Annotation Error Detection: Analyzing the Past and Present for a More Coherent Future

Jan-Christoph Klie*
Ubiquitous Knowledge Processing Lab
Department of Computer Science
Technical University of Darmstadt
www.ukp.tu-darmstadt.de

Bonnie Webber
School of Informatics,
University of Edinburgh

Iryna Gurevych
UKP Lab / TU Darmstadt

Annotated data is an essential ingredient in natural language processing for training and evaluating machine learning models. It is therefore very desirable for the annotations to be of high quality. Recent work, however, has shown that several popular datasets contain a surprising number of annotation errors or inconsistencies. To alleviate this issue, many methods for annotation error detection have been devised over the years. While researchers show that their approaches work well on their newly introduced datasets, they rarely compare their methods to previous work or on the same datasets. This raises strong concerns on methods' general performance and makes it difficult to assess their strengths and weaknesses. We therefore reimplement 18 methods for detecting potential annotation errors and evaluate them on 9 English datasets for text classification as well as token and span labeling. In addition, we define a uniform evaluation setup including a new formalization of the annotation error detection task, evaluation protocol, and general best practices. To facilitate future research and reproducibility, we release our datasets and implementations in an easy-to-use and open source software package.[1]

1. Introduction

Annotated corpora are an essential component in many scientific disciplines, including natural language processing (NLP) (Gururangan et al. 2020; Peters, Ruder, and Smith

* Corresponding author.

[1] `https://github.com/UKPLab/nessie`.

Action Editor: Nianwen Xue. Submission received: 7 June 2022; revised version received: 29 August 2022; accepted for publication: 23 September 2022.

https://doi.org/10.1162/coli_a_00464

2019), linguistics (Haselbach et al. 2012), language acquisition research (Behrens 2008), and the digital humanities (Schreibman, Siemens, and Unsworth 2004). Corpora are used to train and evaluate machine learning models, to deduce new knowledge, and to suggest appropriate revisions to existing theories. Especially in machine learning, high-quality datasets play a crucial role in advancing the field (Sun et al. 2017). It is often taken for granted that gold standard corpora do not contain errors—but alas, this is not always the case. Datasets are usually annotated by humans who can and do make mistakes (Northcutt, Athalye, and Mueller 2021). Annotation errors can even be found in corpora used for shared tasks such as CONLL-2003 (Tjong Kim Sang and De Meulder 2003). For instance, *Durban* is annotated there as PER (person) and *S.AFRICA* as MISC (miscellaneous), but both should be annotated as LOC (location).

Gold standard annotation is also subject to inconsistency, where words or phrases that are intended to refer to the same type of thing (and so should be labeled in the same way) are nevertheless assigned different labels (see, e.g., Hollenstein, Schneider, and Webber 2016). For example, in CONLL-2003, when *Fiorentina* was used to refer to the local football club, it was annotated as ORG, but when *Japan* was used to refer to the Japanese national football team, it was inconsistently annotated as LOC. One reason for annotation inconsistencies is that tokens can be ambiguous, either because they have multiple senses (e.g., the word *club* can refer to an organization or to a weapon), or because metonymy allows something to be referred to by one of its parts or attributes (e.g., the Scottish curling team being referred to as *Scotland*, as in *Scotland beat Canada in the final match*). We further define errors as well as inconsistencies and also discuss ambiguity in detail in § 3.1.

Such annotation errors or inconsistencies can negatively impact a model's performance or even lead to erroneous conclusions (Manning 2011; Northcutt, Athalye, and Mueller 2021; Larson et al. 2020; Zhang et al. 2021). A deployed model that learned errors during training can potentially cause harm, especially in critical applications like medical or legal settings. High-quality labels are needed to evaluate machine learning methods even if they themselves are robust to label noise (e.g., Song et al. 2020). Corpus linguistics relies on correctly annotated data to develop and confirm new theories. Learner corpora containing errors might be detrimental to the language learning experience and teach wrong lessons. Hence, it is imperative for datasets to have high-quality labels.

Cleaning the labels by hand, however, is expensive and time consuming. Therefore, many automatic methods for annotation error detection (AED) have been devised over the years. These methods enable dataset creators and machine learning practitioners to narrow down the instances that need manual inspection and—if necessary—correction. This reduces the overall work needed to find and fix annotation errors (see, e.g., Reiss et al. 2020). As an example, AED has been used to discover that widely used benchmark datasets contain errors and inconsistencies (Northcutt, Athalye, and Mueller 2021). Around 2% of the samples (sometimes even more than 5%) have been found incorrectly annotated in datasets like Penn Treebank (Dickinson and Meurers 2003a), sentiment analysis datasets like SST, Amazon Reviews, or IMDb (Barnes, Øvrelid, and Velldal 2019; Northcutt, Athalye, and Mueller 2021), CoNLL-2003 (Wang et al. 2019; Reiss et al. 2020), or relation extraction in TACRED (Alt, Gabryszak, and Hennig 2020; Stoica, Platanios, and Poczos 2021). AED has likewise been used to find ambiguous instances, for example, for part-of-speech (POS) annotation (Dickinson and Meurers 2003a). Additionally, it has been shown that errors in automatically annotated (silver) corpora can also be found and fixed with the help of AED (Rehbein 2014; Ménard and Mougeot 2019).

While AED methods have been applied successfully in the past (e.g., Reiss et al. 2020), there are several issues that hinder their widespread use. New approaches for AED are often only evaluated on newly introduced datasets that are proprietary or not otherwise available (e.g., Dligach and Palmer 2011; Amiri, Miller, and Savova 2018; Larson et al. 2019). Also, they rarely compare newly introduced methods to previous work or baselines. These issues make comparisons of AED methods very difficult. In addition to that, there is neither agreement on how to evaluate AED methods, nor which metrics to use during their development and application. As a result, it is often not clear how well AED works in practice, especially which AED methods should be applied to which kind of data and underlying tasks. To alleviate these issues, we define a unified evaluation setup for AED, conduct a large-scale analysis of 18 AED methods, and apply them to 9 datasets for text classification, token labeling, and span labeling. This work focuses on errors and inconsistencies related to instance labels. We leave issues such as boundary errors, sentence splitting, or tokenization for future work. The methods presented in this article are particularly suited to the NLP community, but many of them can also be adapted to other tasks (e.g., relation classification) and domains (like computer vision). The research questions we answer are:

RQ1 Which methods work well across tasks and datasets?

RQ2 Does model calibration help to improve AED performance?

RQ3 To what extent are model and AED performance correlated?

RQ4 What (performance) impact does using cross-validation have?

The research reported in this article addresses the aforementioned issues by providing the following contributions:

Evaluation Methodology To unify its findings and establish comparability, we first define the task of AED and a standardized evaluation setup, including an improvement for evaluating span labeling in this context (§ 3.1).

Easy to Use Reference Implementations We survey past work from the last 25 years and implement the 18 most common and generally applicable AED methods (§ 3.2). We publish our implementation in a Python package called NESSIE that is easy to use, thoroughly tested, and extensible to new methods and tasks. We provide abstractions for models, tasks, as well as helpers for cross validation to reduce the boilerplate code needed to a minimum. In addition, we provide extensive documentation and code examples. Our package makes it therefore significantly easier to get started with AED for researchers and practitioners alike.

Benchmarking Datasets We identify, vet, and generate datasets for benchmarking AED approaches, which results in 9 datasets for text classification, token labeling, and span labeling (§ 4). We also publish the collected datasets to facilitate easy comparison and reproducibility.

Evaluation and Analysis Using our implementation, we investigate several fundamental research questions regarding AED (§ 5). We specifically focus on how to achieve the best AED performance for each task and dataset, taking model calibration, usage of cross-validation, as well as model selection into account. Based on our results, we provide recipes and give recommendations on how to best use AED in practice (§ 6).

2. Related Work

This section provides a brief overview of annotation error detection and its related tasks.

Annotation Error Detection. In most works, AED is used as a means to improve the quality of an annotated corpus. As such, the method used is treated as secondary and possible methods are not compared. The work of Amiri, Miller, and Savova (2018) and Larson et al. (2020) are the few instances that implement different methods and baselines, but only use newly introduced datasets. In other cases, AED is just discussed as a minor contribution and not thoroughly evaluated (e.g., Swayamdipta et al. 2020, Rodriguez et al. 2021).

Therefore, to the best of our knowledge, no large-scale evaluation of AED methods exists. Closest to the current study is the work of Dickinson (2015), a survey about the history of annotation error detection. However, that survey does not reimplement, compare, or evaluate existing methods quantitatively. Its focus is also limited to part-of-speech and dependency annotations. Our work fills the aforementioned gaps by reimplementing 18 methods for AED, evaluating the methods against 9 datasets, and investigating the setups in which they perform best.

Annotation Error Correction. After potential errors have been detected, the next step is to have them corrected to obtain gold labels. This is usually done by human annotators who carefully examine those instances that have been detected. Some AED methods can also both detect and correct labels. Only a few groups have studied correction so far (e.g., Květoň and Oliva 2002; Loftsson 2009; Dickinson 2006; Angle, Mishra, and Sharma 2018; Qian et al. 2021). In this study, we focus on detection and leave an in-depth treatment of annotation error correction for future work.

Error Type Classification. Even if errors are not corrected automatically, it may still be worth identifying the type of each error. For instance, Larson et al. (2020) investigate the different errors for slot filling (e.g., incorrect span boundaries, incorrect labels, or omissions). Alt, Gabryszak, and Hennig (2020) investigate error types for relation classification. Yaghoub-Zadeh-Fard et al. (2019) collect tools and methods to find quality errors in paraphrases used to train conversational agents. Barnes, Øvrelid, and Velldal (2019) analyze the types of errors found in annotating for sentiment analysis. While error type classification has not been explicitly addressed in the current study, such classification requires good AED, so the results of the current study can contribute to automate error type classification in the future.

Training with Label Noise. Related to AED is the task of training with noise: Given a dataset that potentially contains label errors, train a model so that the performance impact due to noisy labels is as low as possible (Song et al. 2020). The goal in this setting is not to clean a dataset (labels are left as is), but to obtain a well-performing machine learning model. An example application is learning directly from crowdsourced data without adjudicating it (Rodrigues and Pereira 2018). Training with label noise and AED have in common that they both enable models to be trained when only noisy data is available. Evaluating these models still requires clean data, which AED can help to produce.

3. Annotation Error Detection

In this section we introduce AED and formalize the concept, then categorize state-of-the-art approaches according to our formalization.

3.1 Task Definition

Given an adjudicated dataset with one label per annotated instance, the goal of AED is to find those instances that are likely labeled incorrectly or inconsistently. These candidate instances then can be given to human annotators for manual inspection or used in annotation error correction methods. The definition of *instance* depends on the task and defines the granularity on which errors or inconsistencies are detected. In this article, we consider AED in text classification (where instances are sentences), in token labeling (instances are tokens; e.g., POS tagging), and in span labeling (instances are spans; e.g., named entity recognition [NER]). AED can and has been applied to many domains and tasks, for instance, sentiment analysis (Barnes, Øvrelid, and Velldal 2019; Northcutt, Athalye, and Mueller 2021), relation extraction (Alt, Gabryszak, and Hennig 2020), POS tagging (Dickinson and Meurers 2003a; Loftsson 2009), image classification (Northcutt, Athalye, and Mueller 2021), NER (Wang et al. 2019; Reiss et al. 2020), slot filling (Larson et al. 2020), or speech classification (Northcutt, Athalye, and Mueller 2021).

We consider a label to be **incorrect** if there is a unique, true label that should be assigned but it differs from the label that has been assigned. For example, there is a named entity span Durban in CONLL-2003 which has been labeled PER, whereas in context, it refers to a city in South Africa, so the label should be LOC.

Instances can also be **ambiguous**, that is, there are at least two different labels that are valid given the context. For instance, in the sentence *They were visiting relatives*, *visiting* can either be a verb or an adjective. Ambiguous instances themselves are often more difficult for machine learning models to learn from and predict. Choosing one label over another is neither inherently correct nor incorrect. But ambiguous instances can be annotated *inconsistently*. We consider a label **inconsistent** if there is more than one potential label for an instance, but the choice of resolution was different for similar instances. For example, in the sentence *Stefan Edberg produced some of his vintage best on Tuesday to extend his grand run at the Grand Slams by toppling Wimbledon champion Richard Krajicek*, the entity *Wimbledon* was annotated as LOC. But in the headline of this article, *Edberg extends Grand Slam run, topples Wimbledon champ*, the entity *Wimbledon* was annotated as MISC. We discuss the impact of ambiguity on AED further in § 3.1. An instance that is neither incorrect nor inconsistent is **correct**. If not explicitly stated otherwise, then we refer to both incorrect and inconsistent as incorrect or erroneous.

AED is typically used after a new dataset has been annotated and adjudicated. It is assumed that no already cleaned data and no other data having the same annotation scheme is available.

Flaggers vs. Scorers. We divide automatic methods for AED into two categories, which we dub *flaggers* and *scorers*. Flagging means that methods cast a binary judgment whether the label for an instance is correct or incorrect. Scoring methods give an estimate on how likely it is that an annotation is incorrect. These correspond to classification and ranking.

While flaggers are explicit as to whether they consider an annotation to be incorrect, they do not indicate the likelihood of that decision. On the other hand, while scorers provide a likelihood, they require a threshold value to decide when an annotation is considered an error—for example, those instances with a score above 80%. Those would then be given to human evaluation. Scorers can also be used in settings similar to active learning for error correction (Vlachos 2006).

This distinction between flaggers and scorers regarding AED has not been made in previous work, as typically approaches of one type or the other have been proposed per paper. But it is key to understanding why different metrics need to be used when evaluating flaggers compared to scorers, similarly to unranked and ranked evaluation from information retrieval (see § 5).

Ambiguity. In certain NLP tasks, there exists more than one valid label per instance (Kehler et al. 2007; Plank, Hovy, and Søgaard 2014b; Aroyo and Welty 2015; Pavlick and Kwiatkowski 2019; Basile et al. 2021). While this might reduce the usefulness of AED at first glance, gold labels are not required by AED, as it is about uncovering problems independent of their cause and not assigning a gold label. Instances detected this way are then marked for further processing. They can be, for instance, inspected for whether they are incorrectly or inconsistently annotated. Ambiguous or difficult instances especially deserve additional scrutiny when creating a corpus; finding them is therefore very useful. Once found, several alternatives are possible: (1) Ambiguous cases can be corrected (e.g., Alt, Gabryszak, and Hennig 2020; Reiss et al. 2020); (2) they can be removed (e.g., Jamison and Gurevych 2015); (3) their annotation guidelines can be adjusted to reduce disagreement (e.g., Pustejovsky and Stubbs 2013); (4) the task can eventually be redefined to use soft labels (Fornaciari et al. 2021) or used to learn from disagreement (Paun et al. 2018). Finding such instances is hence very desirable and can be achieved by AED. But similarly to past work on AED, we focus on detecting errors and inconsistencies as a first step and leave evaluating ambiguity detection performance for future work.

3.2 Survey of Existing AED Methods

Over the past three decades, several methods have been developed for AED. Here, we group them by how they detect annotation errors and briefly describe each of them. In this article, we focus on AED for natural language processing, but (as noted earlier in § 1), many of the presented methods can be and have been adjusted to different tasks and modalities. An overview of the different methods is also given in Table 1.

3.2.1 Variation-based. Methods based on the variation principle leverage the observation that similar surface forms are often annotated with only one or at most a few distinct labels. If an instance is annotated with a different, rarer label, then it is possibly an annotation error or an inconsistency. Variation-based methods are relatively easy to implement and can be used in settings in which it is difficult to train a machine learning model, such as low-resource scenarios or tasks that are difficult to train models for, for example, detecting lexical semantic units (Hollenstein, Schneider, and Webber 2016). The main disadvantage of variation-based methods is that they need similar surface forms to perform well, which is not the case in settings like text classification or datasets with diverse instances.

Variation n-grams. The most frequently used method of this kind is variation *n*-grams, which has been initially developed for POS tagging (Dickinson and Meurers 2003a) and later extended to discontinuous constituents (Dickinson and Meurers 2005),

Table 1
Annotation error detection methods evaluated in this work. In most scorer methods, scorer output and erroneous labels are positively correlated. Scorers marked with * show negative correlation.

Abbr.	Method Name	Tasks			Proposed by
		Text	Token	Span	
Flagger methods					
CL	Confident Learning	✓	✓	✓	Northcutt et al. (2021)
CS	Curriculum Spotter	✓	·	·	Amiri et al. (2018)
DE	Diverse Ensemble	✓	✓	✓	Loftsson (2009)
IRT	Item Response Theory	✓	✓	✓	Rodriguez et al. (2021)
LA	Label Aggregation	✓	✓	✓	Amiri et al. (2018)
LS	Leitner Spotter	✓	·	·	Amiri et al. (2018)
PE	Projection Ensemble	✓	✓	✓	Reiss et al. (2020)
RE	Retag	✓	✓	✓	van Halteren (2000)
VN	Variation N-Grams	·	✓	✓	Dickinson and Meurers (2003a)
Scorer methods					
BC	Borda Count	✓	✓	✓	Larson et al. (2020)
CU	Classification Uncertainty	✓	✓	✓	Hendrycks and Gimpel (2017)
DM*	Data Map Confidence	✓	·	·	Swayamdipta et al. (2020)
DU	Dropout Uncertainty	✓	✓	✓	Amiri et al. (2018)
KNN	k-Nearest Neighbor Entropy	✓	✓	✓	Grivas et al. (2020)
LE	Label Entropy	·	✓	✓	Hollenstein et al. (2016)
MD	Mean Distance	✓	✓	✓	Larson et al. (2019)
PM*	Prediction Margin	✓	✓	✓	Dligach and Palmer (2011)
WD	Weighted Discrepancy	·	✓	✓	Hollenstein et al. (2016)

predicate-argument structures (Dickinson and Lee 2008), dependency parsing (Boyd, Dickinson, and Meurers 2008), or slot filling (Larson et al. 2020). For each instance, n-gram contexts of different sizes are collected and compared to each other. It is considered incorrect if the label for an instance disagrees with labels from other instances with the same n-gram context.

Label Entropy and Weighted Discrepancy. Hollenstein, Schneider, and Webber (2016) derive metrics from the surface form and label counts that are then used as scorers. These are the entropy over the label count distribution per surface form or the weighted difference between most and least frequent labels. They apply their methods to find possible annotation errors in datasets for multi-word expressions and super-sense tagging, which are then reviewed manually for tokens that are actual errors.

3.2.2 Model-based. Probabilistic classifiers trained on the to-be-corrected dataset can be used to find annotation errors. Models in this context are usually trained via cross-validation (CV) and the respective holdout set is used to detect errors. After all folds have been used as holdout, the complete dataset is analyzed. Because some methods described below directly use model probabilities, it is of interest whether these are accurately describing the belief of the model. This is not always true, as models often are overconfident (Guo et al. 2017). Therefore, we will evaluate whether **calibration**, that is, tuning probabilities so that they are closer to the observed accuracy, can improve performance (see § 5.2). Several ways have been devised for model-based AED, which are described below. Note that most model-based methods are agnostic to the task itself and rely only on model predictions and confidences. This is why they can easily be used with different tasks and modalities.

Re-tagging. A simple way to use a trained model for AED is to use model predictions directly; when the predicted labels are different from the manually assigned ones, instances are flagged as annotation errors (van Halteren 2000). Larson et al. (2020) apply this using a conditional random field (CRF) tagger to find errors in crowdsourced slot-filling annotations. Similarly, Amiri, Miller, and Savova (2018) use *Retag* for text classification. Yaghoub-Zadeh-Fard et al. (2019) train machine learning models to classify whether paraphrases contain errors and if they do, what kind of error it is. To reduce the need of annotating instances twice for higher quality, Dligach and Palmer (2011) train a model on the labels given by an initial annotator. If the model disagrees with the instance's labeling, then it is flagged for re-annotation. For cleaning dependency annotations in a Hindi treebank, Ambati et al. (2011) train a logistic regression classifier. If the model's label does not agree with the original annotation and the model confidence is above a predefined threshold, then the annotation is considered to be incorrect. *CrossWeigh* (Wang et al. 2019) is similar to *Retag* with repeated CV. During CV, *entity disjoint filtering* is used to force more model errors: Instances are flagged as erroneous if the probability of their having the correct label falls below the respective threshold. As it is computationally much more expensive than *Retag* while being very similar, we did not include it in our comparison.

Classification Uncertainty. Probabilistic classification models assign probabilities that are typically higher for instances that are correctly labeled compared with erroneous ones (Hendrycks and Gimpel 2017). Therefore, the class probabilities of the noisy labels can be used to score these for being an annotation error. Using model uncertainty is basically identical to using the network loss (as, e.g., used by Amiri, Miller, and Savova 2018) because the cross-entropy function used to compute the loss is monotonic. The probability formulation, however, allows us to use calibration more easily later (see § 5.2), which is why we adapt the former instead of using the loss.

Prediction Margin. Inspired by active learning, *Predictive Margin* uses the probabilities of the two highest scoring labels for an instance. The resulting score is simply their difference (Dligach and Palmer 2011). The intuition behind this is that samples with a smaller margin are more likely to be an annotation error, since the smaller the decision margin is the more unsure the model was.

Confident Learning. This method estimates the joint distribution of noisy and true labels (Northcutt, Jiang, and Chuang 2021). A threshold is then learned (the average self-confidence) and instances whose computed probability of having the correct label is below the respective threshold are flagged as erroneous.

Dropout Uncertainty. Amiri, Miller, and Savova (2018) use Monte Carlo dropout (Gal and Ghahramani 2016) to estimate the uncertainty of an underlying model. There are different acquisition methods to compute uncertainty from the stochastic passes. A summary can be found in Shelmanov et al. (2021). The work of Amiri, Miller, and Savova (2018) uses the probability variance averaged over classes.

Label Aggregation. Given T predictions obtained via Monte Carlo dropout, Amiri, Miller, and Savova (2018) use MACE (Hovy et al. 2013), an aggregation technique from crowdsourcing to adjudicate the resulting repeated predictions.

3.2.3 Training Dynamics. Methods based on training dynamics use information derived from how a model behaves during training and how predictions change over the course of its training.

Curriculum and Leitner Spotter. Amiri, Miller, and Savova (2018) train a model via curriculum learning, where the network trains on easier instances during earlier epochs and is then gradually introduced to harder instances. Instances then are ranked by how hard they were perceived during training. They also adapt the ideas of the Zettelkasten (Ahrens 2017) and Leitner queue networks (Leitner 1974) to model training. There, difficult instances are presented more often during training than easier ones. The assumption behind both of these methods is that instances that are perceived harder or misclassified more frequently are more often annotation errors than are easier ones. These two methods require that the instances can be scheduled independently. This is, for instance, not the case for sequence labeling, as the model trains on complete sentences and not individual tokens or spans. Even if they have different difficulties, they would end up in the same batch nonetheless.

Data Map Confidence. Swayamdipta et al. (2020) use the class probability for each instance's gold label across epochs as a measure of confidence. In their experiments, low confidence correlates well with an item having an incorrect label.

3.2.4 Vector Space Proximity. Approaches of this kind leverage dense embeddings of tokens, spans, and texts into a vector space and use their distribution therein. The distance of an instance to semantically similar instances is expected to be smaller than the distance to semantically different ones. Embeddings are typically obtained by using BERT-type models for tokens and spans (Devlin et al. 2019) or S-BERT for sentences (Reimers and Gurevych 2019).

Mean Distance. Larson et al. (2019) compute the centroid of each class by averaging vector embeddings of the respective instances. Items are then scored by the distance between their embedding vector to their centroid. The underlying assumption is that semantically similar items should have the same label and be close together (and thereby close to the centroid) in the vector space. In the original publication, this method was only evaluated on detecting errors in sentence classification datasets, but we extend it to also token and span classification.

k-Nearest-Neighbor Entropy. In the context of NER in clinical reports, Grivas et al. (2020) leverage the work of Khandelwal et al. (2020) regarding nearest-neighbor language models to find mislabeled named entities. First, all instances are embedded into a vector space. Then, the k nearest neighbors of each instance according to their Euclidean distance are retrieved. Their distances to the instance embedding vector are then used to compute a distribution over labels by applying softmax. An instance's score is then the entropy of its distance distribution; if it is large, it indicates uncertainty, hinting at being mislabeled. Grivas et al. (2020) only used this method qualitatively; we have turned their qualitative approach into a method that can be used to score instances automatically and evaluated it on detecting errors in both NER and sentence classification—the latter using S-BERT embeddings. This method was only evaluated on detecting errors in NER datasets, but we apply it to sentence classification as well by using S-BERT embeddings.

3.2.5 Ensembling. Ensemble methods combine the scores or predictions of several individual flaggers or scorers to obtain better performance than the sum of their parts.

Diverse Ensemble. Instead of using a single prediction like *Retag* does, the predictions of several, architecturally different models are aggregated. If most of them disagree on the label for an instance, then it is likely to be an annotation error. Alt, Gabryszak, and Hennig (2020) use an ensemble of 49 different models to find annotation errors in the TACRED relation extraction corpus. In their setup, instances are ranked by how often a model suggests a label different from the original one. Barnes, Øvrelid, and Velldal (2019) use three models to analyze error types on several sentiment analysis datasets; they flag instances for which all models disagree with the gold label. Loftsson (2009) and Angle, Mishra, and Sharma (2018) use an ensemble of different taggers to correct POS tags.

Projection Ensemble. In order to correct the CONLL-2003 named entity corpus, Reiss et al. (2020) train 17 logistic regression models on different Gaussian projections of BERT embeddings. The aggregated predictions that disagree with the dataset were then corrected by hand.

Item Response Theory. Lord, Novick, and Birnbaum (1968) developed *Item Response Theory* as a mathematical framework to model relationships between measured responses of test subjects (e.g., answers to questions in an exam) for an underlying, latent trait (e.g., the overall grasp on the subject that is tested). It can also be used to estimate the discriminative power of an item, namely, how well the response to a question can be used to distinguish between subjects of different ability. In the context of AED, test subjects are trained models, the observations are the predictions on the dataset, and the latent trait is task performance. Rodriguez et al. (2021) have shown that items that negatively discriminate (i.e., where a better response indicates being less skilled) correlate with annotation errors.

Borda Count. Similarly to combining several flaggers into an ensemble, rankings obtained from different scorers can be combined as well. For that, Dwork et al. (2001) propose leveraging Borda counts, a voting scheme that assigns points based on their ranking. For each scorer, given scores for N instances, the instance that is ranked the highest is given N points, the second-highest $N - 1$, and so on (Szpiro 2010). The points assigned by different scorers are then summed up for each instance and form the aggregated ranking. Larson et al. (2019) use this to combine scores for runs of *Mean Distance* with different embeddings and show that this improves overall performance compared to only using individual scores.

3.2.6 Rule-based. Several studies leverage rules that describe which annotations are valid and which are not. For example, to find errors in POS annotated corpora, Květoň and Oliva (2002) developed a set of conditions that tags have to fulfill in order to be valid, especially n-grams that are impossible based on the underlying lexical or morphological information of their respective surface forms. Rule-based approaches for AED can be very effective but are hand-tailored to the respective dataset, its domain, language, and task. Our focus in this article is to evaluate generally applicable methods that can be used for many different tasks and settings. Therefore, we do not discuss rule-based methods further in the current work.

4. Datasets and Tasks

In order to compare the performance of AED methods on a large scale, we need datasets with parallel gold and noisy labels. But even with previous work on correcting noisy corpora, such datasets are hard to find.

We consider three kinds of approaches to obtain datasets that can be used for evaluating AED. First, existing datasets can be used whose labels are then randomly perturbed. Second, there exist adjudicated gold corpora for which the annotations of single annotators exist. Noisy labels are then the unadjucated annotations. These kinds of corpora are mainly obtained from crowdsourcing experiments. Third, there are manually corrected corpora whose both clean and noisy parts have been made public. Because only a few such datasets are available for AED, we have derived several datasets of the first two types from existing corpora.

When injecting random noise we use flipped label noise (Zheng, Awadallah, and Dumais 2021) with a noise level of 5%, which is in a similar range to error rates in previously examined datasets like PENN TREEBANK (Dickinson and Meurers 2003b) or CONLL-2003 (Reiss et al. 2020). In our settings, for a random subset of 5% instances, this kind of noise assigns uniformly a different label from the tagset without taking the original label into account. While randomly injecting noise is simple and can be applied to any existing gold corpus, errors in these datasets are often easy to spot (Larson et al. 2019). This is because errors typically made by human annotators vary with the actual label, which is not true for random noise (Hedderich, Zhu, and Klakow 2021). Note that evaluating AED methods does not require knowing true labels: All that is required are potentially noisy labels and whether or not they are erroneous. It is only correction that needs true labels as well as noisy ones.

As noted earlier, we will address AED in three broad NLP tasks: text classification, token labeling, and span labeling. These have been the tasks most frequently evaluated in AED and on which the majority of methods can be applied. Also, these tasks have many different machine learning models available to solve them. This is crucial for evaluating calibration (§ 5.2) and assessing whether well-performing models lead to better task performance for model-based methods (§ 5.3). To foster reproducibility and to obtain representative results, we then choose datasets that fulfill the following requirements: (1) they are available openly and free of charge, (2) they are for common and different NLP tasks, (3) they come from different domains, and (4) they have high inter-annotator agreement and very few annotation errors. Based on these criteria, we select 9 datasets. They are listed in Table 2 and are described in the following section. We manually inspected and carefully analyzed the corpora to verify that the given gold labels are of very high quality.

4.1 Text Classification

The goal of text classification is to assign a predefined category to a given text sequence (here, a sentence, paragraph, or a document). Example applications are news categorization, sentiment analysis, or intent detection. For text classification, each individual sentence or document is considered its own instance.

Table 2
Dataset statistics. We report the number of instances $|\mathcal{I}|$ and annotations $|A|$ as well as the number of mislabeled ones ($|\mathcal{I}_\epsilon|$ and $|\mathcal{A}_\epsilon|$), their percentage, as well as the number of classes $|\mathcal{C}|$. For token and span labeling datasets, $|\mathcal{A}|$ counts the number of annotated tokens and spans, respectively. *Kind* indicates whether the noisy part was created by randomly corrupting labels (R), or by aggregation (A) from individual annotations like crowdsourcing, or whether the gold labels stem from manual correction (M). Errors for span labeling are calculated via exact span match. *Source* points to the work that introduced the dataset for use in AED if it was created via manual correction and to the work proposing the initial dataset for aggregation or randomly perturbed ones.

Name	$\|\mathcal{I}\|$	$\|\mathcal{I}_\epsilon\|$	$\frac{\|\mathcal{I}_\epsilon\|}{\|\mathcal{I}\|}\%$	$\|\mathcal{A}\|$	$\|\mathcal{A}_\epsilon\|$	$\frac{\|\mathcal{A}_\epsilon\|}{\|\mathcal{A}\|}\%$	$\|\mathcal{C}\|$	Kind	Source
Text classification									
ATIS	4,978	238	4.78	4,978	238	4.78	22	R	Hemphill et al. (1990)
IMDb	24,799	499	2.01	24,799	499	2.01	2	M	Northcutt et al. (2021)
SST	8,544	420	4.92	8,544	420	4.92	2	R	Socher et al. (2013)
Token labeling									
GUM	7,397	3,920	52.99	137,605	6,835	4.97	18	R	Zeldes (2017)
Plank	500	373	74.60	7,876	931	11.82	13	A	Plank et al. (2014a)
Span labeling									
CoNLL-2003	3,380	217	6.42	5,505	262	4.76	5	M	Reiss et al. (2020)
SI Companies	500	224	44.80	1,365	325	23.81	11	M	Larson et al. (2020)
SI Flights	500	43	8.60	1,196	49	4.10	7	M	Larson et al. (2020)
SI Forex	520	63	12.12	1,263	98	7.76	4	M	Larson et al. (2020)

ATIS contains transcripts of user interactions with travel inquiry systems, annotated with intents and slots. For AED on intent classification, we have randomly perturbed the labels.

IMDb contains movie reviews labeled with sentiment. Northcutt, Athalye, and Mueller (2021) discovered that it contains a non-negligible amount of annotation errors. They applied *Confident Learning* to the test set and let crowdworkers check whether the flags were genuine.

SST The STANFORD SENTIMENT TREEBANK is a dataset for sentiment analysis of movie reviews from Rotten Tomatoes. We use it for binary sentiment classification and randomly perturb the labels.

4.2 Token Labeling

The task of token labeling is to assign a label to each token. The most common task in this category is POS tagging. As there are not many other tasks with easily obtainable datasets, we only use two different POS tagging datasets. For token labeling, each individual token is considered an instance.

GUM The GEORGETOWN UNIVERSITY MULTILAYER CORPUS is an open source corpus annotated with several layers from the Universal Dependencies project (Nivre et al. 2020). It has been collected by linguistics students at Georgetown University as part of their course work. Here, the original labels have been perturbed with random noise.

Plank POS contains Twitter posts that were annotated by Gimpel et al. (2011). Plank, Hovy, and Søgaard (2014a) mapped their labels to Universal POS tags and had 500 tweets reannotated by two new annotators. We flag an instance as erroneous if its two annotations disagree.

4.3 Span Labeling

Span labeling assigns labels not to single tokens, but to spans of text. Common tasks that can be modeled that way are NER, slot filling, or chunking. In this work, we assume that spans have already been identified, focusing only on finding label errors and leaving detecting boundary errors and related issues for future work. We use the following datasets:

CoNLL-2003 is a widely used dataset for NER (Tjong Kim Sang and De Meulder 2003). It consists of news wire articles from the Reuters Corpus annotated by experts. Reiss et al. (2020) discovered several annotation errors in the English portion of the dataset. They developed *Projection Ensembles* and then manually corrected the instances flagged by it. While errors concerning tokenization and sentence splitting were also corrected, we ignore them here as being out of scope of the current study. Therefore, we report slightly fewer instances and errors overall in Table 2. Wang et al. (2019) also corrected errors in CONLL-2003 and named the resulting corpus CONLL++. As they only re-annotated the test set and found fewer errors, we use the corrected version of Reiss et al. (2020).

Slot Inconsistencies is a dataset that was created by Larson et al. (2020) to investigate and classify errors in slot filling annotations. It contains documents of three domains (COMPANIES, FOREX, FLIGHTS) that were annotated via crowdsourcing. Errors were then manually corrected by experts.

Span labeling is typically indicated using Begin-Inside-Out (BIO) tags.[2] When labeling a span as `X`, tokens outside the span are labeled `O`, the token at the beginning of the span is labeled `B-X`, and tokens within the span are labeled `I-X`. Datasets for span labeling are also usually represented in this format.

This raises the issues of (1) boundary differences and (2) split entities. First, for model-based methods, models might predict different spans and span boundaries from the original annotations. In many evaluation datasets, boundary issues were also corrected and therefore boundaries for the same span in the clean and noisy data can be different, which makes evaluation difficult. Second, for scorers it does not make much sense to order BIO tagged tokens independently of their neighbors or to alter only parts of a sequence to a different label. This can lead to corrections that split entities, which is often undesirable. Therefore, directly using BIO tags as the granularity of detection and correction for span labeling is problematic.

Hence, we suggest converting the BIO tagged sequences back to a span representation consisting of begin, end, and label. This first step solves the issue of entities potentially being torn apart by detection and correction. Spans from the original data and from the model predictions then need to be aligned for evaluation in order to reduce boundary issues. This is depicted in Figure 1.

2 For simplicity, we describe the BIO tagging format. There are more advanced schemas like BIOES, but our resulting task-specific evaluation is independent of the actual schema used.

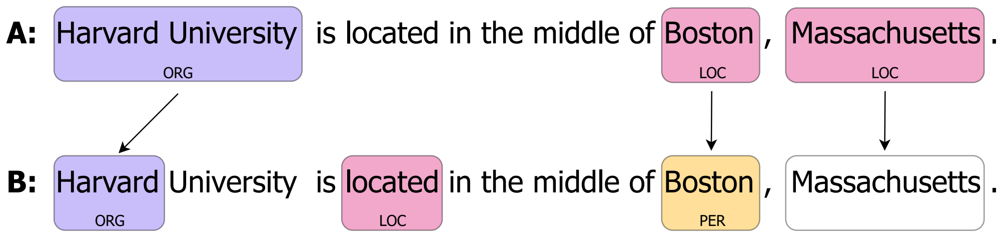

A: Harvard University is located in the middle of Boston , Massachusetts .
ORG
LOC
LOC
B: Harvard University is located in the middle of Boston , Massachusetts .
ORG
LOC
PER

Figure 1
Alignment between original or corrected spans *A* and noisy or predicted spans *B*. The goal is to find an alignment that maximizes overlap. Spans that are in *A* but find no match in *B* are given a match with the same offsets but a special, unique label that is different from all other labels (e.g., *Massachusetts*). Spans that are in *B* but find no match in *A* are dropped (e.g., *located*). Spans from *A* that have no overlapping span in *B* are considered different and cannot be aligned (e.g., *Boston* in *A* and *Massachusetts* in *B*). Span colors here indicate their labels.

We require a good alignment to (1) maximize overlap between aligned spans so that the most likely spans are aligned, (2) be deterministic, (3) not use additional information like probabilities, and (4) not align spans that have no overlap to avoid aligning things that should not be aligned. If these properties are not given, then the alignment and resulting confidences or representations that are computed based on this can be subpar. This kind of alignment is related to evaluation, for example, for NER in the style of MUC-5 (Chinchor and Sundheim 1993), especially for partial matching. Their alignment does not, however, satisfy (1) and (3) in the case of multiple predictions overlapping with a single gold entity. For instance, if the gold entity is *New York City* and the system predicted *York* and *New York*, then in most implementations, the first prediction is chosen and other predictions that also could match are discarded. What prediction is first depends on the order of predictions which is non-deterministic. This also does not choose the optimal alignment with maximum span overlap, which requires a more involved approach.

We thus adopt the following alignment procedure: Given a sequence of tokens, a set of original spans *A* and predicted/noisy spans *B*, align both sets of spans and thereby allow certain leeway of boundaries. The goal is to find an assignment that maximizes overlap of spans in *A* and *B*; only spans of *A* that overlap in at least one token with spans in *B* are considered. This can be formulated as a linear sum assignment problem: Given two sets A, B of equal size and a function that assigns a cost to connect an element of *A* with an element of *B*, find the assignment that minimizes the overall cost (Burkard, Dell'Amico, and Martello 2012). It can happen that not all elements of *A* are assigned a match in *B* and vice versa—we assign a special label that indicates missing alignment in the first case and drop spans of *B* that have no overlap in *A*. For the latter, it is possible to also assign a special label to indicate that a gold entity is missing; in this work, we focus on correcting labels only and hence leave using this information to detect missing spans for future work.

We are not aware of previous work that proposes a certain methodology for this. While Larson et al. (2020) evaluate AED on slot filling, it is not clear on which granularity they measure detection performance or whether and how they align. To the best of our knowledge, we are the first to propose this span alignment approach for span-level AED. Span alignment requires aggregating token probabilities into span probabilities, which is described in § 1.1. This alignment approach can also be extended to other tasks like object classification or matching boxes for optical character recognition. In that case, the metric to optimize is the Jaccard index.

5. Experiments

In this section we first define the general evaluation setup, metrics to be used, and the models that are leveraged for model-based AED. Details on how each method was implemented for this work can be found in Appendix A. In § 5.1 through § 5.4, we then describe our results for the experiments we perform to answer the research questions raised in § 1.

Metrics. As described in § 3.1, we differentiate between two kinds of annotation error detectors, *flaggers* and *scorers*. These need different metrics during evaluation, similar to unranked and ranked evaluation from information retrieval (Manning, Raghavan, and Schütze 2008). Flagging is a binary classification task. Therefore, we use the standard metrics for this task, which are precision, recall, and F1. We also record the percentage of instances flagged (Larson et al. 2020). Scoring produces a ranking, as in information retrieval. We use average precision[3] (AP), Precision@10%, and Recall@10%, similarly to Amiri, Miller, and Savova (2018) and Larson et al. (2019). There are reasons why both precision and recall can be considered the more important metric of the two. A low precision leads to increased cost because many more instances than necessary need to be inspected manually after detection. Similarly, a low recall leads to problems because there still can be errors left after the application of AED. As both arguments have merit, we will mainly use the aggregated metrics F1 and AP. Precision and recall at 10% evaluate a scenario in which a scorer was applied and the first 10% with the highest score—most likely to be incorrectly annotated—are manually corrected. We use the PYTREC-EVAL toolkit to compute these ranking metrics.[4] Recall relies on knowing the exact number of correctly and incorrectly annotated instances. While this information may be available when developing and evaluating AED methods, it is generally not available when actually applying AED to clean real data. One solution to computing recall then is to have experts carefully annotate a subset of the data and then use it to estimate recall overall.

In contrast to previous work, we explicitly do not use ROC AUC and discourage its use for AED, as it heavily overestimates performance when applied to imbalanced datasets (Davis and Goadrich 2006; Saito and Rehmsmeier 2015). Datasets needing AED are typically very imbalanced because there are far more correct labels than incorrect ones.

Models. We use multiple different neural and non-neural model types per task for model-based AED. These are used to investigate the relationship between model and method performances, whether model calibration can improve method performances and for creating diverse ensembles.

For text classification we use seven different models: logistic regression as well as gradient boosting machines (Ke et al. 2017) with either bag-of-word or S-BERT features (Reimers and Gurevych 2019), transformer based on DistilRoBERTa (Sanh et al. 2019),

3 Also known as Area Under the Precision-Recall Curve (AUPR/AUPRC). In AED, AP is also identical to mean average precision (mAP) used in other works.

4 `https://github.com/cvangysel/pytrec_eval`.

BiLSTM based on Flair (Akbik et al. 2019), and FastText (Joulin et al. 2017). For S-BERT, we use `all-mpnet-base-v2` as the underlying model, as it has been shown by their creators to produce sentence embeddings of the highest quality overall.

For token and span labeling, we use four different models: CRFs with the handcrafted features as proposed by Gimpel et al. (2011), BiLSTM + CRF based on Flair (Akbik et al. 2019), transformers with CRF based on DistilRoBERTa (Sanh et al. 2019), and logistic regression (also called maximum entropy model). For the initialization of Flair-based models we use a combination of GloVe (Pennington, Socher, and Manning 2014) as well as Byte-Pair Encoding embeddings (Heinzerling and Strube 2018) and a hidden layer size of 256 for both text classification and sequence labeling. Note that we do not perform extensive hyperparameter tuning for model selection because when using AED in practice, no annotated in-domain data can be held out for tuning since all data must be checked for errors. Also, when comparing models as we do here, it would be prohibitively expensive to carry out hyperparameter tuning across all datasets and model combinations. Instead, we use default configurations that have been shown to work well on a wide range of tasks and datasets.

When using transformers for sequence labeling we use the probabilities of the first subword token. We use 10-fold cross-validation to train each model and use the same model weights for all methods evaluated on the same fold. Thereby, all methods applied to the same fold use the predictions of the same model.

5.1 RQ1 – Which Methods Work Well across Tasks and Datasets?

We first report the scores resulting from the best setup as a reference to the upcoming experiments. Then we describe the experiments and results that lead to this setup. We do not apply calibration to any of the methods for the reported scores because it only marginally improved performance (see § 5.2). For model-based methods, the best performance for text classification and span labeling was achieved using transformers; for token labeling, best performance was achieved using Flair (see § 5.3). Not using cross-validation for model-based AED was found to substantially reduce recall for model-based AED (see § 5.4), so we have used 10-fold cross-validation in comparing model-based methods.

In Table 3, we present the overall performance in F1 and AP across all datasets and tasks. Detailed results including scores for all metrics can be found in Appendix C.

First of all, it can be seen that in datasets with randomly injected noise (ATIS, SST, and GUM), errors are easier to find than in aggregated or hand-corrected ones. Especially in ATIS, many algorithms reach close-to-perfect scores, in particular scorer (> 0.9 AP). We attribute this to the artificial noise injected. The more difficult datasets have usually natural noise patterns that are often harder to solve (Amiri, Miller, and Savova 2018; Larson et al. 2019; Hedderich, Zhu, and Klakow 2021). The three SLOT INCONSISTENCIES datasets are also easy compared to CONLL-2003. On some datasets with real errors—PLANK and SLOT INCONSISTENCIES—the performance of the best methods is already quite good with F1 ≈ 0.5 and AP ≈ 0.4 for PLANK and F1, AP > 0.65 for SLOT INCONSISTENCIES.

Overall, methods that work well are *Classification Uncertainty* (CU), *Confident Learning* (CL), *Curriculum Spotter* (CS), *Datamap Confidence* (DM), *Diverse Ensemble* (DE), *Label Aggregation* (LA), *Leitner Spotter* (LS), *Projection Ensemble* (PE), and *Retag* (RE). Aggregating scorer judgments via *Borda Count* (BC) can improve performance and deliver the second-best AP score based on the harmonic mean. The downside here is very high total runtime (the sum of runtimes of individual scores aggregated), as it requires training

Table 3
F1 and **AP** for all implemented flaggers ● as well as scorers ● evaluated with the best overall setups. We also report the harmonic mean **H** ● computed across all datasets. **L**abel **A**ggregation, **Re**tag, **D**iverse **E**nsemble, and **B**orda **C**ount perform especially well across tasks and datasets. Datasets created via injecting random noise (ATIS, SST, and GUM) are comparatively easier to detect errors in.

	Text			Token		Span				
Method	ATIS	IMDb	SST	GUM	Plank	Comp.	CoNLL	Flights	Forex	H
Flagger										
CL	0.35	0.33	0.34	0.80	0.37	0.50	0.24	0.42	0.57	0.39
DE	0.72	0.30	0.33	0.74	0.48	0.57	0.28	0.55	0.64	0.45
IRT	0.00	0.01	0.02	0.00	0.12	0.41	0.29	0.02	0.62	0.00
LA	0.83	0.33	0.35	0.68	0.49	0.59	0.30	0.66	0.70	0.48
PE	0.54	0.18	0.34	0.58	0.50	0.56	0.25	0.29	0.56	0.36
RE	0.81	0.33	0.34	0.69	0.49	0.64	0.32	0.67	0.70	0.49
VN	·	·	·	0.55	0.30	0.11	0.02	0.29	0.14	0.08
Scorer										
BC	0.98	0.35	0.50	0.92	0.38	0.68	0.14	0.49	0.54	0.41
CS	0.97	0.29	0.21	·	·	·	·	·	·	0.33
CU	0.87	0.28	0.27	0.98	0.42	0.70	0.17	0.68	0.70	0.41
DM	0.98	0.25	0.49	0.95	0.27	0.66	0.14	0.35	0.61	0.36
DU	0.05	0.06	0.05	0.05	0.24	0.43	0.07	0.18	0.32	0.08
KNN	0.13	0.05	0.11	0.21	0.31	0.61	0.12	0.07	0.16	0.12
LE	·	·	·	0.60	0.22	0.41	0.19	0.10	0.11	0.18
LS	0.91	0.31	0.46	·	·	·	·	·	·	0.46
MD	0.14	0.03	0.08	0.12	0.16	0.54	0.06	0.07	0.14	0.08
PM	0.06	0.05	0.05	0.05	0.23	0.54	0.06	0.12	0.25	0.08
WD	·	·	·	0.53	0.39	0.45	0.16	0.11	0.14	0.20

instances of all scorers beforehand, which already perform very well (H_{AP} of *Borda Count* is 0.41 and the best individual scorer has H_{AP} of 0.46). While aggregating scores requires well performing scorers (3 in our setup, see § 1.2) it is more stable across tasks than using individual methods on their own. Most model-based methods (*Classification Uncertainty*, *Confident Learning*, *Diverse Ensemble*, *Label Aggregation*, *Retag*) perform very well overall, but methods based on training dynamics that do not need cross-validation (*Curriculum Spotter*, *Datamap Confidence*, *Leitner Spotter*) are on par or better. In particular, *Datamap Confidence* shows a very solid performance and can keep up with the closely related *Classification Uncertainty*, sometimes even outperforming it while not needing CV. *Confident Learning* specifically has high precision for token and span labeling.

Amiri, Miller, and Savova (2018) argue that prediction loss is not enough to detect incorrect instances because easy ones still can have a large loss. Therefore, more intricate methods like *Leitner Spotter* and *Curriculum Spotter* are needed. We do not observe a large difference between *Classifier Uncertainty* and the two, though. *Datamap Confidence*, as a more complicated sibling of *Classification Uncertainty*, however, outperforms these from time to time, indicating that training dynamics offers an advantage over simply using class probabilities.

Variation n-grams (VN) has high precision and tends to be conservative in flagging items, that is, exhibit low false positives, especially for span classification. *Weighted Discrepancy* works overall better than *Label Entropy*, but both methods almost always perform worse than more intricate ones. When manually analyzing their scores, they mostly assign a score of 0.0 and rarely a different score (less than 10% from our observation, often even lower). This is because there are only very few instances with both surface form overlap and different labels. While the scores for *Prediction Margin* appear to be not good, the original paper (Dligach and Palmer 2011) reports a similarly low performance while their implementation of *Retag* reaches scores that are around two times higher (10% vs. 23% precision and 38% vs. 60% recall). This is similar to our observations. One potential reason why *Classification Uncertainty* produces better results than the related *Prediction Margin* is that the latter does not take the given label into account; it always uses the difference between the two most probable classes. Using a formulation of *Projection Ensemble* that uses the label did not improve results significantly, though.

Methods based on vector proximity—*k-Nearest Neighbor Entropy* (KNN) and *Mean Distance* (MD)—perform sub-par across tasks and datasets. We attribute this to issues in distance calculation for high-dimensional data, as noted for instance by Cui Zhu, Kitagawa, and Faloutsos (2005) in a related setting. In high-dimensional vector spaces, everything can appear equidistant (curse of dimensionality). Another performance-relevant issue is the embedding quality. In Grivas et al. (2020), KNN is used with domain-specific embeddings for biomedical texts. These could have potentially improved performance in their setting, but they do not report quantitative results, though, which makes a comparison difficult. With regard to *Mean Distance*, we only achieve $H = 0.08$. On real data for intent classification, Larson et al. (2019) achieve an average precision of around 0.35. They report high recall and good average precision on datasets with random labels but do not report precision on its own. Their datasets contain mainly paraphrased intents, which makes it potentially easier to achieve good performance. This is similar to how AED applied on our randomly perturbed ATIS dataset resulted in high detection scores. Code and data used in their original publication are no longer available. We were therefore not able to reproduce their reported performances with our implementation and on our data.

Item Response Theory (IRT) does not perform well across datasets and tends to overly flag instances. Therefore, it is preferable to use the model predictions in a *Diverse Ensemble*, which yields much better performance. IRT is also relatively slow for larger corpora as it is optimized via variational inference and needs many iterations to converge. Our hypothesis is that *Item Response Theory* needs more subjects (in our case models) to better estimate discriminability. Compared to our very few subjects (seven for text classification and four for token and span labeling), Rodriguez et al. (2021) used predictions of the SQuAD leaderboard with 161 development and 115 test subjects. To validate this hypothesis, we rerun *Item Response Theory* on the unaggregated predictions of *Projected Ensemble*. While this leads to slightly better performance, it still does not work as well as using predictions in *Diverse Ensemble* or *Projected Ensemble* directly. As it is often unfeasible to have that many models providing predictions, we see *Item Response Theory* only useful in very specific scenarios.

Regarding *Dropout Uncertainty*, after extensive debugging with different models, datasets, and formulations of the method, we were not able to achieve comparably good results with other AED methods evaluated in this work. On real data, Amiri, Miller, and Savova (2018) also report relatively low performances similar to ours. Our implementation delivers results similar to Shelmanov et al. (2021) on misclassification detection. In their paper, the reported scores appear to be very high. But we consider their reported scores an overestimate, as they use ROC AUC (which is overconfident for imbalanced datasets) and not AP to evaluate their experiments. Even when applying the method on debug datasets with the most advantageous conditions that are solvable by other methods with perfect scores, *Dropout Uncertainty* only achieves AP values of around 0.2. The main reason we see for the overall low scores for *Dropout Uncertainty* is that the different repeated prediction probabilities are highly correlated and do not differ much overall. This is similar to the observations of Shelmanov et al. (2021).

Qualitative Analysis. To better understand for which kinds of errors methods work well or fail, we manually analyze the instances in CONLL-2003. It is our dataset of choice for three reasons: (1) span labeling datasets potentially contain many different errors, (2) it is annotated and corrected by humans, and (3) it is quite difficult for AED to find errors in it, based on our previous evaluation (see Table 3). For spans whose noisy labels disagree with the correction, we annotate them as either being inconsistent, a true error, an incorrect correction, or a hallucinated entity. Descriptions and examples for each type of error are given in the following.

True errors are labels that are unambiguously incorrect, for instance, in the sentence *NATO military chiefs to visit Iberia*, the entity *Iberia* was annotated as ORG but should be LOC, as it refers to the Iberian peninsula.

Inconsistencies are instances that were assigned different labels in similar contexts. In CONLL-2003, these are mostly from sports teams that were sometimes annotated as LOC and sometimes as ORG.

Incorrect correction In very few cases, the correction introduced a new error, for example, *United Nations* was incorrectly corrected from ORG to LOC.

Hallucinated entity are spans that were labeled to contain an entity, but they should not have been annotated at all. For example, in the sentence *Returns on treasuries were also in negative territory*, *treasuries* was annotated as MISC but does not contain a named entity. Sometimes, entities that should consist of one span were annotated originally as two entities. This results in one unmatched entity after alignment. We consider this a hallucinated entity as well.

CONLL-2003 was corrected manually by Reiss et al. (2020). After aligning (see § 4.3), we find that there are in total 293 errors. We group them by difficulty based on how often methods were able to detect them. For scorers, we consider the instances with the highest 10% scores as flagged, similarly to how we evaluate precision and recall. For span labeling, we implemented a total of 16 methods. The errors detected at least by half of the methods (50%) are considered *easy*, the ones detected by at least four methods (25%) are considered *medium*, and the rest, *hard* (25%). This results in 50 easy, 78 medium, and 165 hard instances. The distribution of error types across difficulty levels is visualized in Figure 2. It can be seen that true errors are easier to detect than inconsistencies by a significant margin: The easy partition consists only of 50% inconsistencies, whereas in the hard partition, it consists of around 75% inconsistent instances. This can be intuitively explained by the fact that the inconsistencies are not rare, but make up a large fraction of all corrections. It is therefore difficult for a method to learn that it should be flagged when it is only given noisy labels.

We further analyze how each method can deal with the different types of errors across difficulty levels. The percentage of correctly detected errors per type and method is depicted in Table 4. It again can be seen that true errors are easier for methods to detect than inconsistencies; inconsistencies of hard difficulty were almost never detected. Interestingly, scorers that are not model-based (*k-Nearest Neighbor Entropy* (KNN), *Label Entropy* (LE), and *Weighted Discrepancy* (WD)) are able to better detect inconsistencies of

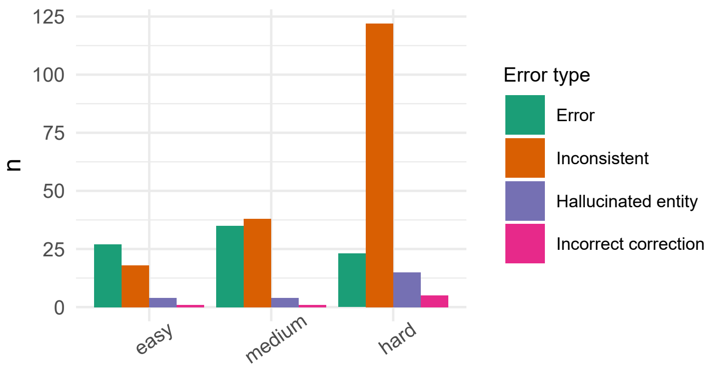


Figure 2
Error counts per difficulty level in CONLL-2003. It can be seen that the number of inconsistencies increases with the difficulty. This indicates that "real" annotation errors are easier to detect than inconsistencies.

Table 4
Percent of errors and inconsistencies detected on CONLL-2003 across methods and difficulty for flaggers ● and scorers ● grouped by error types. It can be seen that real errors (E) are more often detected than inconsistencies (I). Some methods not relying on models (KNN, LE, WD) are sometimes better in spotting inconsistencies than errors, whereas for model-based method it is the opposite. Note that errors concerning incorrect corrections (IC) and hallucinated entities (HE) are quite rare and not reliable to draw conclusions from.

	Flagger							Scorer								
Error	CL	DE	IRT	LA	PE	RE	VN	BC	CU	DM	DU	KNN	LE	MD	WD	PM
Easy																
E	66	96	100	100	100	100	3	92	100	66	25	40	29	14	29	25
I	72	100	100	88	94	94	11	94	100	55	27	61	55	11	55	27
HE	25	100	100	100	100	100	0	100	100	100	100	0	0	50	0	100
IC	0	100	100	100	100	100	0	100	100	100	0	0	0	100	0	0
Medium																
E	40	51	54	88	82	82	0	31	94	31	34	22	0	11	0	25
I	13	26	31	26	52	31	0	15	36	23	21	44	63	15	63	13
HE	0	75	75	25	75	100	0	75	50	75	25	0	0	0	0	50
IC	100	100	100	100	100	100	0	0	100	0	0	0	0	0	0	0
Hard																
E	0	17	13	0	65	0	0	0	0	17	0	17	0	17	0	13
I	0	0	0	0	9	0	0	3	0	5	4	14	4	1	4	4
HE	0	13	13	0	53	6	0	0	0	0	6	0	0	6	0	0
IC	0	20	20	20	20	0	0	0	0	0	20	0	0	20	0	0

medium and sometimes hard difficulty but fail at detecting most errors. We explain this for KNN by the fact that it relies on semantic vector space embeddings that do not rely on the noisy label but on the semantics of its surface form in its context. As neighbors in the space have the same meaning, it is possible to detect errors even though the label is inconsistent in many cases. The same can be said about WD and LE, which rely only on the surface form and how often it is annotated differently. If the correct label is the majority, then it can still detect inconsistencies even if they are quite frequent. But both still do not perform as well as other methods on easier instances; they only find 50% of errors and inconsistencies whereas *Classification Uncertainty* or *Retag* detects almost all (but again fail to find inconsistencies on medium difficulty). *Variation n-grams* (VN), however, do not work well even for easy cases because they rely on contexts around annotations that need to match exactly, which is very rare in this dataset. To summarize, the methods that worked best overall across tasks and datasets are *Borda Count* (BC), *Diverse Ensemble* (DE), *Label Aggregation* (LA), and *Retag* (RE). Inconsistencies appear to be more difficult to detect for most methods, especially for model-based ones. Methods that do not rely on the noisy labels like *k-Nearest Neighbor Entropy*, *Label Entropy*, and *Weighted Discrepancy* were better in finding inconsistencies on more difficult instances when manually analyzing CONLL-2003.

5.2 RQ2 – Does Model Calibration Improve Model-based Method Performance?

Several model-based AED methods, for instance, *Classification Uncertainty*, directly leverage probability estimates provided by a machine learning model (§ 3.2.2). Therefore, it is of interest whether models output class probability distributions that are accurate. For instance, if a model predicts 100 instances and states for all 80% confidence, then the accuracy should be around 0.8. If this is the case for a model, then it is called **calibrated**. Previous studies have shown that models are often not calibrated very well, especially neural networks (Guo et al. 2017). To alleviate this issue, a number of calibration algorithms have been developed. The most common approaches are post hoc, which means that they are applied after the model has already been trained.

Probabilities that are an under- or overestimate can lead to non-optimal AED results. The question arises whether model-based AED methods can benefit from calibration and, if so, to what extent. We are only aware of one study mentioning calibration in the context of annotation error detection. Northcutt, Jiang, and Chuang (2021) claim that their approach does not require calibration to work well, but they did not evaluate it in detail. We only evaluate whether calibration helps for approaches that directly use probabilities and can leverage CV, as calibration needs to be trained on a holdout set. This, for instance, excludes *Curriculum Spotter*, *Leitner Spotter*, and *Datamap Confidence*. Methods that can benefit are *Confident Learning*, *Classifier Uncertainty*, *Dropout Uncertainty*, and *Prediction Margin*.

There are two groups of approaches for post hoc calibration: *parametric* (e.g., Platt Scaling/Logistic Calibration [Platt 1999] or Temperature Scaling [Guo et al. 2017]) or *non-parametric* (e.g., Histogram Binning [Zadrozny and Elkan 2001], Isotonic Regression [Zadrozny and Elkan 2002], or Bayesian Binning into Quantiles [Naeini, Cooper, and Hauskrecht 2015]). On a holdout corpus we evaluate several calibration methods to determine which calibration method to use (see Appendix B). As a result, we apply the best—Platt Scaling—for all experiments that leverage calibration.

Calibration is normally trained on a holdout set. As we already perform cross-validation, we use the holdout set both for training the calibration and for predicting annotation errors. While this would not be optimal if we are interested in generalizing calibrated probabilities to unseen data, we are more interested in downstream task performance. Using an additional fold per round would be theoretically more sound. But our preliminary experiments show that it has the issue of reducing the available training data and thereby hurts the error detection performance more than the calibration helps. Using the same fold for both calibration and applying AED, however, improves overall task performance, which is what matters in our special task setting. We do not leak the values for the downstream tasks (whether an instance is labeled incorrectly or not) but only the labels for the primary task.

To evaluate whether calibration helps model-based methods that leverage probabilities, we train models with cross-validation and then evaluate each applicable method with and without calibration. The same model and therefore the same initial probabilities are used for both. We measure the relative and total improvement in F1 (for flaggers) and AP (for scorers), which are our main metrics. The results are depicted in Figure 3. It can be seen that calibration has the potential of improving the performance of certain methods by quite a large margin. For *Confident Learning*, the absolute gain is up to 3 percentage points (pp) F1 on text classification, 5 pp for token labeling, and up to 10 pp for span labeling. On the latter two tasks, though, there are also many cases with performance reductions. A similar pattern can be seen for *Classification Uncertainty* with up to 2 pp, no impact, and up to 8 pp, respectively. *Dropout Uncertainty* and *Prediction Margin* do not perform well to begin with. But after calibration, they gain 5 to 10 pp AP, especially for span and in some instances for token labeling. In most cases on median, calibration does not hurt the overall performance.

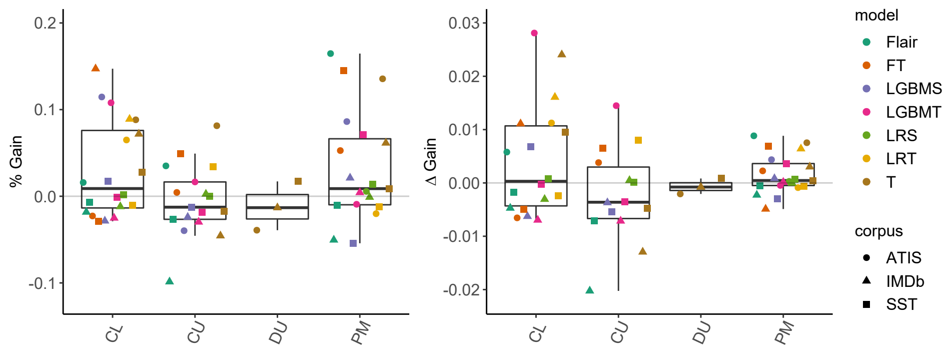
In order to check whether the improvement using calibration is statistically significant, we also use statistical testing. We choose the Wilcoxon signed-rank test (Wilcoxon 1945) because the data is not normally distributed, which is required by the more powerful paired t-test. The alternative hypothesis is that calibration improves method performance, resulting in a one-sided test.

We do not perform a multiple-comparison correction as each experiment works on different data. The p-values can be seen in Table 5. We can see that calibration can improve performance significantly overall in two task and method combinations (text classification + *Confident Learning* and span labeling + *Classification Uncertainty*). For text classification and token labeling, the absolute gain is relatively small. For span labeling, *Classification Uncertainty* benefits the most. The gains for *Dropout Uncertainty* and *Prediction Margins* appear large, but these methods do not perform well in the first place. Hence, our conclusion is that calibration can help model-based AED performance but it is very task- and dataset-specific.

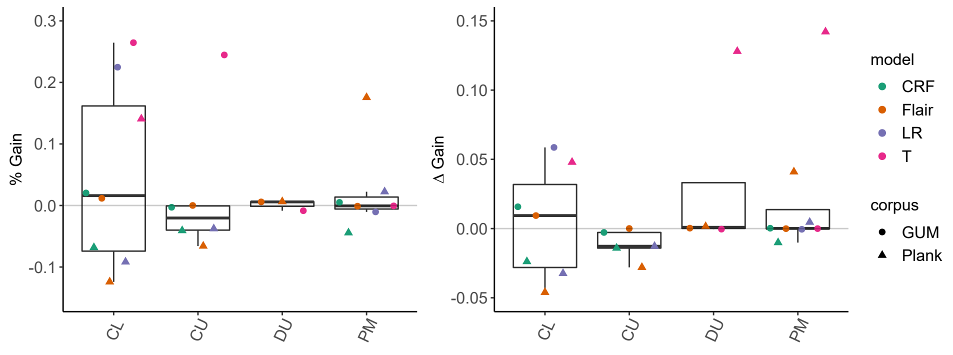
We do not see a clear tendency for which models or datasets benefit the most from calibration. More investigation is needed regarding which calibrator works best for which model and task. We chose the one that reduces the calibration error the most, which is not necessarily the best choice for each setting.

5.3 RQ3 – To What Extent Are Model and Detection Performance Correlated?

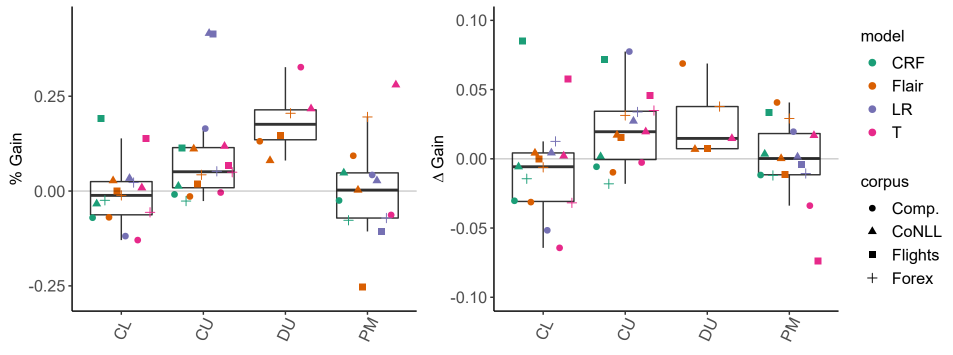
Several AED methods directly use model predictions or probabilities to detect potential annotation errors. This raises the question of how model performance impacts AED performance. Reiss et al. (2020) state that they deliberately use simpler models to find more potential errors in CONLL-2003 and therefore developed *Projection Ensemble*, an ensemble of logistic regression classifiers that use BERT embeddings reduced by different Gaussian projections. Their motivation is to obtain a diverse collection of predictions to have disagreements. They conjecture that using very well-performing models might be detrimental to AED performance as their predictions potentially would not differ that much from the noisy labels as the models learned predicting the noise. In contrast to that, Barnes, Øvrelid, and Velldal (2019) use state-of-the-art models to find annotation errors in different sentiment datasets. But neither Reiss et al. (2020) nor Barnes, Øvrelid, and Velldal (2019) directly evaluate AED performance—rather, they use AED to clean noisy datasets for which the gold labels are unknown. Therefore, the question of how much model and detection performance are correlated has not yet been thoroughly evaluated.



(a) Text classification



(b) Token labeling



(c) Span labeling

Figure 3
Relative and total improvement of model-based AED methods over different corpora, methods, and models when calibrating probabilities. It can be seen that calibration can lead to good improvements, while on median mostly not hurting performance. This plot is best viewed in the electronic version of this paper. Not displayed are extreme (positive) outlier points.

Table 5
p-values forWilcoxon signed-rank test.We check whether calibration improves AED performance on a statistically significant level. Underlined values are significant with $p < 0.05$.

Method	Text	Token	Span
Confident Learning (CL)	<u>0.021</u>	0.230	0.665
Classification Uncertainty (CU)	0.121	0.320	<u>0.003</u>
Dropout Uncertainty (DU)	0.750	0.188	0.320
Prediction Margin (PM)	0.064	0.273	0.628

For answering this question, we leverage the fact that we implemented several models of varying performance for each task. We use two complementary approaches to analyze this question. First, we measure the correlation between model and task performances for the overall score, precision, and recall. Then, we analyze which models lead to the best AED performance.

Throughout this section, scores for flaggers and scorers are coalesced; overall score corresponds to F1 and AP, precision to precision and precision@10%, recall to recall and recall@10%. For reference, model performances are shown in Figure C.1. We choose micro aggregation for measuring model performance, as we are interested in the overall scores and not the scores per class. Using macro aggregation yields qualitatively similar but less significant results.

Correlation. In order to determine whether there exists a positive or negative relationship between model and method performances, we compute Kendall's τ coefficient (Kendall 1938) for each method and dataset. The results are depicted in Table 6. We see that when the test is significant with $p < 0.05$, then there is almost always a moderate to strong monotonic relationship.[5] τ is zero or positive for classification and token labeling, hinting that there is either no relationship or a positive one. For span labeling we observe negative correlation for precision and overall. It is significant in one case only.

One issue with this test is its statistical power. In our setting, it is quite low due to the few samples available per method and task. It is therefore likely that the null hypothesis—in our case, the assumption that there is no relationship between model and method performances—is not rejected even if it should have been. Hence, we next perform additional analysis to see which models overall lead to the best model performances.

Which Models Lead to the Best Method Performances. In order to further analyze the relationship between model and AED performances, we look at which model leads to the best performance on a given dataset. In Figure 4 we show the results differentiated by overall, precision, and recall scores. We observe that in the most cases, the best or second best models lead to the best method performances. It is especially clear for token labeling, where using Flair leads to the best performance in all cases if we look at the overall and precision score. Interestingly, Flair has better performance than transformers for span labeling but the latter is preferred by most methods. Flair only leads to best method performances for most of CONLL-2003 and parts of FLIGHTS. Besides the fact that better models on average lead to better AED performance, we do

5 $|\tau| > 0.07$ indicates a weak, $|\tau| > 0.21$ a moderate, $|\tau| > 0.35$ indicates a strong monotonic relation.

Table 6
Kendall's τ coefficient grouped by task and method measured across datasets. For p, the null hypothesis is $\tau = 0$ and the alternative hypotheses is $\tau \neq 0$. Underlined are significant p-values with $p < 0.05$. Positive correlation is highlighted 🔵, negative correlation is highlighted 🔴.

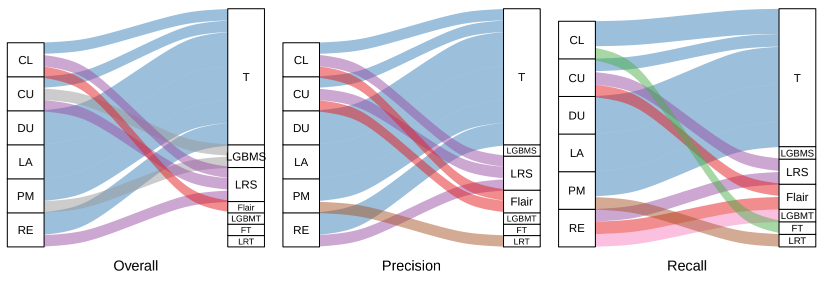
	Overall		Precision		Recall	
Method	τ	p	τ	p	τ	p
Text						
CL	+0.495	0.002	+0.657	0.000	−0.373	0.018
CU	+0.486	0.002	+0.396	0.013	+0.705	0.000
DU	+0.333	0.602	+0.333	0.602	+0.333	0.602
LA	+0.333	0.602	+1.000	0.117	+0.333	0.602
PM	+0.143	0.365	+0.211	0.184	+0.230	0.147
RE	+0.571	0.000	+0.600	0.000	+0.412	0.011
Token						
CL	+0.929	0.001	+0.929	0.001	+0.214	0.458
CU	+0.714	0.013	+0.786	0.006	+0.714	0.013
DU	−0.333	0.497	−0.333	0.497	+0.333	0.497
LA	+1.000	0.042	+0.667	0.174	+0.667	0.174
PM	+0.000	1.000	+0.000	1.000	+0.000	1.000
RE	+0.857	0.003	+0.714	0.013	+0.357	0.216
Span						
CL	−0.033	0.857	−0.183	0.322	+0.101	0.588
CU	+0.017	0.928	−0.250	0.177	+0.300	0.105
DU	−0.429	0.138	−0.429	0.138	+0.286	0.322
LA	−0.357	0.216	−0.643	0.026	+0.143	0.621
PM	−0.317	0.087	−0.319	0.086	+0.202	0.279
RE	−0.167	0.368	−0.217	0.242	−0.109	0.558

not see a consistent pattern that certain methods prefer certain models. A special case, however, is the recall of *Retag*. We indeed observe the assumption of Reiss et al. (2020) that the model with the lowest recall often leads to the highest AED recall (see Figure 4). This is especially pronounced for token and span labeling. For these tasks, *Retag* can use a low-recall model to flag a large fraction of tokens because the model disagrees at many positions with the input labels. This improves recall while being detrimental to precision.

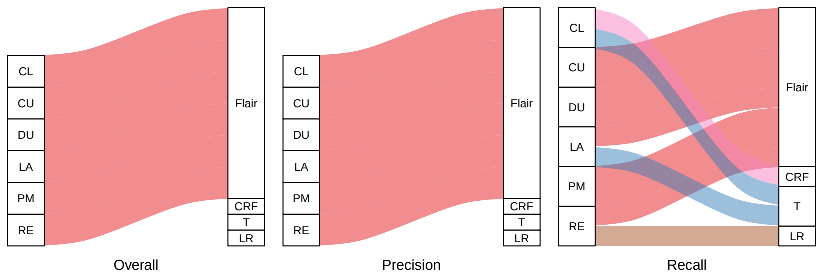
To summarize, overall we see positive correlation between model and AED performances. Using a well-performing model is a good choice for most model-based AED approaches. Neural models perform especially well, although they are more expensive to train. We therefore use transformers for text classification as well as span labeling and Flair for token labeling. Using a low-recall model for *Retag* leads to higher recall for token and span labeling, as conjectured by Reiss et al. (2020). This, however, concurs with lower precision and excessive flagging and thus more annotations need to be inspected.

5.4 RQ4 – What Performance Impact Does Using (or not Using) Cross-validation Have?

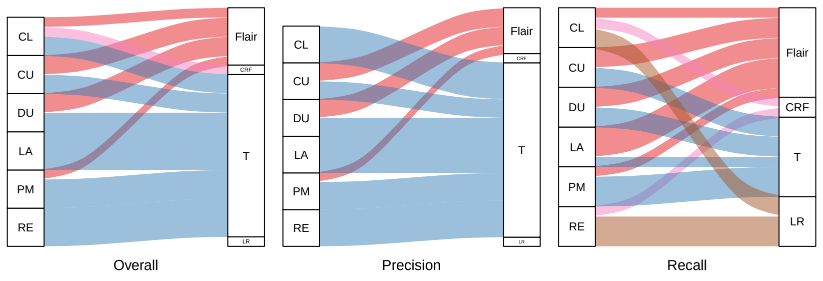
Model-based AED approaches are typically used together with CV (e.g., Amiri, Miller, and Savova 2018, Larson et al. 2020, Reiss et al. 2020). Northcutt, Jiang, and Chuang (2021) explicitly state that *Confident Learning* should only be applied to out-of-sample



(a) Text classification



(b) Token labeling



(c) Span labeling

Figure 4
Model-based methods and how often which model type leads to the best method performance with respect to overall, precision, and recall score. A connection from left to right between a method and a model indicates that using that method with outputs from that model leads to the best task performance. The color of the connection indicates the chosen model, for instance, Flair is 🔴, Transformer 🔵. The model axis is presented in descending order by model performance, aggregated by Borda Count across datasets. This figure is best viewed in color.

predicted probabilities. Amiri, Miller, and Savova (2018) do not mention that they used CV for *Dropout Uncertainty*, *Label Aggregation*, or *Classification Uncertainty*.

When using AED with CV, models are trained on $k - 1$ splits and then detection is done on the remaining k-th set. After all unique folds are processed, all instances are checked. CV is often used in supervised learning where the goal is to find a model configuration as well as hyperparameters that generalize on unseen data. The goal of AED, however, is to find errors in the data at hand. Resulting models are just an instrument and not used afterwards. They therefore will not be applied to unseen data and need not generalize to data other than the one to clean. Hence, the question arises whether CV is really necessary for AED, which has not been analyzed as of yet. Not using CV has the advantage of being much faster and using less energy, since using

Table 7
Performance delta of model-based methods when training models with and without CV. Negative 🔴 values indicate that not using CV performs worse than using it, positive 🔵 values the opposite. It can be seen that overall recall is strongly impacted when not using CV but precision can improve. Flagger and scorer results are separated by a gap.

	Text			Token		Span			
Method	ATIS	IMDb	SST	GUM	Plank	Comp.	CoNLL	Flights	Forex
Δ Precision									
CL	+0.51	+0.77	−0.22	+0.16	+0.27	+0.02	+0.61	+0.08	+0.27
DE	+0.41	+0.11	+0.21	+0.26	+0.23	+0.34	−0.26	+0.59	+0.36
IRT	+0.02	+0.38	+0.03	+0.00	+0.04	+0.07	−0.27	+0.03	−0.41
LA	+0.26	+0.06	+0.28	−0.00	+0.23	+0.01	−0.05	+0.29	+0.32
PE	+0.05	+0.00	+0.01	+0.01	+0.04	+0.14	+0.01	+0.22	+0.11
RE	+0.29	+0.78	+0.19	+0.15	+0.17	−0.03	−0.10	+0.19	+0.29
CU	−0.03	−0.12	−0.20	−0.00	−0.05	−0.05	−0.13	−0.08	−0.29
DU	+0.34	−0.01	+0.12	+0.04	+0.14	+0.00	−0.03	+0.07	+0.14
PM	+0.34	−0.05	+0.05	+0.03	+0.14	−0.17	−0.02	+0.05	+0.07
Δ Recall									
CL	−0.04	−0.63	−0.82	−0.07	−0.25	−0.18	−0.17	−0.24	−0.46
DE	−0.29	−0.25	−0.41	−0.10	−0.34	−0.31	−0.30	−0.39	−0.37
IRT	+0.44	+0.06	+0.63	+0.09	+0.37	+0.26	−0.31	+0.76	−0.11
LA	−0.27	−0.64	−0.83	−0.00	−0.30	−0.18	−0.27	−0.34	−0.49
PE	+0.00	+0.00	−0.00	−0.00	−0.14	−0.13	−0.06	+0.00	−0.11
RE	−0.32	−0.64	−0.82	−0.00	−0.35	−0.17	−0.30	−0.44	−0.55
CU	−0.07	−0.62	−0.40	−0.00	−0.04	−0.01	−0.24	−0.22	−0.28
DU	+0.71	−0.05	+0.25	+0.09	+0.12	+0.00	−0.06	+0.17	+0.14
PM	+0.71	−0.23	+0.09	+0.06	+0.12	−0.04	−0.04	+0.12	+0.07
Δ % Flagged									
CL	−0.02	−0.06	−0.19	−0.02	−0.07	−0.09	−0.02	−0.01	−0.06
DE	−0.05	−0.05	−0.15	−0.03	−0.10	−0.31	−0.06	−0.06	−0.11
IRT	+0.06	−0.91	+0.16	+0.03	+0.13	+0.15	−0.06	+0.07	+0.79
LA	−0.03	−0.06	−0.19	+0.00	−0.10	−0.10	−0.05	−0.03	−0.09
PE	−0.01	−0.00	−0.01	−0.00	−0.04	−0.15	−0.03	−0.08	−0.05
RE	−0.04	−0.06	−0.19	−0.02	−0.10	−0.08	−0.05	−0.03	−0.09

CV increases training time linearly with the number of folds. In the typical setup with 10-fold CV, this means an increase of training time by 10×.

To answer this question we train a single model on all instances and then predict on the very same data. Then we use the resulting outputs to rerun methods that used CV before, which are *Classification Uncertainty*, *Confident Learning*, *Diverse Ensemble*, *Dropout Uncertainty*, *Item Response Theory*, *Label Aggregation*, *Prediction Margin*, *Projection Ensemble*, and *Retag*. The results are listed in Table 7. Overall, it can be seen that not using CV massively degrades recall for model-based methods while the precision improves. This can be intuitively explained by the fact that if the underlying models have already seen all the data, then they overfit to it and hence can re-predict it well. Due to the positive relationship between model and method performances (see § 5.3) this is also reflected downstream; fewer instances are predicted differently than the original labels. This reduces recall and thereby the chance of making errors, thus increasing precision. This can be seen by the reduction in the percentage of flagged instances for flaggers. Interestingly, *Dropout Uncertainty* and *Prediction Margin* are not impacted as much and sometimes even improve when not using CV across all scores, especially for easier datasets. Recall of *Item Response Theory* also improves at the cost of more flagged items and a reduction in precision. *Prediction Ensemble* for text classification is relatively unaffected and for token and span labeling, the performance difference is around ± 0.10 pp. Therefore, it might be a good tradeoff to not use CV with this method as it is already expensive due to its ensembling.

To summarize, not using CV can negatively impact performance—in particular, degrading recall. We therefore recommend the use of CV, even though it increases runtime by the number of folds (in our case, by a factor of ten). In settings where this is an issue, we recommend using methods that inherently do not need CV. These include most heuristics and well-performing approaches like *Datamap Confidence*, *Leitner Spotter*, or *Curriculum Spotter*. If precision is more important than recall, then not using CV might be taken into consideration.

6. Takeaways and Recommendations

This article has probed several questions related to annotation error detection. Our findings show that it is usually better to use well-performing models for model-based methods as they yield better detection performance on average. Using a worse model for *Retag* improves recall at the cost of lower precision. For detection, these models should be trained via cross-validation, otherwise the recall of downstream methods is heavily degraded (while the precision improves). Calibration can improve these model-based annotation error detection methods, but more research is needed to determine when exactly it can be useful. Some model-method combinations achieved relatively large gains after calibration while others did not improve.

Methods that are used frequently in practice—*Retag* and *Classification Uncertainty*—performed well in our experiments. Others did not perform particularly well, especially *Dropout Uncertainty*, *Item Response Theory*, *k-Nearest Neighbor Entropy*, *Mean Distance*, and *Prediction Margin*. For *Mean Distance* in particular, Larson et al. (2019) reported AP of > 0.6 and recall > 0.8 on corpora with artificial noise, which we could not reproduce. Experiments with *Dropout Uncertainty* disseminated in Amiri, Miller, and Savova (2018) reached similar high scores as using *Curriculum Spotter*, *Leitner Spotter*, or *Classification Uncertainty*, but we were not able to make *Dropout Uncertainty* reach similar high scores as the others. *Label Aggregation*, though, which uses the same inputs, performs exceedingly well. For the others, either no scores were reported or they were similarly low as in our experiments.

Experiments on actual corpora have shown that AED methods still have room for improvement. While looking promising on artificial corpora, there is a large performance drop when applying them in practice. Overall, the methods that worked best are *Classification Uncertainty*, *Confident Learning*, *Curriculum Spotter*, *Datamap Confidence*, *Diverse Ensemble*, *Label Aggregation*, *Leitner Spotter*, *Projection Ensemble*, and *Retag*. More complicated methods are not necessarily better. For instance, *Classification Uncertainty* and *Retag* perform well across tasks and datasets while being easy to implement. Model-based methods require k-fold cross-validation. Therefore, if runtime is a concern, then *Datamap Confidence* is a good alternative. It performs well while only needing to train one model instead of k. In case the data or its corresponding task to correct is not suitable for machine learning, methods like *Label Entropy*, *K-Nearest-Neighbor Entropy*, or *Variation n-grams* still can be applied. As the latter usually has high precision it is often worthwhile to apply it whenever the data is suitable for it; that is, if the data has sufficient surface form overlap. Individual scorer scores can be aggregated via *Borda Count* but it tremendously increases runtime. While not yielding significantly better results in our experiments, results aggregated that way were much more stable across datasets and tasks while individual scorers sometimes had performance drops in certain settings.

Manual analysis of CONLL-2003 showed that finding inconsistencies is often more difficult than finding annotation errors. While model-based methods were often quite good in the latter, they performed poorly when detecting inconsistencies. Methods that do not rely on the noisy labels but on the surface form or semantics like *k-Nearest Neighbor Entropy*, *Label Entropy*, and *Weighted Discrepancy* have shown the opposite behavior. They each have their own strengths and it can be worth combining both types of methods.

7. Conclusion

Having annotated corpora with high-quality labels is imperative for many branches of science and for the training of well-performing and generalizing models. Previous work has shown that even commonly used benchmark corpora contain non-negligible numbers of annotation errors. In order to assist human annotators with detecting and correcting these errors, many different methods for annotation error detection have been developed. To date, however, methods have not been compared, so it has been unclear what method to choose under what circumstances. To close this gap, we surveyed the field of annotation error detection, reimplemented 18 methods, collected and generated 9 datasets for text classification, token labeling, and span labeling, and evaluated method performance in different settings. Our results show that AED can already be useful in real use cases to support data cleaning efforts. But especially for more difficult datasets, the performance ceiling is far from reached yet.

In the past, the focus of most works researching or using AED was to clean data and not to develop a method. The method was only a means to achieve a cleaned corpus and not the target itself. Also, several studies proposed algorithms for different use cases and AED was one application to it just mentioned briefly at the end without in-depth evaluation, rendering it unclear how well the method performs. We therefore strongly encourage authors who introduce new AED methods to compare their method to previous work and on the same corpora to foster reproducibility and to bring the performance of new methods into context. This article surveys, standardizes, and answers several fundamental questions regarding AED so that future work has a stable footing for research. For this, we also make our implementation and datasets publicly available.

Limitations and Future Work. While we thoroughly investigated many available methods on different datasets and tasks, there are some limitations to our work. First, the datasets that we used were only in English. Therefore, it would be interesting to investigate AED on different languages. One first step could be the work by Hedderich, Zhu, and Klakow (2021), who created a corpus for NER in Estonian with natural noise patterns. Hand-curated datasets with explicitly annotated errors are rare. We therefore also used existing, clean datasets and injected random noise, similarly to previous works. These datasets with artificial errors have been shown to overestimate the ability of AED methods, but are still a good estimator for the maximal performance of methods. The next step is to create benchmark corpora that are designed from the ground up for the evaluation of annotation error detection. As creating these requires effort and is costly, a cheaper way is to aggregate raw crowdsourcing data. This is often not published along with adjudicated corpora, so we urge researchers to also publish these alongside the final corpus.

AED was evaluated on three different tasks with nine NLP datasets. The tasks were chosen based on the number of datasets and model types available to answer our research questions. Most AED methods are task-agnostic; previous work, for instance, investigated question answering (Amiri, Miller, and Savova 2018) or relation classification (Alt, Gabryszak, and Hennig 2020; Stoica, Platanios, and Poczos 2021). Hence, AED can and has been applied in different fields like computer vision (Northcutt, Athalye, and Mueller 2021). But these works are plagued by the same issues that most previous AED works have (e.g., only limited comparison to other works and quantitative analysis, code and data not available). Having several fundamental questions answered in this article, future work can now readily apply and investigate AED on many different tasks, domains, and in many different settings, while leveraging our findings (which are summarized in § 6). It would especially be interesting to evaluate and apply AED on more hierarchical and difficult tasks, such as semantic role labeling or natural language inference.

While we investigated many relevant research questions, these were mostly about model-based methods as well as flaggers. To date, scorers have been treated as a black box, so it would be worth investigating what makes a good scorer—for example, what makes *Classification Uncertainty* better than *Prediction Margin*. Also, leveraging scorers as uncertainty estimates for correction is a promising application, similar to the works of Dligach and Palmer (2011) or Angle, Mishra, and Sharma (2018).

This work also only focuses on errors, inconsistencies, and ambiguities related to instance labels. Some datasets also benefit from finding errors concerning tokenization, sentence splitting, or missing entities (e.g., Reiss et al. 2020). We also did not investigate the specific kinds of errors made. This can be useful information and could be leveraged by human annotators during manual corrections. It would be especially interesting to investigate the kinds of errors certain models and configurations were able to correct—for instance, whether using no cross-validation finds more obvious errors but with a higher precision. We leave detection of errors other than incorrect labels or error kind detection for future work because we did not find a generic way to do it across the wide range of evaluated datasets and tasks used in this article.

Finally, we implemented each method as described and performed only basic hyperparameter tuning. We did not tune them further due to the prohibitive costs for our large-scale setup. This is especially true for the considered machine learning models, where we kept the parameters mostly default for all regardless of the dataset and domain. We are sure that one can certainly improve scores for each method, but our implementations should still serve as a lower bound. However, we do not expect large gains from further optimization and no large shifts in ranking between the methods.

Appendix A. Hyperparameter Tuning and Implementation Details

In this section we briefly describe implementation details for the different AED methods used throughout this article. As the tuning data we select one corpus for each task type. For text classification we subsample 5,000 instances from the training split of AG NEWS (Zhang, Zhao, and LeCun 2015); the number of samples is chosen as it is around the same data size as our other datasets. As the corpus for token labeling we choose the English part of PARTUT (Sanguinetti and Bosco 2015) and their POS annotations and inject 5% random noise. For span labeling we use CONLL-2003 (Reiss et al. 2020) to which we apply 5% flipped label noise.

A.1 Aggregating Probabilities and Embeddings for Span Labeling

When converting BIO-tagged sequences to spans for alignment (see § 4.3) consisting only of start and end position as well as its label, the probabilities assigned to each BIO-tag representing the span need to be aggregated. The same needs to be done for creating span embeddings from token embeddings. As an example, consider NER for persons and locations with a tagset of `B-PER`, `I-PER`, `B-LOC`, `I-LOC`. It has to be aggregated so that spans have labels `PER` or `LOC`. Look at a span of two tokens that has been tagged as `B-PER, I-PER`. Then the probability for `PER` needs to be aggregated from the `B-PER` and `I-PER` tags. We evaluate our CONLL-2003 tuning data. We use a Maxent sequence tagger to evaluate *Confident Learning* with 10-fold cross-validation for this hyperparameter selection. In addition, for *k-Nearest-Neighbor Entropy* we evaluate aggregation schemes to create span embeddings from individual token embeddings. Overall, we do not observe a large difference between max, mean, or median aggregation. The results can be seen in Table A.1. We choose aggregating via arithmetic mean because it is slightly better in terms of F1 and AP than the other methods.

A.2 Method Details

In the following we describe the choices we made when implementing the various AED methods evaluated in this article.

Diverse Ensemble Our diverse ensemble uses the predictions of all different model types trained for the task and dataset, similarly to Loftsson (2009), Alt, Gabryszak, and Hennig (2020), and Barnes, Øvrelid, and Velldal (2019).

Spotter and Datamap Confidence The implementations of *Datamap Confidence* as well as *Curriculum Spotter* and *Leitner Spotter* require callbacks or a similar functionality to obtain predictions for every epoch which only HuggingFace Transformers provide. That is why we only evaluate these methods in combination with a transformer.

Dropout Uncertainty In our implementation we use mean entropy, which we observed in preliminary experiments to perform slightly better overall than the other version evaluated by Shelmanov et al. (2021).

Variation *n*-grams We follow Wisniewski (2018) for our implementation and use generalized suffix trees to find repetitions. If there are repetitions of length more than one in the surface forms that are tagged differently, we look up the respective tag sequence that occurs most often in the corpus and flag the positions of all other repetitions where

Table A.1
Impact of different aggregation functions for span alignment and embeddings.

	CL			KNN		
Aggregation	P	R	F1	AP	P@10%	R@10%
min	0.149	0.594	0.238	0.307	0.325	0.326
max	0.761	0.881	0.817	0.308	0.325	0.326
mean	0.766	0.878	0.818	0.318	0.331	0.333
median	0.765	0.876	0.817	0.316	0.330	0.332

they disagree with the majority tags. We convert tokens and sentences to lower case to slightly increase recall while slightly reducing precision. We do not flag an instance if its label is the most common label. This yields far better results as the most common label is most often correct and should not be flagged. When using *Variation n-grams* for span labeling, we use a context of one token to the left and right of the span, similarly to Larson et al. (2020).

Projection Ensemble In our implementation we flag an instance if the majority label of the ensemble disagrees with the given one.

Label Aggregation In the original work that evaluated using *Label Aggregation* for AED (Amiri, Miller, and Savova 2018), MACE (Hovy et al. 2013) was used. We use Dawid-Skene (Dawid and Skene 1979), which has similar performance as MACE (Paun et al. 2018) but many more available implementations (we use Ustalov et al. (2021)). The difference between the two (MACE modeling annotator spam) is not relevant here.

Mean Distance We compare different embedding methods and metrics for *Mean Distance*. For that we use the Sentence Transformers[6] library and evaluate S-BERT embeddings (Reimers and Gurevych 2019), Universal Sentence Encoder (Cer et al. 2018), and average GloVe embeddings (Pennington, Socher, and Manning 2014). We evaluate on our AG NEWS tuning data. As our Universal Sentence Encoder implementation we use `distiluse-base-multilingual-cased-v1` from Sentence Transformers. The Universal Sentence Encoder embeddings as used in the original implementation of *Mean Distance* (Larson et al. 2019) overall perform not better than all S-BERT embeddings. `lof` refers to Local Outlier Factor, a clustering metric proposed by Breunig et al. (2000). Using *all-mpnet-base-v2* together with Euclidean distance works best here and we use this throughout our experiments.

Item Response Theory We use the setup from Rodriguez et al. (2021), that is, a 2P IRT model that is optimized via variational inference and the original code of the authors. We optimize for 10,000 iterations. *Item Response Theory* uses the collected predictions of all models for the respective task, similarly to *Diverse Ensemble*.

Label Entropy and Weighted Discrepancy We implement equations (2) and (3) in Hollenstein, Schneider, and Webber (2016) but assign the minimum score (meaning no error) if the current label is the most common label. This yields far better results because the most common label is most often correct and should not be downranked.

K-Nearest-Neighbor Entropy To evaluate which embedding aggregation over transformer layers works best for *k-Nearest-Neighbor Entropy*, we evaluate several different configurations on our PARTUT tuning data. We chose this task and not span labeling as span labeling requires an additional aggregation step to combine token embeddings to span embeddings (see § 1.1). The transformer of choice is RoBERTa (Liu et al. 2019), as it has better performance than BERT while still being fast enough. We also compare with several non-transformer embeddings: GloVe 6B (Pennington, Socher, and Manning 2014), Byte-Pair Encoding (Heinzerling and Strube 2018), and a concatenation of both. We follow Devlin et al. (2019) regarding which configurations to try. The results can be seen in Table A.2. The best scoring embedder is RoBERTa, using only the last layer that will be the configuration used throughout this work for obtaining token and span

6 `https://www.sbert.net/`.

Table A.2
The performance impact of using different embedding types and configurations for KNN entropy on UD ParTUT.

Embedder	AP	P@10%	R@10%
Last Hidden	0.265	0.255	0.256
Sum All Layers	0.237	0.254	0.254
First Hidden	0.230	0.269	0.270
Sum Last 4 Hidden	0.227	0.246	0.246
Second-to-Last Hidden	0.224	0.244	0.244
Concat Last 4 Hidden	0.149	0.193	0.194
Glove	0.148	0.169	0.169
Glove + BPE	0.148	0.175	0.175
BPE	0.147	0.170	0.170

Table A.3
Evaluation of scorers and their aggregation via Borda Count for text classification and span labeling. Highlighted in gray are the runs of Borda Count aggregation.

Method	AP	P@10%	R@10%
BC_{top3}	0.848	0.460	0.947
BC_{top2}	0.824	0.456	0.938
DM	0.819	0.448	0.922
BC_{top5}	0.794	0.454	0.934
LS	0.706	0.448	0.922
CU	0.521	0.426	0.877
MD	0.422	0.344	0.708
CS	0.296	0.390	0.802
BC_{all}	0.268	0.244	0.502
KNN	0.055	0.062	0.128
DU	0.055	0.062	0.128
PM	0.050	0.046	0.095

(a) Text

Method	AP	P@10%	R@10%
DM	0.963	0.932	0.934
BC_{top3}	0.897	0.863	0.865
CU	0.881	0.837	0.839
BC_{top2}	0.881	0.837	0.839
BC_{top5}	0.716	0.625	0.626
WD	0.665	0.632	0.633
LE	0.567	0.579	0.580
BC_{all}	0.350	0.378	0.379
MD	0.206	0.231	0.232
DU	0.103	0.104	0.104
PM	0.102	0.104	0.104
KNN	0.100	0.101	0.101

(b) Span

embeddings. For sentence embeddings we will use *all-mpnet-base-v2* (see § 1.2). To compute the KNN entropy, we use the code of the original authors.

Borda Count In order to evaluate which scores to aggregate via *Borda Count* we evaluate three settings on our AG NEWS tuning data. We either aggregate the five best, three best, or all scorer outputs. As the underlying model for model-based methods we use transformers, because we need repeated probabilities for *Dropout Uncertainty*. The results are listed in Table A.3. It can be seen that aggregating only the three best scores leads to far superior performance. Hence, we choose this setting when evaluating *Borda Count* aggregation during our experiments.

Appendix B. Calibration

From the most common calibration methods we select the best method by calibrating probabilities for models trained on our AG NEWS tuning data. We use 10-fold cross-validation where eight parts are used for training models, one for calibrating and one for evaluating the calibration. The results can be seen in Figure B.1. We follow Guo et al.

(2017) and use the Expected Calibration Error (ECE) (Naeini, Cooper, and Hauskrecht 2015) as the metric for calibration quality. We decide to use one method for all task types and finally choose *Logistic Calibration* (also known as Platt Scaling), which performs well across tasks. We use the implementations of Küppers et al. (2020).

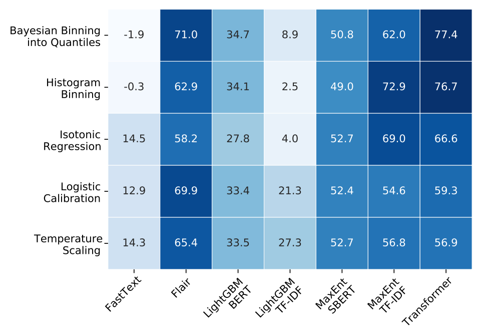


Figure B.1
Percentage decrease of the Expected Calibration Error (ECE) after calibration, when training models on our AG NEWS tuning data. Higher is better.

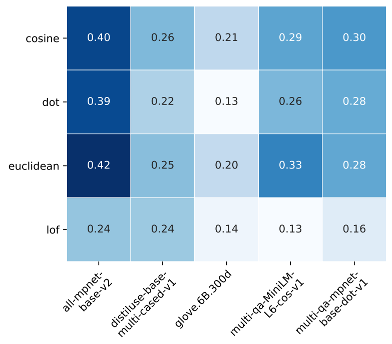
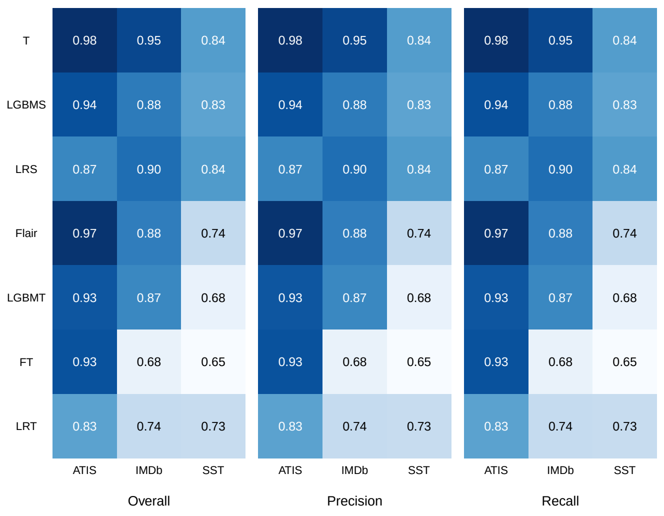
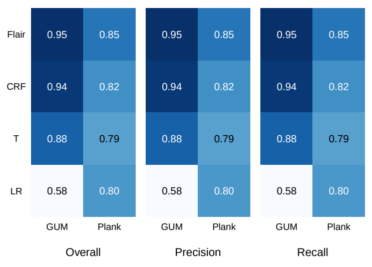


Figure B.2
Average Precision of using *Mean Distance* with different embedders and similarity metrics on our AG NEWS tuning data.

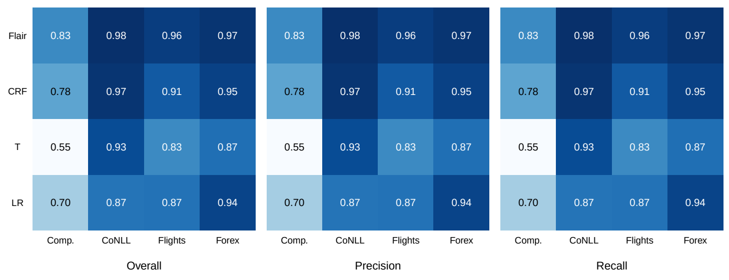
Appendix C. Best Scores



(a) Text classification



(b) Token labeling



(c) Sequence labeling

Figure C.1
Model performances across tasks and datasets. The model axis is ordered in descending order by the respective models' overall performance via Borda Count.

C	M	P	R	F1	%F
ATIS	CL	0.43	0.30	0.35	0.03
	DE	0.56	1.00	0.72	0.09
	IRT	0.00	0.00	0.00	0.91
	LA	0.71	1.00	0.83	0.07
	PE	0.37	1.00	0.54	0.13
	RE	0.67	1.00	0.81	0.07
IMDb	CL	0.23	0.63	0.33	0.06
	DE	0.19	0.73	0.30	0.08
	IRT	0.01	0.30	0.01	0.93
	LA	0.22	0.64	0.33	0.06
	PE	0.10	0.62	0.18	0.12
	RE	0.22	0.64	0.33	0.06
SST	CL	0.22	0.82	0.34	0.19
	DE	0.21	0.82	0.33	0.20
	IRT	0.01	0.16	0.02	0.82
	LA	0.22	0.84	0.35	0.19
	PE	0.22	0.84	0.34	0.19
	RE	0.21	0.82	0.34	0.19

C	M	AP	P@10%	R@10%
ATIS	BC	0.98	0.48	1.00
	CS	0.97	0.48	1.00
	CU	0.87	0.48	1.00
	DM	0.98	0.48	1.00
	DU	0.05	0.06	0.13
	KNN	0.13	0.13	0.27
	LS	0.91	0.48	1.00
	MD	0.14	0.17	0.35
	PM	0.06	0.06	0.13
IMDb	BC	0.35	0.14	0.71
	CS	0.29	0.13	0.64
	CU	0.28	0.15	0.74
	DM	0.25	0.13	0.63
	DU	0.06	0.08	0.39
	KNN	0.05	0.05	0.27
	LS	0.31	0.13	0.67
	MD	0.03	0.04	0.18
	PM	0.05	0.07	0.35
SST	BC	0.50	0.37	0.74
	CS	0.21	0.27	0.54
	CU	0.27	0.29	0.59
	DM	0.49	0.33	0.67
	DU	0.05	0.04	0.09
	KNN	0.11	0.11	0.22
	LS	0.46	0.33	0.65
	MD	0.08	0.10	0.20
	PM	0.05	0.05	0.10

(a) Text classification

C	M	P	R	F1	%F
Comp.	CL	0.83	0.35	0.50	0.18
	DE	0.57	0.58	0.57	0.43
	IRT	0.32	0.59	0.41	0.79
	LA	0.78	0.48	0.59	0.26
	PE	0.59	0.53	0.56	0.38
	RE	0.80	0.53	0.64	0.28
	VN	0.69	0.06	0.11	0.04
CoNLL	CL	0.39	0.18	0.24	0.02
	DE	0.26	0.30	0.28	0.06
	IRT	0.27	0.31	0.29	0.06
	LA	0.29	0.31	0.30	0.06
	PE	0.17	0.47	0.25	0.15
	RE	0.30	0.33	0.32	0.06
	VN	0.60	0.01	0.02	0.00
Flights	CL	0.92	0.27	0.42	0.01
	DE	0.41	0.80	0.55	0.07
	IRT	0.01	0.20	0.02	0.93
	LA	0.60	0.73	0.66	0.05
	PE	0.18	0.68	0.29	0.14
	RE	0.61	0.73	0.67	0.05
	VN	1.00	0.17	0.29	0.01
Forex	CL	0.73	0.46	0.57	0.06
	DE	0.52	0.81	0.64	0.16
	IRT	0.49	0.85	0.62	0.18
	LA	0.65	0.76	0.70	0.12
	PE	0.45	0.72	0.56	0.16
	RE	0.67	0.75	0.70	0.11
	VN	1.00	0.07	0.14	0.01

C	M	AP	P@10%	R@10%
Comp.	BC	0.68	0.83	0.20
	CU	0.70	0.87	0.21
	DM	0.66	0.82	0.19
	DU	0.43	0.50	0.12
	KNN	0.61	0.75	0.18
	LE	0.41	0.38	0.09
	MD	0.54	0.59	0.14
	PM	0.54	0.64	0.15
	WD	0.45	0.48	0.11
CoNLL	BC	0.14	0.13	0.24
	CU	0.17	0.18	0.34
	DM	0.14	0.12	0.23
	DU	0.07	0.08	0.15
	KNN	0.12	0.13	0.24
	LE	0.19	0.17	0.32
	MD	0.06	0.05	0.09
	PM	0.06	0.07	0.14
	WD	0.16	0.17	0.32
Flights	BC	0.49	0.24	0.63
	CU	0.68	0.29	0.76
	DM	0.35	0.25	0.66
	DU	0.18	0.10	0.27
	KNN	0.07	0.07	0.20
	LE	0.10	0.10	0.27
	MD	0.07	0.11	0.29
	PM	0.12	0.12	0.32
	WD	0.11	0.12	0.32
Forex	BC	0.54	0.49	0.49
	CU	0.70	0.71	0.71
	DM	0.61	0.66	0.65
	DU	0.32	0.32	0.32
	KNN	0.16	0.14	0.14
	LE	0.11	0.09	0.09
	MD	0.14	0.20	0.20
	PM	0.25	0.30	0.30
	WD	0.14	0.20	0.20

(b) Span labeling

C	M	P	R	F1	%F
GUM	CL	0.73	0.90	0.80	0.06
	DE	0.59	1.00	0.74	0.08
	IRT	0.00	0.00	0.00	0.91
	LA	0.51	1.00	0.68	0.10
	PE	0.41	1.00	0.58	0.12
	RE	0.53	1.00	0.69	0.09
	VN	0.47	0.66	0.55	0.07
Plank	CL	0.47	0.31	0.37	0.08
	DE	0.45	0.51	0.48	0.13
	IRT	0.07	0.47	0.12	0.84
	LA	0.46	0.53	0.49	0.14
	PE	0.48	0.53	0.50	0.13
	RE	0.47	0.52	0.49	0.13
	VN	0.55	0.21	0.30	0.04

C	M	AP	P@10%	R@10%
GUM	BC	0.92	0.47	0.95
	CU	0.98	0.50	1.00
	DM	0.95	0.49	0.98
	DU	0.05	0.05	0.10
	KNN	0.21	0.19	0.38
	LE	0.60	0.34	0.69
	MD	0.12	0.14	0.29
	PM	0.05	0.05	0.11
	WD	0.53	0.39	0.79
Plank	BC	0.38	0.42	0.36
	CU	0.42	0.51	0.43
	DM	0.27	0.37	0.31
	DU	0.24	0.28	0.24
	KNN	0.31	0.39	0.33
	LE	0.22	0.24	0.21
	MD	0.16	0.19	0.16
	PM	0.23	0.29	0.24
	WD	0.39	0.43	0.37

(c) Token labeling

Figure C.2
AED results achieved with using the respective best models across all flaggers ● and scorers ● for text classification, span, and token labeling.

Acknowledgments
We thank Andreas Rücklé, Edwin Simpson, Falko Helm, Ivan Habernal, Ji-Ung Lee, Michael Bugert, Nafise Sadat Moosavi, Nils Dycke, Richard Eckart de Castilho, Tobias Mayer, and Yevgeniy Puzikov, and our anonymous reviewers for the fruitful discussions and helpful feedback that improved this article. We are especially grateful for Michael Bugert and his implementation of the span matching via linear sum assignment and Amir Zeldes, Andreas Grivas, Beatrice Alex, Hadi Amiri, Jeremy Barnes, as well as Stefan Larson for answering our questions regarding their publications and for making datasets and code available. This research work has been funded by the "Data Analytics for the Humanities" grant by the Hessian Ministry of Higher Education, Research, Science and the Arts.

References

Ahrens, Sönke. 2017. *How to Take Smart Notes: One Simple Technique to Boost Writing, Learning and Thinking: For Students, Academics and Nonfiction Book Writers*. CreateSpace, North Charleston, SC.

Akbik, Alan, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 54–59.

Alt, Christoph, Aleksandra Gabryszak, and Leonhard Hennig. 2020. TACRED revisited: A thorough evaluation of the TACRED relation extraction task. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1558–1569. https://doi.org/10.18653/v1/2020.acl-main.142

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

Amiri, Hadi, Timothy Miller, and Guergana Savova. 2018. Spotting spurious data with neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2006–2016. https://doi.org/10.18653/v1/N18-1182

Angle, Sachi, Pruthwik Mishra, and Dipti Mishra Sharma. 2018. Automated error correction and validation for POS tagging of Hindi. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, pages 11–18.

Aroyo, Lora and Chris Welty. 2015. Truth is a lie: Crowd truth and the seven myths of human annotation. *AI Magazine*, 36(1):15–24. https://doi.org/10.1609/aimag.v36i1.2564

Barnes, Jeremy, Lilja Øvrelid, and Erik Velldal. 2019. Sentiment analysis is not solved! Assessing and probing sentiment classification. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 12–23. https://doi.org/10.18653/v1/W19-4802

Basile, Valerio, Michael Fell, Tommaso Fornaciari, Dirk Hovy, Silviu Paun, Barbara Plank, Massimo Poesio, and Alexandra Uma. 2021. We need to consider disagreement in evaluation. In *Proceedings of the 1st Workshop on Benchmarking: Past, Present and Future*, pages 15–21. https://doi.org/10.18653/v1/2021.bppf-1.3

Behrens, Heike, editor. 2008. *Corpora in Language Acquisition Research: History, Methods, Perspectives*, volume 6 of *Trends in Language Acquisition Research*. John Benjamins Publishing Company, Amsterdam. https://doi.org/10.1075/tilar.6.03beh

Boyd, Adriane, Markus Dickinson, and W. Detmar Meurers. 2008. On detecting errors in dependency treebanks. *Research on Language and Computation*, 6(2):113–137. https://doi.org/10.1007/s11168-008-9051-9

Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying density-based local outliers. *ACM SIGMOD Record*, 29(2):93–104. https://doi.org/10.1145/335191.335388

Burkard, Rainer, Mauro Dell'Amico, and Silvano Martello. 2012. *Assignment Problems: Revised Reprint*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611972238

Cer, Daniel, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in*

Natural Language Processing: System Demonstrations, pages 169–174. https://doi.org/10.18653/v1/D18-2029

Chinchor, Nancy and Beth Sundheim. 1993. MUC-5 evaluation metrics. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, pages 69–78. https://doi.org/10.3115/1072017.1072026

Cui Zhu, H. Kitagawa, and C. Faloutsos. 2005. Example-based robust outlier detection in high dimensional datasets. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 829–832.

Davis, Jesse and Mark Goadrich. 2006. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning - ICML '06*, pages 233–240.

Dawid, A. P. and A. M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28. https://doi.org/10.2307/2346806

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Dickinson, Markus. 2006. From detecting errors to automatically correcting them. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 265–272.

Dickinson, Markus. 2015. Detection of annotation errors in Corpora. *Language and Linguistics Compass*, 9(3):119–138. https://doi.org/10.1111/lnc3.12129

Dickinson, Markus and Chong Min Lee. 2008. Detecting errors in semantic annotation. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 605–610.

Dickinson, Markus and W. Detmar Meurers. 2003a. Detecting errors in part-of-speech annotation. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 107–114. https://doi.org/10.3115/1067807.1067823

Dickinson, Markus and W. Detmar Meurers. 2003b. Detecting inconsistencies in treebanks. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, pages 1–12.

Dickinson, Markus and W. Detmar Meurers. 2005. Detecting errors in discontinuous structural annotation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics - ACL '05*, pages 322–329. https://doi.org/10.3115/1219840.1219880

Dligach, Dmitriy and Martha Palmer. 2011. Reducing the need for double annotation. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 65–73.

Dwork, Cynthia, Ravi Kumar, Moni Naor, and D. Sivakumar. 2001. Rank aggregation methods for the Web. In *Proceedings of the Tenth International Conference on World Wide Web - WWW '01*, pages 613–622.

Fornaciari, Tommaso, Alexandra Uma, Silviu Paun, Barbara Plank, Dirk Hovy, and Massimo Poesio. 2021. Beyond black & white: Leveraging annotator disagreement via soft-label multi-task learning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2591–2597. https://doi.org/10.18653/v1/2021.naacl-main.204

Gal, Yarin and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1050–1059.

Gimpel, Kevin, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47. https://doi.org/10.21236/ADA547371

Grivas, Andreas, Beatrice Alex, Claire Grover, Richard Tobin, and William Whiteley. 2020. Not a cute stroke: Analysis of Rule- and Neural Network-based Information Extraction Systems for Brain Radiology Reports. In *Proceedings of the 11th International Workshop on Health Text Mining and Information Analysis*, pages 24–37. https://doi.org/10.18653/v1/2020.louhi-1.4

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of* *the 34th International Conference on Machine Learning*, pages 1321–1330.

Gururangan, Suchin, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360. https://doi.org/10.18653/v1/2020.acl-main.740

Haselbach, Boris, Kerstin Eckart, Wolfgang Seeker, Kurt Eberle, and Ulrich Heid. 2012. Approximating theoretical linguistics classification in real data: The case of German "nach" particle verbs. In *Proceedings of COLING 2012*, pages 1113–1128.

Hedderich, Michael A., Dawei Zhu, and Dietrich Klakow. 2021. Analysing the noise model error for realistic noisy label data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7675–7684. https://doi.org/10.1609/aaai.v35i9.16938

Heinzerling, Benjamin and Michael Strube. 2018. BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 2989–2993.

Hemphill, Charles T., John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the Workshop on Speech and Natural Language*, pages 96–101. https://doi.org/10.3115/116580.116613

Hendrycks, Dan and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of International Conference on Learning Representations*, pages 1–12.

Hollenstein, Nora, Nathan Schneider, and Bonnie Webber. 2016. Inconsistency detection in semantic annotation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3986–3990.

Hovy, Dirk, 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*, pages 1120–1130.

Jamison, Emily and Iryna Gurevych. 2015. Noise or additional information? Leveraging crowdsource annotation item agreement for natural language tasks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 291–297. https://doi.org/10.18653/v1/D15-1035

Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. https://doi.org/10.18653/v1/E17-2068

Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1–9.

Kehler, A., L. Kertz, H. Rohde, and J. L. Elman. 2007. Coherence and coreference revisited. *Journal of Semantics*, 25(1):1–44. https://doi.org/10.1093/jos/ffm018, PubMed: 22923856

Kendall, M. G. 1938. A new measure of rank correlation. *Biometrika*, 30(1–2):81–93. https://doi.org/10.1093/biomet/30.1-2.81

Khandelwal, Urvashi, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations (ICLR)*, pages 1–13.

Küppers, Fabian, Jan Kronenberger, Amirhossein Shantia, and Anselm Haselhoff. 2020. Multivariate confidence calibration for object detection. In *2nd Workshop on Safe Artificial Intelligence for Automated Driving (SAIAD)*, pages 1–9.

Květoň, Pavel and Karel Oliva. 2002. (Semi-)automatic detection of errors in PoS-tagged corpora. In *COLING 2002: The 19th International Conference on Computational Linguistics*, pages 1–7. https://doi.org/10.3115/1072228.1072249

Larson, Stefan, Adrian Cheung, Anish Mahendran, Kevin Leach, and Jonathan K. Kummerfeld. 2020. Inconsistencies in crowdsourced slot-filling annotations: A typology and identification methods. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5035–5046. https://doi.org/10.18653/v1/2020.coling-main.442

Larson, Stefan, Anish Mahendran, Andrew Lee, Jonathan K. Kummerfeld, Parker Hill, Michael A. Laurenzano, Johann Hauswald, Lingjia Tang, and Jason Mars. 2019. Outlier detection for improved data quality and diversity in dialog systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 517–527. https://doi.org/10.18653/v1/N19-1051

Leitner, Sebastian. 1974. *So Lernt Man Leben [How to Learn to Live]*. Droemer-Knaur, Munich.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arxiv preprints 11692.

Loftsson, Hrafn. 2009. Correcting a POS-Tagged corpus using three complementary methods. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 523–531. https://doi.org/10.3115/1609067.1609125

Lord, F. M., M. R. Novick, and Allan Birnbaum. 1968. *Statistical Theories of Mental Test Scores*. Addison-Wesley, Oxford, England.

Manning, Christopher D. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, volume 6608, pages 171–189. https://doi.org/10.1007/978-3-642-19400-9_14

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York.

Ménard, Pierre André and Antoine Mougeot. 2019. Turning silver into gold: Error-focused corpus reannotation with active learning. In *Proceedings - Natural Language Processing in a Deep Learning World*, pages 758–767. https://doi.org/10.26615/978-954-452-056-4_088

Naeini, Mahdi Pakdaman, Gregory F. Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2901–2907.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043.

Northcutt, Curtis, Lu Jiang, and Isaac Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411. https://doi.org/10.1613/jair.1.12125

Northcutt, Curtis G., Anish Athalye, and Jonas Mueller. 2021. Pervasive label errors in test sets destabilize machine learning benchmarks. In *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, pages 1–13.

Paun, Silviu, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing Bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6(0):571–585. https://doi.org/10.1162/tacl_a_00040

Pavlick, Ellie and Tom Kwiatkowski. 2019. Inherent disagreements in human textual inferences. *Transactions of the Association for Computational Linguistics*, 7:677–694.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. https://doi.org/10.3115/v1/D14-1162

Peters, Matthew E., Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? Adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14. https://doi.org/10.18653/v1/W19-4302

Plank, Barbara, Dirk Hovy, and Anders Søgaard. 2014a. Learning part-of-speech taggers with inter-annotator agreement loss. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 742–751.

Plank, Barbara, Dirk Hovy, and Anders Søgaard. 2014b. Linguistically debatable or just plain wrong? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 507–511. https://doi.org/10.3115/v1/P14-2083

Platt, John C. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods.

Advances in Large Margin Classifiers, 10(3):1–9.

Pustejovsky, J. and Amber Stubbs. 2013. *Natural Language Annotation for Machine Learning*. O'Reilly Media, Sebastopol, CA.

Qian, Kun, Ahmad Beirami, Zhouhan Lin, Ankita De, Alborz Geramifard, Zhou Yu, and Chinnadhurai Sankar. 2021. Annotation inconsistency and entity bias in MultiWOZ. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 326–337.

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

Reimers, Nils and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3980–3990. https://doi.org/10.18653/v1/D19-1410

Reiss, Frederick, Hong Xu, Bryan Cutler, Karthik Muthuraman, and Zachary Eichenberger. 2020. Identifying incorrect labels in the CoNLL-2003 corpus. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 215–226. https://doi.org/10.18653/v1/2020.conll-1.16

Rodrigues, Filipe and Francisco Pereira. 2018. Deep learning from crowds. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1611–1618.

Rodriguez, Pedro, Joe Barrow, Alexander Miserlis Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. 2021. Evaluation examples are not equally informative: How should that change NLP leaderboards? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4486–4503. https://doi.org/10.18653/v1/2021.acl-long.346

Saito, Takaya and Marc Rehmsmeier. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3):1–21. https://doi.org/10.1371/journal.pone.0118432, PubMed: 25738806

Sanguinetti, Manuela and Cristina Bosco. 2015. PartTUT: The Turin University Parallel Treebank. In Basili, Roberto, Cristina Bosco, Rodolfo Delmonte, Alessandro Moschitti, and Maria Simi, editors, *Harmonization and Development of Resources and Tools for Italian Natural Language Processing within the PARLI Project*, volume 589, pages 51–69. https://doi.org/10.1007/978-3-319-14206-7_3

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. In *Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing*, pages 1–5.

Schreibman, Susan, Ray Siemens, and John Unsworth, editors. 2004. *A Companion to Digital Humanities*. Blackwell Publishing Ltd, Malden, MA, USA.

Shelmanov, Artem, Evgenii Tsymbalov, Dmitri Puzyrev, Kirill Fedyanin, Alexander Panchenko, and Maxim Panov. 2021. How certain is your transformer? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1833–1840. https://doi.org/10.18653/v1/2021.eacl-main.157

Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Song, Hwanjun, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2020. Learning from noisy labels with deep neural networks: A survey. arxiv preprint, 2007.8199. https://doi.org/10.1109/TNNLS.2022.3152527, PubMed: 35254993

Stoica, George, Emmanouil Antonios Platanios, and Barnabas Poczos. 2021. Re-TACRED: Addressing shortcomings of the TACRED dataset. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence 2021*, pages 13843–13850. https://doi.org/10.1609/aaai.v35i15.17631

Sun, Chen, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–13.

Swayamdipta, Swabha, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and

Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293. https://doi.org/10.18653/v1/2020.emnlp-main.746

Szpiro, George. 2010. *Numbers Rule: The Vexing Mathematics of Democracy, from Plato to the Present*. Princeton University Press. https://doi.org/10.1515/9781400834440

Tjong Kim Sang, Erik F. and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147. https://doi.org/10.3115/1119176.1119195

Ustalov, Dmitry, Nikita Pavlichenko, Vladimir Losev, Iulian Giliazev, and Evgeny Tulin. 2021. A general-purpose crowdsourcing computational quality control toolkit for Python. In the *Ninth AAAI Conference on Human Computation and Crowdsourcing: Works-in-Progress and Demonstration Track*, pages 1–4.

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

Vlachos, Andreas. 2006. Active annotation. In *Proceedings of the Workshop on Adaptive Text Extraction and Mining (ATEM 2006)*, pages 64–71.

Wang, Zihan, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019. CrossWeigh: Training named entity tagger from imperfect annotations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5153–5162. https://doi.org/10.18653/v1/D19-1519, PubMed: 31303768

Wilcoxon, Frank. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80. https://doi.org/10.2307/3001968

Wisniewski, Guillaume. 2018. Errator: A tool to help detect annotation errors in the Universal Dependencies project. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 4489–4493.

Yaghoub-Zadeh-Fard, Mohammad Ali, Boualem Benatallah, Moshe Chai Barukh, and Shayan Zamanirad. 2019. A study of incorrect paraphrases in crowdsourced user utterances. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 295–306. https://doi.org/10.18653/v1/N19-1026

Zadrozny, Bianca and Charles Elkan. 2001. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 609–616.

Zadrozny, Bianca and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 694–700.

Zeldes, Amir. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612. https://doi.org/10.1007/s10579-016-9343-x

Zhang, Xiang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, pages 649–657.

Zhang, Xin, Guangwei Xu, Yueheng Sun, Meishan Zhang, and Pengjun Xie. 2021. Crowdsourcing learning as domain adaptation: A case study on named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5558–5570. https://doi.org/10.18653/v1/2021.acl-long.432

Zheng, Guoqing, Ahmed Hassan Awadallah, and Susan Dumais. 2021. Meta label correction for noisy label learning. In *Proceedings of the Thirty-fifth AAAI Conference on Artificial Intelligence 2021*, pages 11053–11061. https://doi.org/10.1609/aaai.v35i12.17319