutes (annotator 1, answers) and 157 minutes (annotator 2, weblog) for correcting 500 instances each. This includes the time needed to consult the annotation guidelines, as both annotators had no prior experience with the extended English Web treebank guidelines. We expect that the amount of time needed for correction will decrease when the annotators become more familiar with the annotation scheme. Results are shown in Table 8.

	VI-AL *with human annotator*					
	answers			*weblog*		
N	**# tp**	**ED prec**	**rec**	**# tp**	**ED prec**	**rec**
100	71	68.0	10.8	62	62.0	20.3
200	103	63.5	20.2	112	56.0	36.7
300	177	58.0	27.6	156	52.0	51.1
400	224	55.3	35.1	170	42.5	55.7
500	259	51.2	40.6	180	36.0	59.0

Table 8: POS results for VI-AL with a **human annotator** on 2 web genres (true positives, precision and recall for error detection on 5,000 tokens)

As expected, precision as well as recall are lower for the human annotators as compared to the simulation study (Table 6). However, even with some annotation noise we were able to detect more than 40% of all errors in the *answers* data and close to 60% of all errors in the *weblog* corpus, by manually inspecting only 10% of the data. This results in an increase in POS label accuracy from 88.8 to 92.5% for the *answers* corpus and from 93.9 to 97.5% for the *weblogs*, which is very close to the 97.8% we obtained in the simulation study (Table 4).

6 Conclusions

In the paper, we addressed a severely understudied problem, namely the detection of errors in *automatically* annotated language resources. We present an approach that combines an unsupervised generative model with human supervision in an AL framework. Using POS tagging and NER as test cases, we showed that our model can detect errors with high precision *and* recall, and works especially well in an out-of-domain setting. Our approach is language-agnostic and can be used without retraining the classifiers, which saves time and is of great practical use in an AL setting. We also showed that combining an unsupervised generative model with human supervision is superior to using a query-by-committee strategy for AL.

Our system architecture is generic and can be applied to any classification task, and we expect it to be of use in many annotation projects, especially when dealing with non-standard data or in out-of-domain settings.

Acknowledgments

This research has been conducted within the Leibniz Science Campus "Empirical Linguistics and Computational Modeling", funded by the Leibniz Association under grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art (MWK) of the state of Baden-Württemberg.

References

Bharat Ram Ambati, Mridul Gupta, Rahul Agarwal, Samar Husain, and Dipti Misra Sharma. 2011. Error detection for treebank validation. In *Proceedings of the 9th Workshop on Asian Language Resources*. Chiang Mai, Thailand, ALR9, pages 23–30.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation* 43(3):209–226.

Eyal Beigman and Beata Beigman Klebanov. 2009. Learning with annotation noise. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore, ACL'09, pages 280–287.

Alan Joseph Bekker and Jacob Goldberger. 2016. Training deep neural-networks based on unreliable labels. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*. ICASSP.

Darina Benikova, Seid Muhie Yimam, Prabhakaran Santhanam, and Chris Biemann. 2015. GermaNER: Free open German Named Entity Recognition tool. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology (GSCL'15)*. Essen, Germany, pages 31–38.

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*. Madrid, Spain, WWW'09, pages 51–60.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank. Technical Report LDC2012T13, Philadelphia: Linguistic Data Consortium.

David Cohn, Zoubin Ghahramani, and Michael Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research* 4:129–145.

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

Marcus Dickinson and Detmar W. Meurers. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*. Budapest, Hungary, EACL'03, pages 107–114.

Eleazar Eskin. 2000. Automatic corpus correction with anomaly detection. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*. NAACL'00, pages 148–153.

Manaal Faruqui and Sebastian Padó. 2010. Training and evaluating a German Named Entity Recognizer with semantic generalization. In *Proceedings of the Conference on Natural Language Processing*. KONVENS'10, pages 129–133.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested Named Entity Recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP'09, pages 141–150.

Jesús Giménez and Lluís Màrquez. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. Lisbon, Portugal, LREC, pages 43–46.

Péter Halácsy, András Kornai, and Csaba Oravecz. 2007. HunPos: An open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Prague, Czech Republic, ACL'07, pages 209–212.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia, USA, NAACL-HLT'13, pages 1120–1130.

Mark Johnson. 2007. Why doesn't EM find good HMM pos-taggers? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Prague, Czech Republic, EMNLP'07, pages 296–305.

Pavel Kveton and Karel Oliva. 2002. (Semi-)automatic detection of errors in pos-tagged corpora. In *Proceedings of the 19th International Conference on Computational Linguistics*. Taipei, Taiwan, COLING'02, pages 1–7.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden, ACL'10, pages 504–513.

Hrafn Loftsson. 2009. Correcting a POS-tagged corpus using three complementary methods. In *Proceedings of the 12th Conference of the European Chapter*

of the ACL. Athens, Greece, EACL'09, pages 523–531.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014. Tagging the web: Building a robust web tagger with neural network. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland, ACL'14, pages 144–154.

Christopher D. Manning. 2011. Part-of-speech tagging from 97linguistics? In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing*. Tokyo, Japan, CICLing'11, pages 171–189.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, ACL'16, pages 412–418.

Ines Rehbein. 2014. POS error detection in automatically annotated corpora. In *Proceedings of the 8th Linguistic Annotation Workshop*. LAW VIII, pages 20–28.

Dennis Reidsma and Jean Carletta. 2008. Reliability measurement without limits. *Computational Linguistics* 34(3):319–326.

Vitor Rocio, Joaquim Silva, and Gabriel Lopes. 2007. Detection of strange and wrong automatic part-of-speech tagging. In *Proceedings of the Aritficial Intelligence 13th Portuguese Conference on Progress in Artificial Intelligence*. Guimarães, Portugal, EPIA07, pages 683–690.

Helmut Schmid. 1999. Improvements in part-of-speech tagging with an application to German. In Susan Armstrong, Kenneth Church, Pierre Isabelle, Sandra Manzi, Evelyne Tzoukermann, and David Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*, Kluwer Academic Publishers, Dordrecht, volume 11 of *Text, Speech and Language Processing*, pages 13–26.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. Pittsburgh, Pennsylvania, USA, COLT'92, pages 287–294.

Victor Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD'08, pages 614–622.

Rion Snow, Brendan O'Connor, Daniel 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*. Honolulu, Hawaii, EMNLP'08, pages 254–263.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the SIGNLL Conference on Computational Natural Language Learning*. Edmonton, Canada, CoNLL'03, pages 142–147.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Edmonton, Canada, NAACL'03, pages 173–180.

Hans van Halteren. 2000. The detection of inconsistency in manually tagged text. In *Proceedings of the COLING-2000 Workshop on Linguistically Interpreted Corpora*. Centre Universitaire, Luxembourg, pages 48–55.

Liyue Zhao, Gita Sukthankar, and Rahul Sukthankar. 2011. Incremental relabeling for active learning with noisy crowdsourced annotations. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing*. PASSAT and SocialCom, pages 728–733.