Random Splitting Negatively Impacts NER Evaluation: Quantifying and Eliminating the Overestimation of NER Performance

Florian Babl[1], **Moritz Hennen**[1,2], **Jakob Murauer**[1], **Michaela Geierhos**[1]

[1]University of the Bundeswehr Munich, Germany
[2]Ludwig Maximilian University of Munich, Germany
Correspondence: florian.babl@unibw.de

Abstract

In named entity recognition (NER), models are evaluated on their ability to identify entity mentions in text. However, standard evaluation methods often rely on test sets that contain named entities already present in the training data, raising concerns about overestimation of model performance. This work investigates the impact of varying degrees of entity contamination on a dataset level on the generalization ability and reported F1 scores of three state-of-the-art NER models. Experiments on five standard benchmarks show that F1 scores for contaminated entities statistically significantly inflate reported F1 scores as contamination rates increase, with F1 performance gaps ranging from 2-10% compared to entities not seen during training. To address these inflated F1 scores, we additionally propose a novel NER dataset splitting method using a minimum cut algorithm to minimize train-test entity leakage. While our splitting method ensures near-zero entity contamination, we also compare new and existing dataset splits on named entity sample counts.

1 Introduction

Named entity recognition (NER) is a critical component of information extraction that involves the identification of named entities (NEs) in unstructured text. While significant progress has been made in NER in recent years, there is still a pressing need for more accurate evaluation of model performance. The existence of *test set contamination* is a key aspect of this evaluation. This occurs when the model is evaluated using documents that contain NEs present during training. This leads to an overestimation of the model's performance, which does not accurately reflect the generalization ability of an NER model.

Despite the availability of many NER datasets, there has been surprisingly little research on the degree of their test set contamination. Most of the existing datasets have been generated using standard random splitting methods (see Section 4). However, we show that this approach is often the source of significant contamination. Therefore, we perform an in-depth analysis of seven widely used NER datasets, namely ACE05 (Li and Ji, 2014), ADE (Gurulingappa et al., 2012), CoNLL03 (Sang and Meulder, 2003), CoNLL04 (Roth and Yih, 2004; Gupta et al., 2016), GENIA (Kim et al., 2003), NYT (Riedel et al., 2010), and SciERC (Luan et al., 2018). We find a worrisome amount of test set contamination in all of them. For example, as shown in Figure 1, NEs such as *Idaho* occur both during training and at test time, encouraging model memorization rather than generalization, making the evaluation a test of a model's ability to memorize.

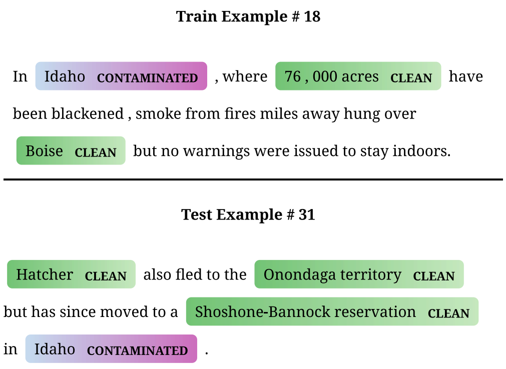
Train Example # 18
In Idaho CONTAMINATED , where 76 , 000 acres CLEAN have been blackened , smoke from fires miles away hung over Boise CLEAN but no warnings were issued to stay indoors.
Test Example # 31
Hatcher CLEAN also fled to the Onondaga territory CLEAN but has since moved to a Shoshone-Bannock reservation CLEAN in Idaho CONTAMINATED .

Figure 1: Example of entity contamination (here: Idaho) in the CoNLL04 (Roth and Yih, 2004) splits by Gupta et al. (2016) between a training and test sample.

We investigate the effect of increasing contamination rates on model generalization, i.e., the ability of the model to generalize to previously unseen NEs, through a series of studies using three fundamentally different state-of-the-art NER architectures. We found that even small rates of contamination statistically significantly inflate reported test F1 scores, thereby overestimating NER performance. To address this issue, we propose a method to quantify the degree of overestimation as a delta of the original F1 score. Furthermore, based on our findings and the lack of existing research and approaches, we identify a need for creating NER dataset splits that focus on creating a test set that fairly evaluates a model's ability to generalize. We also argue that ideally, a test set should contain as many unseen NEs as possible to fairly test the model's generalization ability (Søgaard et al., 2021). In order to test the generalization performance of NER models separately from their memorization ability, we publish two subsets of *contaminated* and *clean* entities for the test sets of all datasets. These allow the calculation of the ΔF1 for each dataset. We also provide and demonstrate a method for generating clean training and test splits on all datasets using a *minimum cut algorithm* (Karypis and Kumar, 1998). In brief, our main contributions are as follows:

1. We provide an overview of the contamination rate for the widely used datasets ADE, ACE05, CoNLL03, CoNLL04, GENIA, NYT, and SciERC.

2. We perform a detailed analysis to test how NE contamination rates statistically affect the ability of an ensemble of models to generalize to unseen NEs, and find how this leads to an overestimation of NER performance.

3. Following the call for better evaluation by Søgaard et al. (2021) and Gorman and Bedrick (2019), we derive a measure to quantify the degree of overestimation of a model's F1 score.

4. We use a minimum cut algorithm (Karypis and Kumar, 1998) to create NER training, development, and test sets that contain a minimal amount of NE contamination between them.

The dataset splits and code for reproducing experiments, analyzing contamination of other NER datasets, and the algorithm for creating clean NER dataset splits are available on GitHub.

2 Related Work

The difficulty of NER models to generalize to unseen NEs has been studied and identified previously (Augenstein et al., 2017; Bernier-Colborne and Langlais, 2020; Lin et al., 2021; Tu and Lignos, 2021). In particular, Augenstein et al. (2017) focus on evaluating NER performance on unseen samples, and find lower F1 and recall scores in various genres, and a tendency to overfit to previously seen tokens. Together with Bernier-Colborne and Langlais (2020) and Lin et al. (2021) they report percentages of unseen NEs on several datasets. However, they all fail to analyze how different rates of seen NEs for a given dataset statistically affect the F1 score on previously seen and unseen NEs, especially on current transformer and state-of-the-art systems. Agarwal et al. (2021) analyze transformers but find no improvement in BERT (Devlin et al., 2019)'s generalization ability to unseen NEs compared to GloVe-based (Pennington et al., 2014) approaches, while ELMo (Peters et al., 2018) and FLAIR (Akbik et al., 2019) are better at generalizing to different domains and contamination rates – referred to in Fu et al. (2020) as entity coverage ratio. They also find that whether the context of the test entity has been seen before affects the generalization ability of NER models, not just the previously seen surface shapes of NEs.
Most existing approaches to generalization testing focus on extending the training or test data to create unseen NEs (Morris et al., 2020; Ribeiro et al., 2020; Lin et al., 2021), while others test on different domains (Augenstein et al., 2017; Agarwal et al., 2020; Fu et al., 2020; Bernier-Colborne and Langlais, 2020). For example, Lin et al. (2021) used the semantic classes in Wikidata (Vrandecic and Krötzsch, 2014) and BERT to replace target entities for a given input to create unseen samples and provide statistics on entities seen during training for the OntoNotes (Weischedel et al., 2013) dataset. While these approaches are easy to use, they require additional steps and do not address the root cause of contamination.

A common approach to splitting a newly created NER dataset into training, validation, and test sets is to randomly split it in an 80-10-10 or similar ratio (see Table 1). While there is research evaluating standard splits versus multiple random splits (Gorman and Bedrick, 2019) and multiple domains of test sets (Søgaard et al., 2021; Vajjala and Balasubramaniam, 2022), to the best of our knowledge, there is no research on creating NER test sets that do not share NEs with the training set – except for augmentation techniques.

Creating a method for generating clean training and test sets would solve the aforementioned overfitting and scoring on the same NEs problems. This method would ensure fair evaluation and a better approximation of the model's real-world performance (see Section 5.1 and Augenstein et al. (2017)). Most importantly, this method would allow for a better partitioning of future NER datasets.

3 Methodology

In the following, we first lay the theoretical groundwork for our analysis and define the terms *contamination* and *clean* (Section 3.1), introduce a new measure to quantify the overestimation of NER model F1 scores (Section 3.2), and frame dataset contamination as a minimum cut problem in order to reduce the contamination of NER dataset splits (Section 3.3).

3.1 Entity and Sample Contamination

Given training and test sets $\mathcal{D}_{train}, \mathcal{D}_{test}$ with $\Sigma = \{train, test\}$ consisting of samples $(\mathbf{x}_i, E_i)$ of documents $\mathbf{x}_i$ and NEs $E_i = \{(\mathbf{x}_{i,n\ldots m}, t)\}$ of type $t \in T$ and case-sensitive words[1] $\mathbf{x}_{i,n} \ldots \mathbf{x}_{i,m}$ $(n \leq m)$, we define the set of entities in split $split$ as the union of all entities present during training and testing, respectively:

$$\mathcal{E}_{split} = \bigcup_{(\mathbf{x}_i, E_i) \in \mathcal{D}_{split}} E_i \quad (split \in \Sigma) \tag{1}$$

Additionally, we define the opposite split for a given split $split$ as $\omega(split)$:

$$\omega(train) = test \qquad \omega(test) = train \tag{2}$$

We define an NE $e_{ij} \in E_i$ of sample $(\mathbf{x}_i, E_i)$ as ***contaminated*** if it is a member of the NEs of the opposite split $\omega(split)$:

$$e_{ij} \textit{ contaminated} \Leftrightarrow e_{ij} \in E_i \wedge e_{ij} \in \mathcal{E}_{\omega(split)}$$

Similarly, we define a sample $(\mathbf{x}_i, E_i)$ as ***partially contaminated*** if and only if

$$\left| E_i \cap \mathcal{E}_{\omega(split)} \right| > 0 \tag{3}$$

holds, i.e., at least one entity in a given sample is a member of the entity set of the opposite split $\mathcal{E}_{\omega(split)}$. Furthermore, we define a sample as ***fully contaminated*** if and only if

$$\left| E_i \cap \mathcal{E}_{\omega(split)} \right| \equiv |E_i| \tag{4}$$

holds, i.e., every entity in a given sample is contained in the opposite split $\mathcal{E}_{\omega(split)}$.

Finally, we define a function $\mathbf{c}(split)$ as the subset of samples in the dataset split $split$ that are *partially contaminated*:

$$\begin{aligned} \mathbf{c}(\mathcal{D}_{split}) = \big\{ (\mathbf{x}_i, E_i) \mid E_i \cap \mathcal{E}_{\omega(split)} \neq \emptyset \\ \wedge (\mathbf{x}_i, E_i) \in \mathcal{D}_{split} \big\} \end{aligned} \tag{5}$$

Inversely, we define

$$\neg\mathbf{c}(\mathcal{D}_{split}) = \mathcal{D}_{split} \setminus \mathbf{c}(\mathcal{D}_{split}) \tag{6}$$

as the subset of not contaminated, ***clean*** samples in split $split$.

3.2 Quantifying the Overestimation of NER Performance due to Entity Contamination

To quantify the degree to which a model's NER performance is overestimated, we propose a simple formula

$$F_{1(clean)} = \frac{2 * P_{(orig.)} * R_{(clean\,test)}}{P_{(orig.)} + R_{(clean\,test)}} \tag{7}$$

$$\Delta F_1 = F_{1(orig.)} - F_{1(clean)} \tag{8}$$

where the precision $P_{(orig.)}$ is measured with respect to the original test set, providing an accurate measure of a model's ability to capture all NEs ***relevant*** for the NER task. Since we are interested in quantifying the ability of the NER model to generalize, i.e., to identify named entities at test time that are not contaminated – named entities that were not seen during training – the set of relevant named entities in this context $E_i^{(clean)} = E_i \setminus \mathcal{E}_{train}$ is only a subset of the set of all named entities of any given sample $(\mathbf{x}_i, E_i) \in \mathcal{D}_{test}$. Note that the number of test samples remains unchanged.

Measuring Generalization Recall. We can directly use the set of non-contaminated entities $E_i^{(clean)}$ to calculate the non-contaminated recall performance of a model, since this metric quantifies how many of the ***non-contaminated*** named entities a model was able to identify at test time. A visualization of the intuition for R_{clean} is shown in Figure 2. Given a set of named entities O_i identified by an NER model, the recall is thus defined as follows:

$$\begin{aligned} & R_{(clean)} \Leftrightarrow \\ & \Leftrightarrow \frac{|E_i^{(clean)} \cap O_i|}{|E_i^{(clean)} \cap O_i| + |E_i^{(clean)} \setminus O_i|} \\ & \Leftrightarrow \frac{|E_i^{(clean)} \cap O_i|}{|E_i^{(clean)}|}. \end{aligned} \tag{9}$$

[1] Although this definition relies on words, a dataset built using the character- or token-level boundaries of NEs is applicable in the same way.

Approximating Generalization Precision. As shown in Figure 2, calculating the precision of an NER model with respect to just the non-contaminated entities will cause contaminated named entities (◡) being counted as false positives (Tu and Lignos, 2021), resulting in $P_{(clean)} + P_{contam.} = P_{(orig.)}$, as shown in Equation 11.

There are two ways to approach the proper measurement of a model's precision with respect to only the non-contaminated entities:

1. Over-approximation: $P_{(clean)} \approx P_{(orig.)}$ This approach makes it easy for future researchers to adopt our method, since the only requirement for computing $F1_{(clean)}$ becomes computing R_{clean}, which is trivially possible using the exact same test set, but with only non-contaminated NEs as targets[2], as shown in Figure 2 and Equation 9. In practice, this amounts to running the test inference twice, once with the filtered test set (where only non-contaminated named entities are targets), obtaining the original precision and the clean recall from the original and filtered runs, respectively, and finally computing ΔF1.

2. Computing $P_{(clean)^*}$ as

$$P_{(clean)^*} = \frac{\text{◠}}{\text{◠} + \text{☾}} = \frac{\text{◠}}{\text{◠☾}}$$
$$= \frac{|O_i \cap E_i{}^{(clean)}|}{|O_i \cap E_i{}^{(clean)}| + |O_i \setminus E_i|} \tag{10}$$
$$= \frac{|O_i \cap E_i{}^{(clean)}|}{|O_i \setminus E_i{}^{(contam.)}|}$$

However, this approach has a major limitation: It creates significant overhead for future researchers because the output of a model must first be filtered to include only non-contaminated NEs ($O_i \setminus E_i{}^{(contam.)}$) before calculating true and false positives for any given sample.

To further clarify this issue, assume an NER model for persons outputs three named entities, $O_i = \{\text{Alice}, \text{Bob}, \text{Charlie}\}$, for the input $\mathbf{x} =$ *"Alice and Bob meet at Checkpoint Charlie"* and targets $E_i = \{\text{Alice}, \text{Bob}\}$, where Alice was already present during training, i.e. is contaminated.

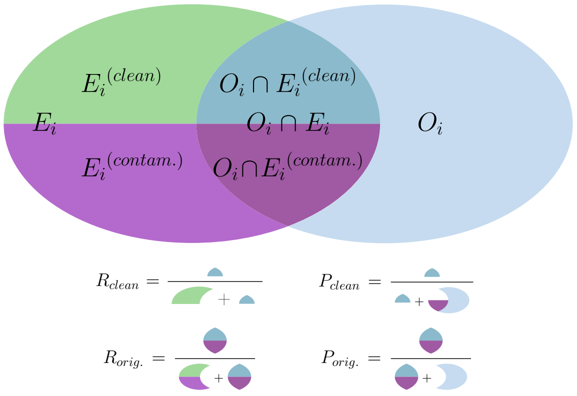

Figure 2: To illustrate the problem of using the naive $P_{(clean)}$ to determine the precision of an NER model on non-contaminated NEs: True positives that are contaminated (◡) would count as false positives when calculating $P_{(clean)}$. O_i is a set of named entities predicted by an NER model, E_i models the target entities for sample i.

$$\begin{aligned}
&P_{(orig.)}\\
\Leftrightarrow\ &\frac{|O_i \cap E_i|}{|O_i \cap E_i| + |O_i \setminus E_i|}\\
\Leftrightarrow\ &\frac{\left|O_i \cap \left(E_i{}^{(clean)} \cup E_i{}^{(contam.)}\right)\right|}{|O_i|}\\
\Leftrightarrow\ &\frac{\left|O_i \cap E_i{}^{(clean)}\right| + \left|O_i \cap E_i{}^{(contam.)}\right|}{|O_i|}\\
\Leftrightarrow\ &\frac{\left|O_i \cap E_i{}^{(clean)}\right|}{|O_i|} + \frac{\left|O_i \cap E_i{}^{(contam.)}\right|}{|O_i|} \qquad (11)\\
\Leftrightarrow\ &\frac{\left|O_i \cap E_i{}^{(clean)}\right|}{\left|O_i \cap E_i{}^{(clean)}\right| + \left|O_i \setminus E_i{}^{(clean)}\right|}\\
+\ &\frac{\left|O_i \cap E_i{}^{(contam.)}\right|}{\left|O_i \cap E_i{}^{(contam.)}\right| + \left|O_i \setminus E_i{}^{(contam.)}\right|}\\
=\ &P_{(clean)} + P_{(contam.)}
\end{aligned}$$

Therefore the non-contaminated precision $P_{(clean)^*} = \frac{|\{\text{Bob}\}|}{|\{O_i \setminus \text{Alice}\}|} = 0.5$ (following Eq. 10), compared to $P_{(clean)} \approx P_{(orig.)} = \frac{|\{\text{Bob},\text{Alice}|}{|\{O_i\}|} = \frac{2}{3}$. Note that for $P_{(clean)^*}$ Alice gets excluded from the model outputs as it is a contaminated NE. This must be known in advance. If $F1_{(clean)}$ were to use $P_{(clean)^*}$, Section 3 would display an even larger ΔF1, as $P_{(orig.)} \geq P_{(clean)^*}$.

In the end, we decided to approximate $P_{(clean)^*} \approx P_{(orig.)}$, since we still want to quantify how precise a model is at retrieving all relevant named entities, i.e., how many named entities are

[2] These splits are published on GitHub.

actual false positives, and not just ***contaminated*** named entities that are not part of ${E_i}^{(clean)}$.

3.3 Sample Contamination as a Minimum Cut Problem

To the best of our knowledge, we are the first to introduce the minimum cut algorithm (Karypis and Kumar, 1998) as a tool to split an NER dataset into non-contaminated splits. To minimize the contamination between documents, we encode each document in a dataset as a node in a weighted graph, where the weight of the connection between nodes is equal to the number of named entities they share. Using a minimum cut algorithm, we then retrieve three partitions of the original graph with as few edge crossings (i.e., partially contaminated samples) as possible.

Let $\{(x_i, E_i)\}$ denote all documents from the entire dataset. For any two nodes i and j, we assign an edge weighted by the size of the intersection of E_i and E_j, i.e.,

$$A_{ij} = |E_i \cap E_j| \tag{12}$$

Conceptually, two documents from the dataset have a weighted connection equal to the number of entities shared by both documents.

We use METIS (Karypis and Kumar, 1998) to perform a constrained minimum cut, partitioning the entire dataset into three new subsets such that each split maintains the original split ratio (see Table 1). By minimizing the total edge weight crossing the partitions, we obtain a segmentation that preserves strongly connected documents within the same subset, while ensuring that each subset roughly meets the prescribed size requirements for the training, development, and test sets. This approach allows us to create new training, development, and test sets that minimize NE contamination between them.

4 Datasets

While it is known for ACE05 and CoNLL04 that both datasets were created using random splitting (Li and Ji, 2014; Gupta et al., 2016), we assume a similar splitting strategy for the other datasets in our analysis (ADE, CoNLL03, GENIA, NYT, and SciERC), as random splitting is considered standard practice (Gorman and Bedrick, 2019), resulting in similar patterns of contamination across all datasets, as shown in Figure 3. This is further justified by the split ratios for all datasets in Table 1.

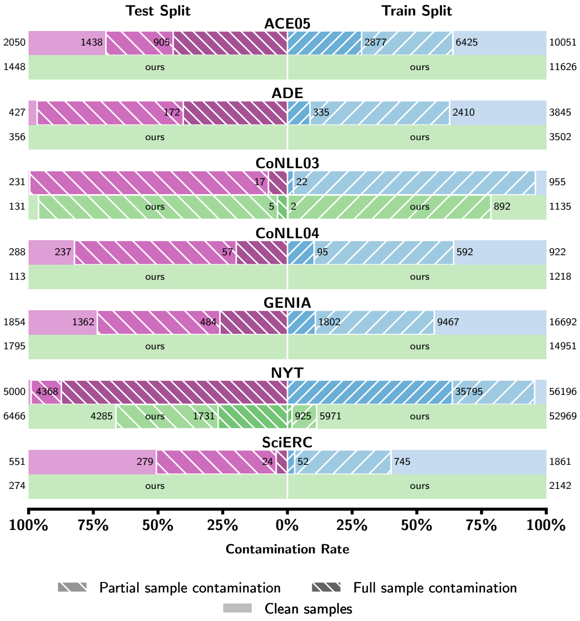

Figure 3: Original training (blue) and test (purple) sample contamination rates for the test and train splits of ACE05, ADE, CoNLL03, CoNLL04, GENIA, NYT, and SciERC (development splits are not included here). With our method (green), we can reduce contamination rates across the board, resulting in non-contaminated datasets for ACE05, ADE, CoNLL04, GENIA, and SciERC, while reducing contamination for CoNLL03 and NYT.

Dataset	**Train** (%)	**Dev** (%)	**Test** (%)
ACE05	69.22	16.66	14.12
ADE	90.01	0.00	9.99
CoNLL03	68.12	15.42	16.48
CoNLL04	63.98	16.03	19.99
GENIA	90.00	0.00	10.00
NYT	84.89	7.55	7.56
SciERC	69.26	10.23	20.51

Table 1: Original training, development, and test split ratios relative to total dataset size.

To quantify the effect of different degrees of contamination on the generalization ability of NER models, we sample new training sets $\mathcal{C}^{\gamma}_{train} \subset \mathcal{D}_{train}$ with increasing partial contamination rates γ from 0 to 100% in 10% increments from the original training sets using Algorithm 1.

The size of these sets is fixed to the minimum of the amount of clean and partially contaminated training samples (Eq. 13), since at most $|\neg\mathbf{c}(\mathcal{D}_{train})|$ samples can be ***clean*** while at most $|\mathbf{c}(\mathcal{D}_{train})|$ samples can be ***partially contaminated***.

Hyperparameter	**ACE05**			**GENIA**			**SciERC**			**NYT**			**CoNLL04**		
	Diffusion	ITER	ASP	Diffusion	ITER	ASP	Diffusion	ITER	ASP	Diffusion	ITER	ASP	Diffusion	ITER	ASP
Epochs	100	25	200	100	25	200	100	25	200	100	25	200	100	25	200
Learning rate (T5)	$2.0e^{-5}$	$3.0e^{-5}$	$5.0e^{-6}$	$3.0e^{-5}$	$3.0e^{-5}$	$5.0e^{-6}$	$2.0e^{-5}$	$3.0e^{-5}$	$5.0e^{-6}$	$2.0e^{-5}$	$3.0e^{-5}$	$5.0e^{-6}$	$2.0e^{-5}$	$3.0e^{-5}$	$5.0e^{-6}$
Learning rate (Task)	-	$3.0e^{-4}$	$1.0e^{-5}$	-	$3.0e^{-4}$	$1.0e^{-5}$	-	$3.0e^{-4}$	$1.0e^{-5}$	-	$3.0e^{-4}$	$1.0e^{-5}$	-	$3.0e^{-4}$	$1.0e^{-5}$
Weight decay	0.01	0.08	0.1	0.01	0.08	0.1	0.01	0.08	0.1	0.01	0.08	0.1	0.01	0.08	0.1
Warmup steps	0.1	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.1	0.1	0.01	0.05	0.1	0.01	0.05
Task weight decay	0.01	0.11	0.1	0.01	0.11	0.1	0.01	0.1093	0.1	0.01	0.11	0.1	0.01	0.11	0.1
Task warmup steps	0.1	0.05	0.05	0.1	0.05	0.05	0.1	0.05	0.1	0.1	0.05	0.05	0.1	0.05	0.05
Batch size	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
Dropout	0.1	0.2	0.0	0.1	0.2	0.0	0.3	0.2	0.3	0.1	0.2	0.0	0.1	0.2	0.0

Table 2: Hyperparameters for three models on five datasets. The following GPUs were used for training: ITER was trained on an H100, ASP on an RTX 4090, and DiffusionNER on an A100. The total computing time is approximately 976.25 hours.

We keep both the size and content of the original test set fixed in our experiments. For each of the eleven contamination rates, we sample five splits, resulting in 55 training splits for each dataset.

$$\min(|\neg\mathbf{c}(\mathcal{D}_{train})|, |\mathbf{c}(\mathcal{D}_{train})|) \qquad (13)$$

Algorithm 1 Algorithm for creating a training split with contamination level $\gamma \in [0, 100]$ for seed $\sigma \in [0, 4]$.

```
Require: $\mathcal{D}_{train}$, $\mathcal{D}_{test}$, $\gamma$, $\sigma$
    $N \leftarrow \min(|\neg\mathbf{c}(\mathcal{D}_{train})|, |\mathbf{c}(\mathcal{D}_{train})|)$
    $N_{clean} = N \cdot (1 - \gamma/100)$
    $N_{contam.} = N \cdot (\gamma/100)$
    $A \leftarrow \textsc{RandSelect}(\neg\mathbf{c}(\mathcal{D}_{train}), N_{clean}, \sigma)$
    $B \leftarrow \textsc{RandSelect}(\mathbf{c}(\mathcal{D}_{train}), N_{contam.}, \sigma)$
    $\mathcal{C}^{\gamma,i}_{train} \leftarrow A \cup B$
```

Furthermore, we duplicate the original test set for each of the 55 training splits into three separate test sets

$$\begin{aligned} \mathcal{C}^{\gamma}_{test} &= \mathcal{D}_{test} \\ \mathcal{C}^{\gamma}_{clean\,test} &= \{(\mathbf{x}_i, E_i \setminus \mathcal{E}_{train})\} \\ \mathcal{C}^{\gamma}_{contam.\,test} &= \{(\mathbf{x}_i, E_i \cap \mathcal{E}_{train})\} \\ &\forall (\mathbf{x}_i, E_i) \in \mathcal{D}_{test} \end{aligned}$$

where both $\mathcal{C}^{\gamma}_{clean\,test}$ and $\mathcal{C}^{\gamma}_{contam.\,test}$ contain all samples from the test set, but only entities that are either ***clean*** or ***contaminated***.

5 Experiments

We outline the experimental setup for three main investigations: contamination training across multiple NER models, the application of the minimum cut algorithm for dataset splitting, and the quantification of state-of-the-art model overestimation due to contamination. The main results and evaluations are also presented, focusing on correlations between model evaluation and contamination, overestimation of F1 values in the state of the art, and analysis of datasets generated using minimum cut.

5.1 Contamination Training

To test how NER models are affected by different degrees of dataset contamination, we train an ensemble of three different state-of-the-art NER architectures on 55 training splits (see Section 4) for each of the five respective datasets. We use the encoder-decoder model ASP (Liu et al. (2022), *flan-t5-base*), the diffusion-based approach DiffusionNER (Shen et al. (2023), *bert-large-cased*) and the encoder-based model ITER (Hennen et al. (2024), *deberta-v3-small*). If available, we use existing hyperparameter configurations (see Table 2). We do not perform any hyperparameter optimization, as we do not want to find the optimal performance of each model, but rather observe the behavior of a model as it is exposed to increasing contamination rates.

Results. Our observations, supported by figures, tables and significance tests, are as follows:

1. When comparing different state-of-the-art architectures at different degrees of contamination, all suffer a similar performance loss on *clean* NEs compared to *contaminated* NEs (see Figures 4, 5, and 11). Especially a higher Recall Delta in Figure 4 with higher contamination illustrates the models are failing to classify enough clean NEs. The original Recall is hence not a meaningful measure since it is inflated by contaminated NEs. This is further verification of the need for test sets of clean – non-contaminated – NEs, for an accurate measure of model performance.

2. In addition, the contamination rate has a moderate to strong Pearson correlation with the ΔF_1 (0.57). This statistically proves that

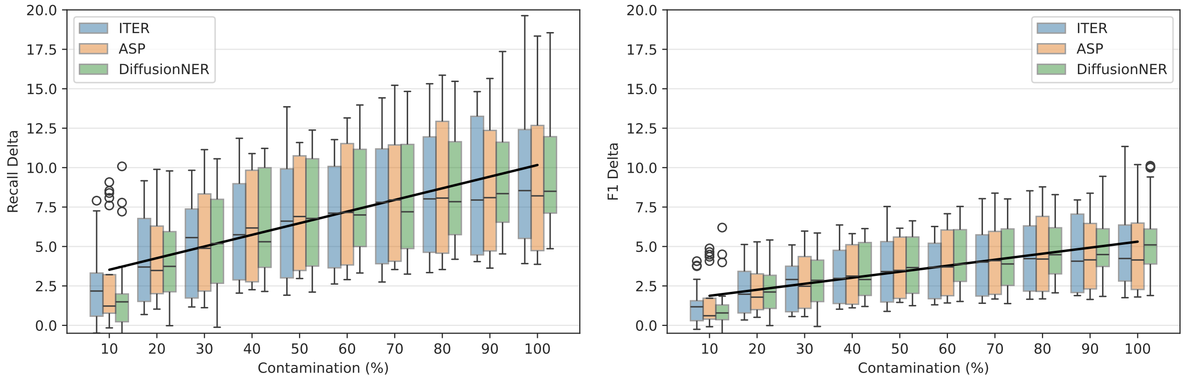

Figure 4: Recall delta ($R_{orig.} - R_{clean}$) and F1 delta (ΔF_1) for each model over all datasets.

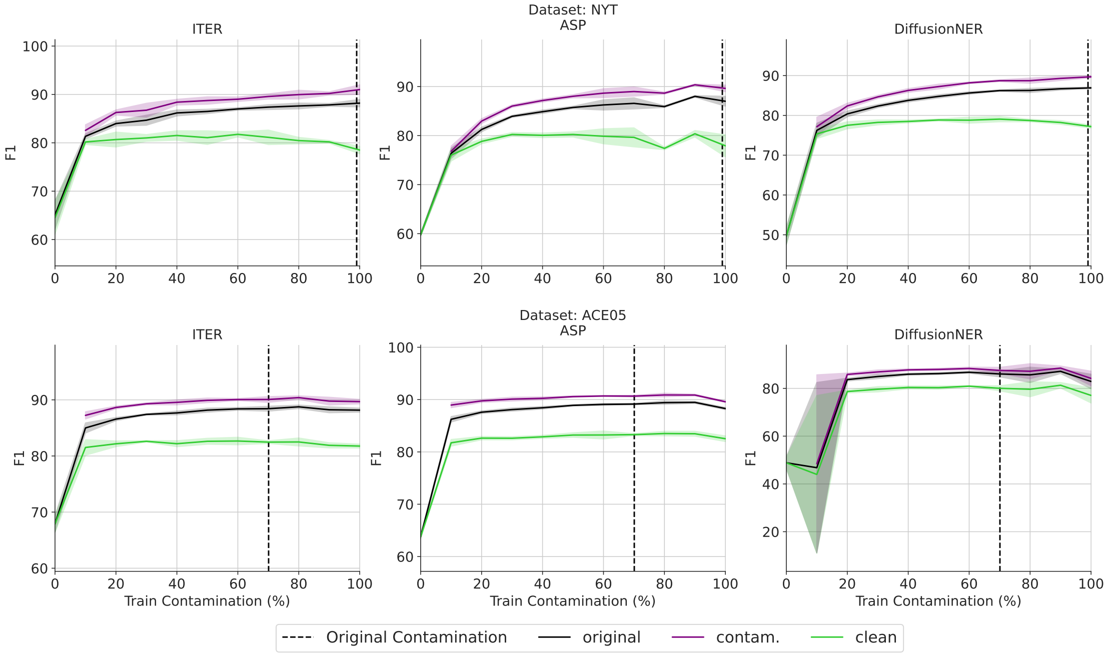

Figure 5: ITER, ASP, DIFFUSIONNER performance on original, *clean* and *contaminated* NYT and ACE05 test sets measured by original F_1, $F_{1(clean)}$ (Eq. 7) and $F_{1(conta)}$ respectively. Other datasets are visualized in Figure 11.

higher contamination leads to more inflated F1 scores, again highlighting the need to evaluate the aforementioned performance gap in future work. All of this indicates, that a lower contamination rate leads to a more meaningful F1 score, which motivates the creation of non-contaminated NER datasets.

3. To compute the Pearson correlation, we focus on the influence of the contamination rate on the different F1 values measured over all 825 evaluated models (see Figure 6). This evaluation is also done on a per-model basis in Figure 10. The contamination rate has a weaker correlation (0.18) with $F1_{clean}$ on *clean* NEs while it correlates moderately to strongly with *contaminated* F1 (0.52) – determined in the same way as in Equation 7, confirming that the contamination mostly benefits *contaminated* NEs.

4. We find no difference in performance per class when it comes to *clean* NEs from over- or under-represented classes (see Figure 9), although this is only assessable for ITER.

5.2 Quantifying Overestimation

To test how state-of-the-art models are overestimated by contaminated NEs, we train and test DIFFUSIONNER, ASP, and ITER on the full train set of all datasets and quantify the decrease in performance with $F_{1(clean)}$ and ΔF_1 using the *original*,

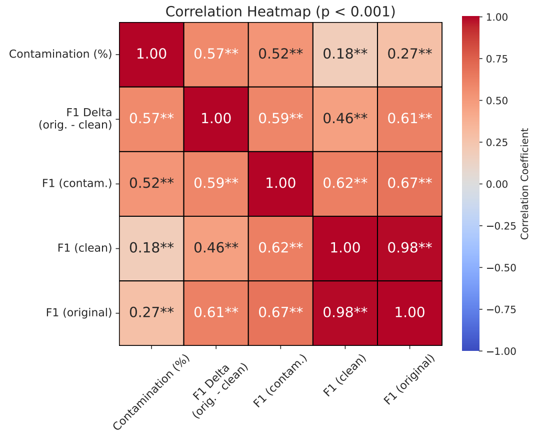

Figure 6: Correlation across all datasets, models, contamination rates. ** denotes high statistical significance.

contaminated and *clean* test splits.

Results. Training ASP, DiffusionNER and ITER on all original datasets, with evaluation on all three test splits per dataset, produces the following results shown in Table 3.

There is a noticeable difference between F1$_{clean}$ and F1$_{orig}$ for all architectures. This further suggests that the generalization ability of current state-of-the-art models is being overestimated. The recall distance between the original and clean scores is roughly twice the ΔF1 – in part because we are approximating $P_{(clean)^*}$ with $P_{(orig.)}$ – an over-approximation when $E_i^{(contam.)}$ is large in terms of size. This evaluation highlights the applicability of ΔF1 and recall to gain more meaningful insight into the real-world performance of future models.

Dataset contam. %	**Model**	**Recall**↑	***clean*** **Recall** ↑	$\Delta R\downarrow$	**F1** ↑	***clean*** **F1**↑	$\Delta F1\downarrow$
ACE05 70%	DiffusionNER	87.22	71.00	16.22	86.73	77.88	8.85
	ASP	90.50	78.58	11.92	90.20	83.86	6.34
	ITER	89.66	76.91	12.75	89.53	82.69	6.84
CoNLL04 83%	DiffusionNER	88.23	83.73	4.50	87.42	85.15	2.27
	ASP	89.16	82.90	6.26	88.30	85.11	3.19
	ITER	91.41	87.43	3.98	91.23	89.21	2.02
GENIA 74%	DiffusionNER	77.86	71.90	5.96	79.07	75.88	3.19
	ASP	–	–	–	–	–	–
	ITER	79.95	73.07	6.88	80.82	77.15	3.67
NYT 99%	DiffusionNER	96.41	81.82	14.59	95.03	87.35	7.68
	ASP	94.42	68.66	25.76	94.39	79.48	14.91
	ITER	94.53	72.65	21.88	94.48	82.12	12.36
SciERC 50%	DiffusionNER	70.21	66.67	3.54	66.26	64.64	1.62
	ASP	68.75	65.25	3.50	67.86	66.11	1.75
	ITER	71.25	67.39	3.86	69.24	67.36	1.88

Table 3: Comparison of Recall and F1 scores (*original*, *clean*) and ΔF1 when training on original training set. ITER (476M) uses *deberta-v3-large*, ASP (229M) uses *FLAN-T5-base* and DiffusionNER (381M) uses *bert-large-cased*. Training runs for ASP on GENIA did not converge.

5.3 Minimum Cut

To evaluate the minimum cut algorithm in generating dataset splits, we apply it to the datasets from Figure 3. We then evaluate its effectiveness by attempting to reconstruct the original dataset proportions, as shown in Table 1.

Results. By applying the minimum cut algorithm as described in Section 3.3, we achieve 0% contamination on ACE05, SciERC, GENIA, and ADE, and significantly reduce contamination in the remaining datasets.

In addition, for the resulting splits, it is crucial to consider the number of entity type samples per split, to ensure that each split contains a sufficient number of NE type samples for effective training and testing. Figure 13 visualizes all NE type counts for all minimum cut dataset splits and compares them to the original datasets. While the algorithm does not focus on one NE type distribution per se, it produces adequate splits for almost all (see Figures 3 and 13). For ACE05 and GENIA where the algorithm did not achieve satisfactory (at least 20) entity type counts, we try 80-10-10 split ratios and find 0% contamination as well as adequate counts per class. We encourage future users of the algorithm to try different train-dev-test sizes and consider the number of NE types.

6 Conclusion and Future Work

Our work extends and statistically substantiates the effects of entity contamination identified in previous research. Based on our results – specifically the correlation between contamination and the model performance gap between *clean* and *contaminated* NEs and the associated inflation of reported F1 scores – and the calls for better scoring (Søgaard et al., 2021), we create a novel method for splitting NER datasets that can help other researchers create *clean* splits and scores for existing and future datasets. Performance on new entities is quantified by the introduced metrics $F1_{clean}$ and $\Delta F1$, which are easy to calculate for future researchers and provide intuitive insights into the memorization-generalization gap. Furthermore, we share *contaminated* and *clean* test splits for all datasets for fairer and better generalization comparisons between existing and future models.

Future work could analyze possible relationships between contamination rate and different training and model sizes. While achieving 0% contamination should be considered ideal, alternative data splitting techniques could be explored to account for NE types while minimizing contamination as much as possible. Although expensive and highly complex, one could explore relation extraction, analyze overlapping triples and relations, and apply similar data splitting methods to this area of research.

7 Limitations

Our work has several limitations. First, our experiments do not account for the fact that a higher per-sample contamination rate increases both the number of NEs per sample and the NE contamination rate (see Figures 7 and 8). However, we see that the number of entities per class remains mostly the same across contamination splits (see Figure 9).

Second, we only analyze supervised training methods, which means that our results may not be directly applicable to Large Language Models (LLMs) or other training paradigms, thus stimulating future research. Third, in our experiments we are limited to relatively small, English-language datasets – mainly because of Equation 13. Fourth, we deliberately exclude the development set from our analysis to avoid complexities such as increased overlap. Finally, our minimum cut partitioning approach does not take into account the number of entity classes. Therefore, we need to include an additional step to verify that all named entity types are sufficiently represented across training, development, and test splits. When using the minimum cut algorithm, increasingly powerful computing resources are required as the dataset size increases. We also acknowledge that the creation of minimum cut splits reduces comparability between existing and future models, but we argue that future NER datasets should consider more robust partitioning strategies, and new models should additionally be evaluated with $\Delta F1$ on the published *contaminated* and *clean* test splits.

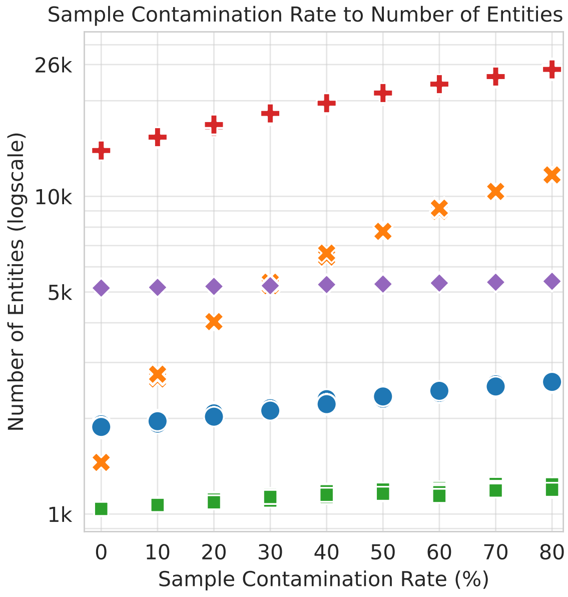

Figure 7: For some datasets, the number of entities cannot be controlled by randomly sampling from the contaminated part of the train set. The more named entities in a sample, the greater the chance of contamination, so more contamination occurs as the number of entities per named entity increases.

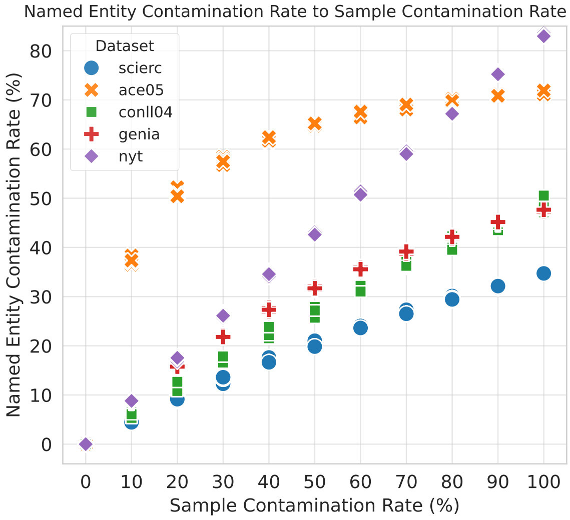

Figure 8: The correlation between NE contamination and sample contamination for each dataset is mostly linearly increasing. Not all values in (0,0) can be displayed.

8 Licenses

We provide information about the license or terms of use and/or distribution of datasets used in this work. ACE05 (Li and Ji, 2014) uses the LDC User Agreement for Non-Members. We provide a script in our Github to re-create all clean and contaminated test splits, as well as splits with different contamination levels required for our experiments. ADE (Gurulingappa et al., 2012) is publicly available here[3], but has no attributed license. For CoNLL04, (Roth and Yih, 2004; Gupta et al., 2016) published at NAACL 2004 is thus under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 International License. This also

[3] https://huggingface.co/datasets/ade-benchmark-corpus/ade_corpus_v2

applies to CoNLL03 (Sang and Meulder, 2003). GENIA (Kim et al., 2003) is listed in the GENIA Project License for Annotated Corpora. NYT (Riedel et al., 2010) is listed under the User License Agreement for The New York Times Annotated Corpus (LDC2008T19). SciERC (Luan et al., 2018) was presented at EMNLP 2018 and is therefore licensed under Creative Commons Attribution 4.0 International License

References

Oshin Agarwal, Yinfei Yang, Byron C. Wallace, and Ani Nenkova. 2020. Entity-switched datasets: An approach to auditing the in-domain robustness of named entity recognition models. *CoRR*, abs/2004.04123.

Oshin Agarwal, Yinfei Yang, Byron C. Wallace, and Ani Nenkova. 2021. Interpretability analysis for named entity recognition to understand system predictions and how they can improve. *Comput. Linguistics*, 47(1):117–140.

Alan Akbik, 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 54–59. Association for Computational Linguistics.

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Generalisation in named entity recognition: A quantitative analysis. *Comput. Speech Lang.*, 44:61–83.

Gabriel Bernier-Colborne and Philippe Langlais. 2020. Hardeval: Focusing on challenging tokens to assess robustness of NER. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, pages 1704–1711. European Language Resources Association.

Jacob Devlin, 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Jinlan Fu, Pengfei Liu, and Qi Zhang. 2020. Rethinking generalization of neural models: A named entity recognition case study. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7732–7739. AAAI Press.

Kyle Gorman and Steven Bedrick. 2019. We need to talk about standard splits. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2786–2791. Association for Computational Linguistics.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2537–2547.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *J. Biomed. Informatics*, 45(5):885–892.

Moritz Hennen, Florian Babl, and Michaela Geierhos. 2024. ITER: iterative transformer-based entity recognition and relation extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 11209–11223. Association for Computational Linguistics.

George Karypis and Vipin Kumar. 1998. *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. In *Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology, June 29 - July 3, 2003, Brisbane, Australia*, pages 180–182.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 402–412. The Association for Computer Linguistics.

Bill Yuchen Lin, Wenyang Gao, Jun Yan, Ryan Moreno, and Xiang Ren. 2021. Rockner: A simple method to create adversarial examples for evaluating the robustness of named entity recognition models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3728–3737. Association for Computational Linguistics.

Tianyu Liu, Yuchen Eleanor Jiang, Nicholas Monath, Ryan Cotterell, and Mrinmaya Sachan. 2022. Autoregressive structured prediction with language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 993–1005. Association for Computational Linguistics.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3219–3232. Association for Computational Linguistics.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 119–126. Association for Computational Linguistics.

Jeffrey Pennington, 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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Marco Túlio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with checklist. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4902–4912. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III*, volume 6323 of *Lecture Notes in Computer Science*, pages 148–163. Springer.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning, CoNLL 2004, Held in cooperation with HLT-NAACL 2004, Boston, Massachusetts, USA, May 6-7, 2004*, pages 1–8. ACL.

Erik F. Tjong Kim Sang 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, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Diffusionner: Boundary diffusion for named entity recognition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 3875–3890.

Anders Søgaard, Sebastian Ebert, Jasmijn Bastings, and Katja Filippova. 2021. We need to talk about random splits. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 1823–1832. Association for Computational Linguistics.

Jingxuan Tu and Constantine Lignos. 2021. TMR: evaluating NER recall on tough mentions. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop, EACL 2021, Online, April 19-23, 2021*, pages 155–163. Association for Computational Linguistics.

Sowmya Vajjala and Ramya Balasubramaniam. 2022. What do we really know about state of the art ner? In *Proceedings of the Thirteenth Language Resources and Evaluation Conference, LREC 2022, Marseille, France, 20-25 June 2022*, pages 5983–5993. European Language Resources Association.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. OntoNotes Release 5.0.

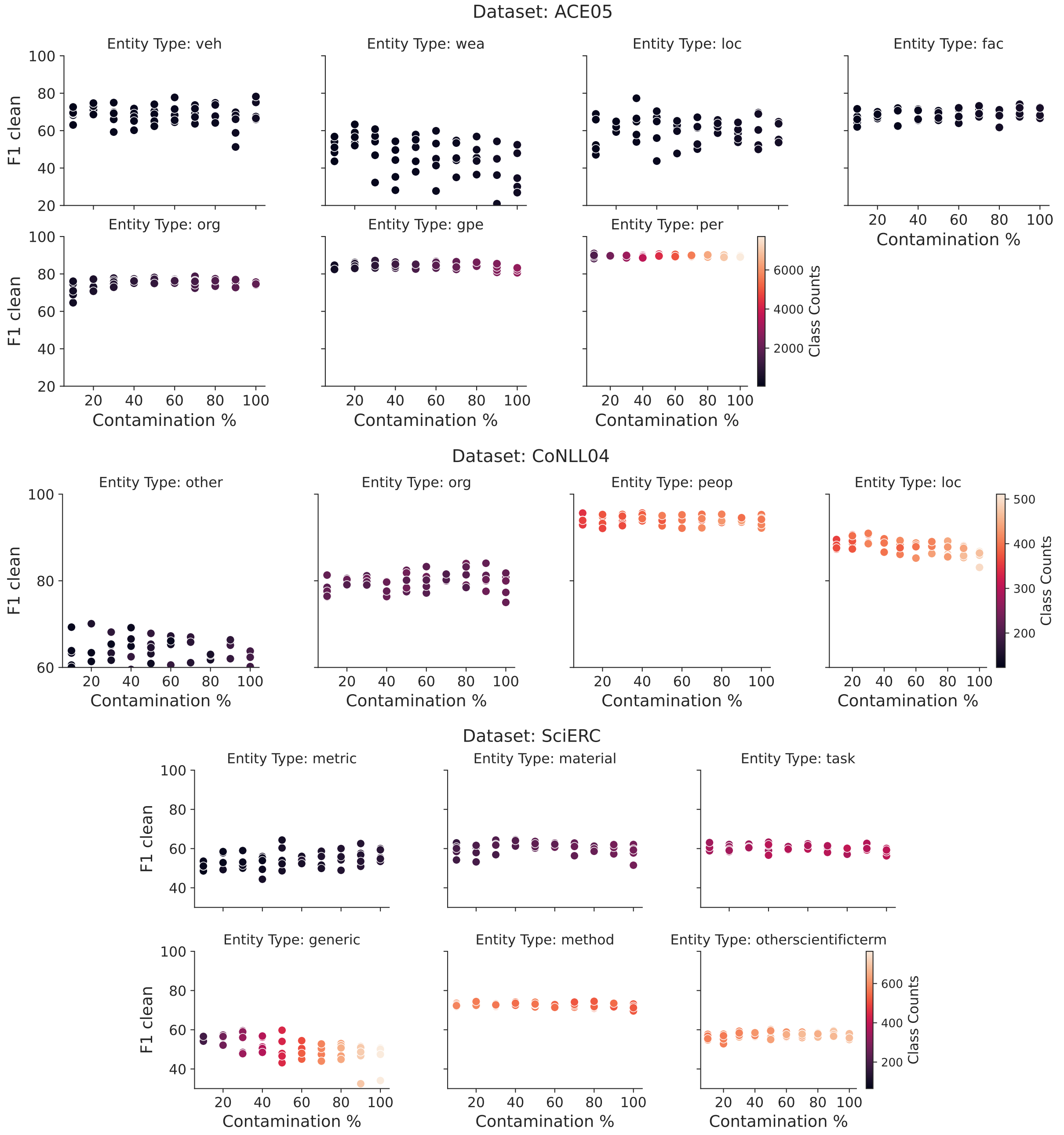

Figure 9: Per-class performance degradation over all datasets on ITER with respect to training counts in each contamination split. No class-specific scores are available for ASP and DIFFUSIONNER.

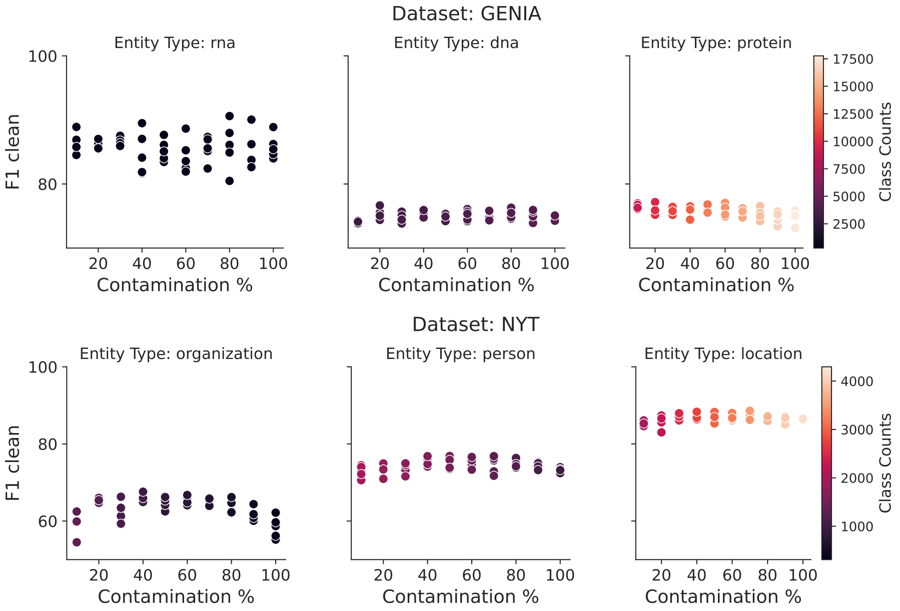

Figure 9: [continued] Per-class performance degradation over all datasets on ITER with respect to training counts in each contamination split. No class-specific scores are available for ASP and DIFFUSIONNER.

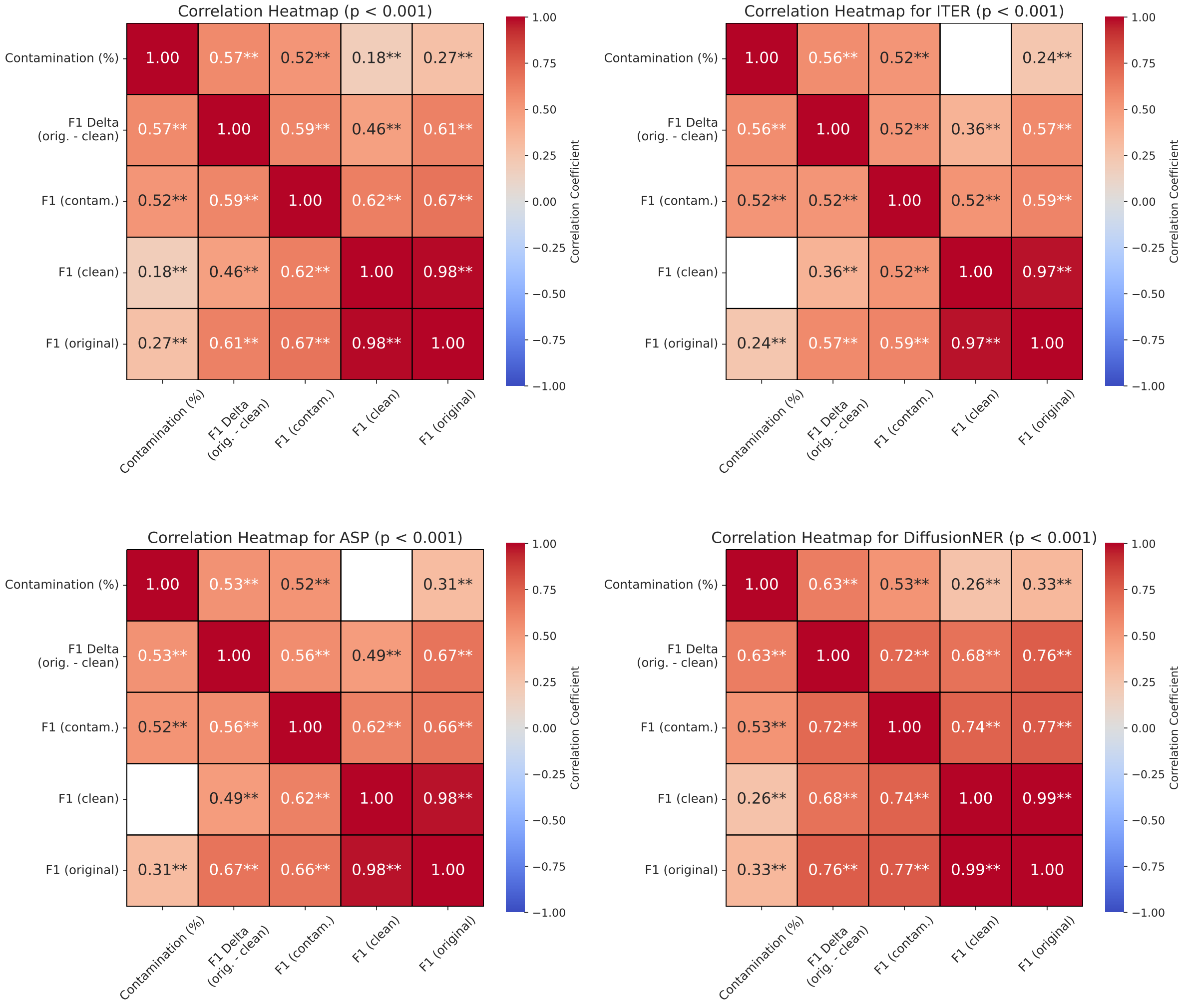

Figure 10: Combined correlation statistics and per model. Missing values are not statistically significant.

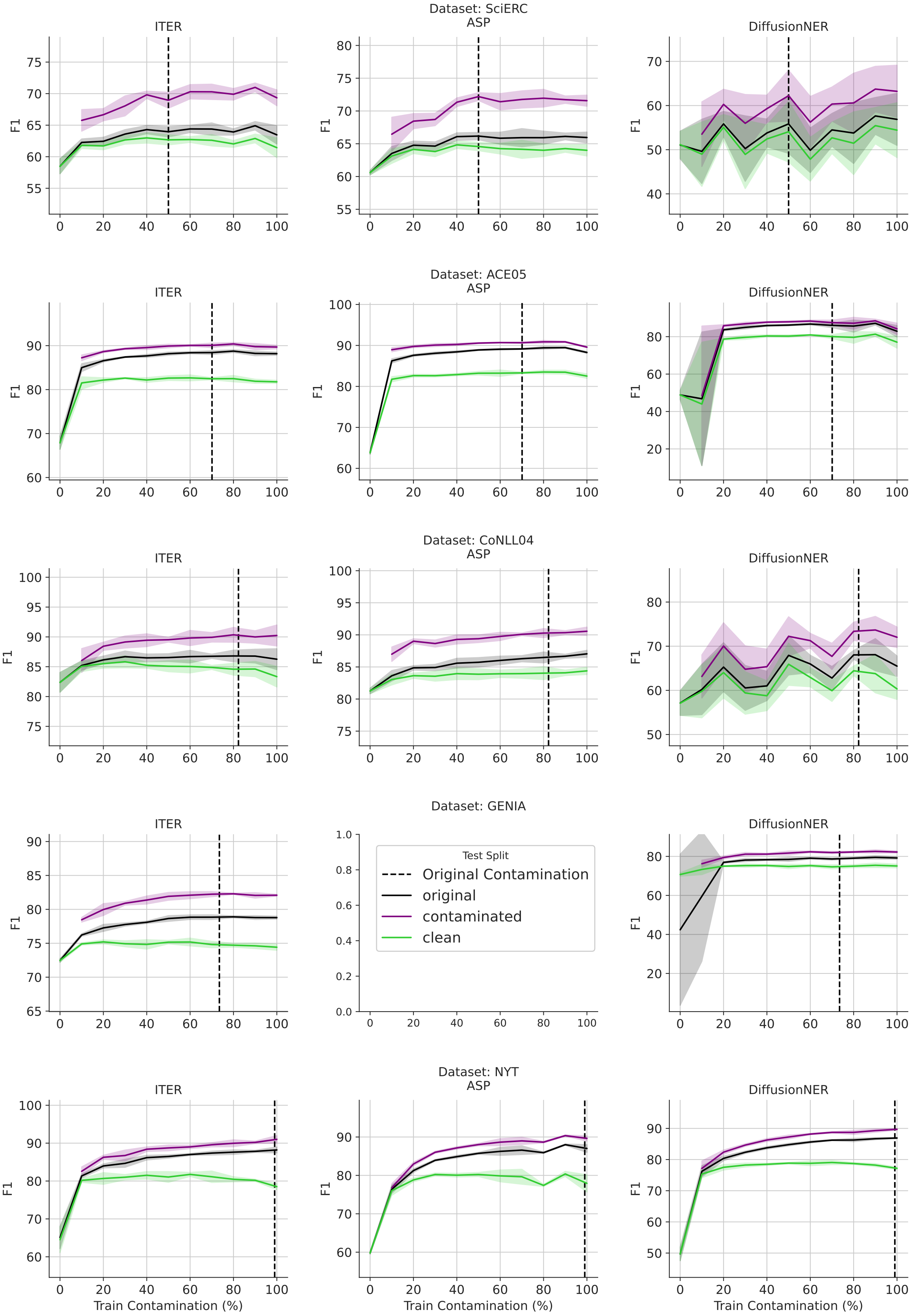

Figure 11: Detailed model performance on each dataset over all contamination rates. Training ASP on GENIA did not converge. We hypothesize that due to the highly nested NEs. DIFFUSIONNER may have imperfect hyperparameters as it struggles on splits with a lower number of NEs (ACE05 and GENIA).

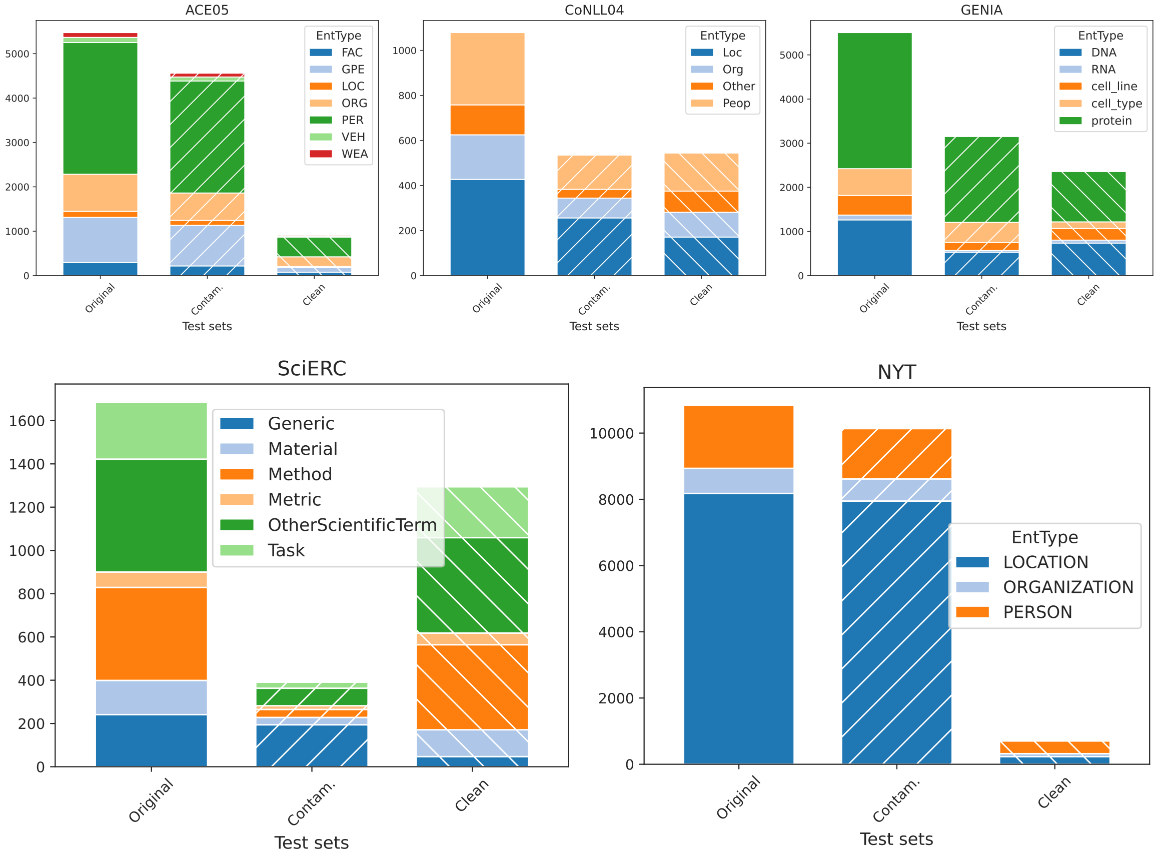

Figure 12: Per-class comparison between the original test set and the *contaminated* and *clean* entity splits – created depending on the original training set – and sharing for easier and better generalization and memorization evaluations.

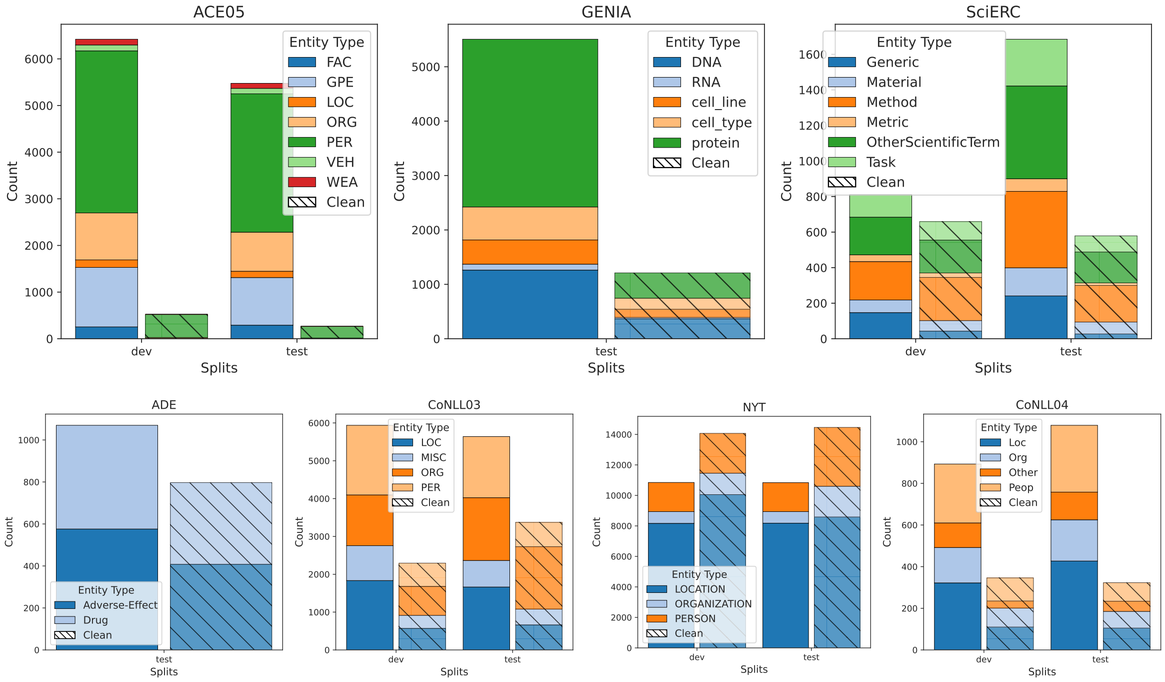

Figure 13: Entity class counts for original development and test sets vs. our minimum cut clean splits for all datasets. For all datasets, the training splits were successfully created and omitted for simplicity, readability and y-axis scaling issues.