SeqScore: Addressing Barriers to Reproducible Named Entity Recognition Evaluation

Chester Palen-Michel and **Nolan Holley** and **Constantine Lignos**
{cpalenmichel,lignos}@brandeis.edu
nrh2@williams.edu
Michtom School of Computer Science
Brandeis University

Abstract

To address a looming crisis of unreproducible evaluation for named entity recognition, we propose guidelines and introduce SeqScore, a software package to improve reproducibility. The guidelines we propose are extremely simple and center around transparency regarding how chunks are encoded and scored. We demonstrate that despite the apparent simplicity of NER evaluation, unreported differences in the scoring procedure can result in changes to scores that are both of noticeable magnitude and statistically significant. We describe SeqScore, which addresses many of the issues that cause replication failures.

1 Introduction

There are many complex tasks in natural language processing (NLP) where current evaluation standards are based around evolving metrics designed to correlate well with human judgments, some complex and some simple. For example, every year sees the introduction and careful evaluation of new metrics for machine translation, summarization, and natural language generation.

However, named entity recognition (NER) and other chunk extraction tasks have largely been evaluated the same way since the CoNLL shared tasks of the early 2000s (Tjong Kim Sang and Buchholz, 2000; Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). Following the CoNLL chunking and NER shared tasks, a true positive prediction typically requires exact matches[1] in span (the tokens or characters in a chunk) and the type assigned to the chunk (e.g., person).

With such a simple metric, it would seem that performing exact match NER evaluation would be trivially simple. Precision, recall, and F1 are easy to compute; all that is required is to count true positives, false positives, and false negatives. But when it comes to evaluation, challenges emerge in how evaluation is actually implemented. In the case of NER, as we will demonstrate these challenges emerge in the process of converting token-level annotations and system predictions into spans.

We do not think it is sufficient to point out these issues without attempting to provide a solution. Inspired by successful efforts in the machine translation community to address similar issues (Post, 2018), we began developing a toolkit and set of practices in summer 2020 to improve the replicability of experiments for NER. Our toolkit, SeqScore, provides researchers the necessary tools to score, validate, and examine both system outputs and annotation. SeqScore is open source and has been publicly released.[2]

This paper provides clear, easy-to-follow guidelines that facilitate reproducibility for NER (and other chunking task) experiments, explains them, provides a toolkit for easily following them, and then presents experiments using SeqScore that shows the impact of following them. The contribution of this paper is that it introduces and justifies guidelines for NER experiment reproducibility and provides a toolkit that makes them easy to follow.

2 Guidelines for reproducibility

We propose that in order to have sound and reproducible NER evaluation, the following guidelines should be followed:

1. Report what chunk encoding scheme was used (e.g. BIO).
2. Use an external scorer—not one internal to the system—and report which scorer was used.
3. Be explicit regarding what form of invalid label sequence repair was used.

[1] While there have been efforts to promote partial matching (Chinchor, 1998; Segura-Bedmar et al., 2013) and focusing on rarer entities (Derczynski et al., 2017), micro-averaaged exact match F1 is still the most common metric in use for NER.

[2] https://github.com/bltlab/seqscore

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 40–50

https://doi.org/10.26615/978-954-452-056-4_005

4. Only score against a gold standard that faithfully follows the chunk encoding scheme (e.g. BIO) in use.

5. Use good statistical practices when reporting results.

Many of these will seem like obvious ideas or practices that should be taken as a given. However, we have found almost no papers provide enough information to determine whether they are compliant with all of these guidelines specifically, very few papers report what scorer was used to produce the reported scores, and many that provide accompanying code do not include any evaluation code.

We examined several papers with state of the art NER results on the CoNLL 2003 dataset considering guidelines 1, 2, and 3. Of these papers Liu et al. (2019) follow 1, 2, and 3. Yamada et al. (2020) explicitly follows guidelines 2 and 3. Luoma and Pyysalo (2020) met guideline 1. Akbik et al. (2019) give details of their scoring decision for a previous paper, Akbik et al. (2018), mentioning they fixed an prior error in scoring, but do not explicitly detail how they fixed their scoring procedure for the baseline in Akbik et al. (2019). All other papers we surveyed did not explicitly satisfy guidelines 1, 2, and 3 (Wang et al., 2020, 2021; Shahzad et al., 2021; Baevski et al., 2019; Yu et al., 2020; Jiang et al., 2019; Li et al., 2020; Devlin et al., 2019).

As an example of a common departure from these practices, many papers that perform NER experiments publish the scores produced by NCRF++ (Yang et al., 2018). As previously detailed by Lignos and Kamyab (2020), NCRF++ uses an internal scorer with an undocumented label sequence repair method, so reporting any numbers from it would be contrary to guidelines 2 and 3. As Lignos and Kamyab demonstrated, on a specific subset of models that produce a high number of invalid transitions, that scorer produces F1 scores approximately half a point higher than the most commonly-used external scorer.

Guideline 4, which requires that the annotation precisely follow the chunk encoding scheme, also seems obvious. However, it was not actually followed for 2 of the 4 datasets for the CoNLL NER evaluations in 2002–3, as only the English and Dutch data were free of errors of this type (see Section 3.4). As these datasets are arguably the most famous NER datasets in existence, this is surprising. While this would only have a very minor impact on evaluation results, an evaluation cannot be reproducible if different scorers might interpret the gold standard differently due to differences in how invalid label sequences are handled (see Section 3.2). When examining other NER datasets, we have found more pervasive occurrences of invalid label sequences.

We will not discuss guideline 5 in any detail as practices change over time, but we will highlight the need to report a distribution of scores, rather than a single score. Reimers and Gurevych (2017) demonstrate this clearly for NER specifically, and SeqScore supports aggregating scoring across multiple runs and reporting summary statistics.

Many of these rules may seem like common sense, but by enumerating them, we provide a published "checklist" for researchers to follow.

3 The mechanics of NER evaluation

We now turn to explaining the mechanics of NER evaluation to explain why following these guidelines is important. In this section, we explain the subtleties of working with chunk encodings, which will reinforce the importance of following the first three guidelines.

3.1 The CoNLL tradition

Evaluating named entity recognition (NER) and similar chunking tasks is conceptually straightforward. The primary metrics are the precision, recall, and F1 of the extracted chunks, often called phrases, or for NER specifically, entities or mentions. The CoNLL-2000 shared task on chunking (Tjong Kim Sang and Buchholz, 2000) set the first and most long-lasting standard for distributing data for and evaluating chunking tasks.

Briefly, this standard—which we will call "CoNLL-style"—is that each dataset (train, etc.) is represented in a sentence-split, tokenized, delimited format. Each sentence consists of a sequence of lines, and each line contains at least a token and a label for that token. This format was accompanied by a scoring script, `conlleval`.[3] The labels give information about the spans of the chunks, using encoding schemes that have developed from the original IOB representation of Ramshaw and Marcus (1995).

While some models may use more complex encodings, the current standard for datasets is that

[3] https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt

chunks are encoded using BIO (begin, inside, outside), where the first token of each chunk receives a B- label, any following tokens in the chunk receive an I- label, and any tokens not contained in a chunk receive O label. This standard format, albeit with minor variations, has been used continuously for NER datasets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003; Benikova et al., 2014; Derczynski et al., 2017, among others).

Not every evaluation of these type of tasks has used this structure. Many datasets (e.g., Doddington et al., 2004; Hovy et al., 2006; Strassel and Tracey, 2016) use start and end offsets as the primary method of identifying spans, which can avoid issues related to tokenization and completely dissociates the annotation from the encoding of chunks using labels. As we show later, this dissociation automatically removes a major source of non-reproducibility in evaluation.

3.2 Scoring and repair

When it comes to evaluating system output, while the CoNLL-style format is truly simple, the process of using it for evaluation only *seems* simple.

The fundamental problem is that there is generally nothing that forces a system's output—or even the annotation (see Section 3.4)—to follow the intended state machine of the scheme for encoding chunks. While using a CRF may reduce—and constrained decoding (Lester et al., 2020) can eliminate—invalid label transitions, we must still be able to provide reproducible scoring methods for models that do not use these approaches.

As shown in Table 1, if we are using BIO encoding, a system could produce the sequence `O I-ORG`, illegally entering the "inside" state without going through "begin." Similarly, we might encounter a `B-MISC I-ORG` transition, beginning a chunk of type MISC but then continuing into an ORG chunk. Handling these invalid transitions requires an implicit or explicit *repair* method.

3.3 Repairs in practice

Since the `conlleval` scoring script, scoring predicted labels and repairing the invalid sequences that they contain have gone hand in hand. SeqScore follows this tradition, allowing for scoring labels that contain invalid sequences, but unlike `conlleval`, its repair methods are configurable, and unlike any other scorer we are aware of it supports inspecting the repaired label sequences through writing them to a file. By requiring the user to select the repair method and making a previously invisible feature visible, we are making it easy for users to follow guideline 3.

The user can specify whether to perform `conlleval`-style repair, to *discard* invalid sequences, or to make no repairs (*none*), which will raise an error if any invalid sequences are encountered. The differences between these repair methods are show in Table 1. Due to the complexities of attempting to repair invalid label sequences in BIOES,[4] repair is only supported for BIO and IOB encodings.

For example, given an I- followed by another I- of differing type such as `I-ORG I-LOC`, one could coerce either the first or second tag to match the other and maintain that this is all one mention. Another option is to treat the second tag as B- and begin a new mention. The latter is what most scorers do, but it should be noted that this is not *a priori* the correct choice.

As we describe each repair method in more detail, we will use examples from actual output on the CoNLL 2003 English data. We used SeqScore to find the invalid transitions in the BERT (Devlin et al., 2019) model output from Tu and Lignos (2021) and selected examples of each type.

For BIO, the possible invalid transitions are an I- preceded by O (Table 2), an I- preceded by an I- of a different type (Table 3), and an I- preceded by a B- of a different type (Table 4). `conlleval`-style scorers take the approach of changing any unexpected I- to a B- (thus our name "begin repair"), while discard-style scorers discard tokens started by an invalid sequence.

While begin and discard are the dominant repair methods in use, other methods are possible. Stanza's (Qi et al., 2020) undocumented approach (shown in Table 1) most closely resembles the discard repair method but does not discard all invalid sequences. For invalid sequences caused by a type mismatch, Stanza uses the type of the last token as the type for the whole mention, and unlike begin or discard, keeps the entire span as a single mention. For example, `B-ORG I-ORG E-LOC` would be decoded as one mention of type LOC, since LOC is the type of the last token. While we have described the repair methods that we are aware of, others may exist whether intentionally or as accidental deviations from more common repair methods.

[4] See Kroutikov (2019) for a discussion of the large number of ways to score invalid BIOES sequences.

Encoding	his	Liberal	Democratic	party	and	the	Russian	Duma
Valid	O	B-ORG	I-ORG	I-ORG	O	O	B-MISC	B-ORG
Invalid	O	**I-ORG**	I-ORG	I-ORG	O	O	B-MISC	**I-ORG**
Begin Repair	O	B-ORG	I-ORG	I-ORG	O	O	B-MISC	B-ORG
Discard Repair	O	O	O	O	O	O	B-MISC	O
Stanza Repair	O	O	O	O	O	O	B-ORG	I-ORG

Table 1: Valid and invalid BIO label sequences and repairs of the invalid sequence for the sentence fragment *his Liberal Democratic party and the Russian Duma* from the CoNLL-03 English training data (lines 3633–40). Labels that cause invalid transitions are bolded.

Repair	Labels					
None	O	**I-ORG**	I-ORG	O	B-PER	I-PER
Begin	O	**B-ORG**	I-ORG	O	B-PER	I-PER
Discard	O	**O**	O	O	B-PER	I-PER

Table 2: Original and repaired labels for *Trade and Industry Secretary Ian Lang* (CoNLL-03 English)

Repair	Labels				
None	O	B-ORG	I-ORG	**I-LOC**	O
Begin	O	B-ORG	I-ORG	**B-LOC**	O
Discard	O	B-ORG	I-ORG	**O**	O

Table 3: Original and repaired labels for *the Oceanic Control Center in* (CoNLL-03 English)

Lignos and Kamyab (2020) demonstrate the variation that occurs due to different repair methods for invalid label transitions, finding that at least one NER toolkit takes an alternate approach to handling invalid transitions that consistently produces higher F1 scores for some models than scoring with `conlleval`. Its approach is not incorrect; these "edge cases" can be interpreted different ways. However, the result is that different scorers can produce different scores for the same output, even though they claim to measure the same thing.

With these facts in mind, we believe that we are approaching a replicability crisis for NER and other chunking tasks, as scores cannot reliably be compared across papers, and replications can fail due to lack of information about the scoring procedure.

Repair	Labels				
None	O	B-LOC	**I-ORG**	I-ORG	O
Begin	O	B-LOC	**B-ORG**	I-ORG	O
Discard	O	B-LOC	**O**	O	O

Table 4: Original and repaired labels for *(Rangoon) University early* (CoNLL-03 English)

3.4 Invalid transitions in gold standards

Most discussions of invalid label sequences focus on the system output, but widely-used annotated data often contains invalid sequences as well. For example, the CoNLL-02 Spanish data is BIO-encoded but contains three invalid O to I- transitions, one in each of the train, testa, and testb subsets. The original IOB-encoded CoNLL-03 German data contains 10 invalid transitions.[5]

While they may not have major impacts on scores, these invalid sequences represent a replication issue. Any scorer using the discard repair can *remove mentions from the gold standard*; even if the number of mentions removed is small, it is not an acceptable evaluation practice for the scorer to effectively change the gold standard. If two researchers use different repair methods and the annotation contains invalid transitions, they are not only evaluating their system output differently but also not evaluating against the same gold standard.

One of the design tenets of SeqScore is that the detection and repair of invalid label sequences is *explicit* and *configurable*. SeqScore supports validating IO, IOB (IOB1), BIO (IOB2), and the isomorphic BIOES, BILOU, BMES, and BMEOW (Radford et al., 2015) encodings.

Here is an example of validating the CoNLL-02 Spanish training data using SeqScore:

```
$ seqscore validate --labels BIO
esp.train
Encountered 1 errors in 1 tokens, 8323
sequences, and 1 documents in esp.train
Invalid transition O -> I-LOC for token
'San' on line 221619
```

Our recommendation is that validation (and if

[5] We validated the CoNLL-02 Dutch, CoNLL-03 English, GermEval 2014 (Benikova et al., 2014), and W-NUT 2017 Emerging and Rare Entities (Derczynski et al., 2017) data sets and found no issues. The CoNLL-03 German data was corrected in a later BIO-encoded release.

needed, repair) be run on any invalid gold standards before scoring. Doing so guarantees that the gold standard faithfully follows the chunk encoding and has no invalid transitions, so regardless of the repair method used, the gold standard will be interpreted the same way. This practice satisfies guideline 4.

3.5 Label conversion

SeqScore also supports conversion between valid IO, IOB, BIO, BIOES, BMES, and BMEOW encodings using the `convert` subcommand. To prevent malformed output, it raises an error if the input contains any invalid sequences. The input can be repaired using the `repair` subcommand if it is IOB or BIO encoded. By separating repair and conversion, there are no "hidden" changes.

Many other label scheme converters convert labels at the token (rather than mention) level, which allows invalid sequences to propagate from the input to the output, sometimes with unexpected results. For example, Stanza converts to BIOES before scoring, passing along invalid label sequences from BIO to BIOES. While in BIO invalid transitions are limited to invalid I- labels, when invalid BIO sequences are converted to BIOES, there are many potential ways to convert them depending on how the input was interpreted.

4 Experiments

Our paper so far has discussed the importance of following the proposed guidelines but has not quantified the impact of doing so. We conducted a series of experiments on NER datasets to examine the extent to which the variations in scores from different repair methods applied to system outputs could lead to different results. These experiments also demonstrate the usefulness of SeqScore as a package for producing a reproducible and complete set of results for sequence labeling tasks.

As we show in the following experiments, NER using large multilingual models fine-tuned on lower-resourced datasets can show significant variation due to the scoring method used. We selected lower-resourced datasets for two reasons. First, we believe that this a major frontier for innovation in NER, and many new results will be reported in this area for years to come. Second, unlike higher-resourced datasets, the current state of the art for these datasets involves the application of large language models, which are particularly prone to producing invalid transitions.

Lang.	Begin	Discard	Δ	p-value
amh	71.19 ±1.20	**71.87** ±1.11	0.69	0.15
hau	89.78 ±0.41	**90.12** ±0.47	0.34	0.19
ibo	84.18 ±0.94	**84.57** ±0.86	0.39	0.33
kin	73.29 ±1.39	**74.51** ±1.26	1.22	0.06
lug	80.02 ±0.90	**80.32** ±0.85	0.30	0.29
luo	74.43 ±1.60	**74.96** ±1.56	0.53	0.27
pcm	87.89 ±0.72	**88.48** ±0.71	0.59	**0.03**
swa	87.43 ±0.55	**87.79** ±0.62	0.36	0.17
wol	64.74 ±1.82	**65.19** ±1.70	0.45	0.50
yor	77.63 ±0.17	**78.40** ±1.04	0.77	0.07

Table 5: Comparison of F1 scores across repair methods using XLM-R and MasakhaNER data.

Lang.	Begin	Discard	Δ	p-value
hau	86.87 ±0.38	**87.36** ±0.32	0.49	**0.01**
ibo	84.82 ±0.77	**85.14** ±0.72	0.32	0.32
kin	72.14 ±1.07	**73.41** ±1.00	1.27	**0.02**
lug	80.42 ±1.04	**80.83** ±1.05	0.41	0.29
luo	73.37 ±1.52	**74.18** ±1.53	0.81	0.15
pcm	87.97 ±0.62	**88.47** ±0.52	0.50	0.10
swa	86.73 ±0.49	**87.12** ±0.52	0.39	0.13
wol	65.35 ±1.58	**66.29** ±1.58	0.94	0.26
yor	78.96 ±0.86	**79.87** ±0.75	0.91	**0.03**

Table 6: Comparison of F1 scores across repair methods using mBERT and MasakhaNER data.

4.1 Comparing repair methods

We evaluated two multilingual models, multilingual BERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020), on the MasakhaNER datasets (Adelani et al., 2021) which cover 10 African languages. Both of the models were trained for 50 epochs, using 10 different random seeds; we report the mean and standard deviation (as mean ± std. deviation) of F1 across the seeds. Amharic was excluded from the mBERT experiments, as mBERT was not trained on its character set and thus predicts no names. XLM-R was trained on all 10 MasakhaNER languages.

For each language, we report the difference in mean F1 score between the begin and discard repair methods for all languages except Amharic. We also examine the difference in mean F1 score between two models that are scored using different repair methods. We provide statistical significance of each comparison using the Wilcoxon rank-sum test, which is computed using the ten F1 scores for each configuration. We use the Wilcoxon rank-sum test as it provides a robust comparison between two distributions without assuming that the scores are normally distributed, as there is no guarantee that scores follow such a distribution.

As shown in Tables 5 and 6, the discard repair method universally produces higher F1 scores. Using the significance threshold of $p < 0.05$ (bolded), a handful of the comparisons between repair methods are statistically significant, specifically the mBERT scores for Hausa, Kinyarwanda, and Yoruba and the XLM-R scores for Nigerian Pidgin. The results that are not statistically significant still demonstrate a noteworthy difference in F1, with the discard score being 0.61 points higher than begin on average across all comparisons, statistically significant or not.

It is not obvious why some models show a statistically significant difference with the discard repair method while others do not. Table 7 shows the average count of invalid label sequences across models and languages. While it is notable that Kinyarwanda and Yoruba have higher counts and happen to be significant in experiments using mBERT, Hausa has a comparatively low count as does Nigerian Pidgin which also had significant results (see Tables 5 and 6).

We also performed a small qualitative exploration of invalid transitions. We classified the invalid transitions in three ways: the begin strategy repairs in such a way that the repaired entity is correct, discard correctly discards a system predicted entity where there should be none, or neither correctly repairs the predicted entity. In the case that neither repair method is correct, discard favors a higher F1 since the begin strategy creates a false negative and false positive, while discard creates only a false negative.

We examined the invalid transitions for XLM-R Nigerian Pidgin and Wolof—selected due to having the lowest and highest p-values, respectively—in the test set output from the runs with the median scores.[6] Wolof had 3 of 13 invalid transitions correctly repaired by begin, while discard correctly repaired only 2. For Nigerian Pidgin, begin correctly repaired only 1 of 12 while discard correctly repaired 4 of 12.

While Nigerian Pidgin shows a larger gap between the effectiveness of the two repair methods, ultimately the number of repaired transitions is quite small due to their relative rarity and the small size of data sets for lower-resourced languages and thus it is difficult to draw conclusions. Our analysis could not identify a simple explanation for the differences observed across languages, and this merits further examination in future work.

[6] As there were an even number of runs (10), we used the higher of the two median runs for each language.

Lang.	XLM-R	mBERT
amh	13.9 ±4.33	-
hau	11.5 ±3.27	15.7 ±4.06
ibo	19.8 ±6.64	18.8 ±3.80
kin	39.3 ±7.04	40.3 ±8.87
lug	14.9 ±3.92	15.5 ±4.50
luo	12.7 ±5.01	14.9 ±3.93
pcm	17.8 ±6.81	12.9 ±5.59
swa	15.0 ±3.23	15.9 ±3.00
wol	10.8 ±4.07	17.3 ±5.36
yor	32.4 ±6.92	29.6 ±7.88

Table 7: Means and standard deviations of across runs of the number of invalid transitions repaired for system output for each model and language.

4.2 Simulating a real scenario

While we have shown that using a different repair method can sometimes lead to significant differences in F1 scores, using different repair methods on the exact same system output is not what happens in practice. Instead, the current situation is more likely to be that two different system outputs are unknowingly evaluated using differing repair methods by different authors.

To simulate a more likely situation, suppose one researcher has trained an mBERT model and another has trained an XLM-R model but neither explicitly mentions what repair strategy was used while scoring. We will now explore how the use of different repair methods would affect the conclusions drawn from researchers unknowingly using different scoring procedures when comparing their models.

In Table 8, we compare XLM-R using the begin repair to mBERT using discard and XLM-R using the discard repair to mBERT using begin. Suppose one team used XLM-R with the discard method to evaluate on Kinyarwanda while the other used mBERT and begin. The team using XLM-R with discard for scoring would have a score of 74.51 compared with the other team's score of 72.14, for a statistically significant difference in F1 of 2.37. If the teams switched scoring methods, the mean scores are much closer at 73.29 compared with 73.41, reducing the difference between the score to 0.12, which is statistically indistinguishable.

While Kinyarwanda is the most dramatic example, the difference in F1 changes considerably depending on the combination of repair methods used. Of the 9 language datasets, 7 show a change in

Lang.	XLM-R Begin	mBERT Discard	Δ	p-value	XLM-R Discard	mBERT Begin	Δ	p-value
hau	89.78	87.36	2.42	**0.0002**	90.12	86.87	3.25	**0.0002**
ibo	84.18	85.14	0.96	**0.0233**	84.57	84.82	0.25	0.5708
kin	73.29	73.41	0.12	0.9397	74.51	72.12	2.37	**0.0012**
lug	80.02	80.83	0.81	**0.0494**	80.32	80.42	0.10	0.8206
luo	74.43	74.18	0.25	0.8206	74.96	73.37	1.59	**0.0257**
pcm	87.89	88.47	0.58	**0.0452**	88.48	87.97	0.51	0.1306
swa	87.43	87.12	0.31	0.2265	87.79	86.73	1.06	**0.0012**
wol	64.74	66.29	1.55	0.0696	65.19	65.35	0.16	0.8206
yor	77.63	79.87	2.42	**0.0002**	78.40	78.96	0.56	0.3447

Table 8: Comparison crossing models and repair methods.

whether the difference between models is statistically significant depending on which repair method is used with each model, highlighting the important of guideline 3. If in addition to using different repair methods, if one researcher reported their best test set score with no information about the distribution—as we have discovered a recent state of the art NER paper did—there would likely be even larger differences.

These experiments show that if researchers do not report their full scoring procedure, they may inadvertently obfuscate which models actually perform better and their claims of improvement may just be statistical noise. However, if researchers use SeqScore, they can evaluate their system using multiple repair techniques without the risk of using two entirely different scorers, and their evaluations will be replicable by other researchers.

5 Comparison with other toolkits

To place SeqScore in the context of other work and to address the question of novelty, we compare SeqScore to other similar tools.

5.1 Design Considerations

The primary goal of SeqScore is to provide a highly usable scorer for chunk extraction sequence labeling tasks such as NER. But SeqScore is not just a scorer; it is designed to address the entire lifecycle of working with data: validating and examining annotation, converting between various chunk encodings, identifying and repairing invalid label sequences, and finally producing scores. SeqScore is implemented in Python, and like Git it uses subcommands to perform each task, for example `score` to score, and `validate` to validate files. While other packages exist for scoring and handling invalid label sequences, no other package has the convenience of everything in one place.

We believe this convenience lessens the barriers to providing more detailed reporting of scoring methods, and that this convenience and packaging together of all of these features is a novelty of SeqScore. For example, unlike `conlleval` and every other NER scorer we examined, SeqScore supports aggregating scores across multiple prediction files for the same reference. This enables the now-common practice of reporting the mean and standard deviation across runs, aiding in following guideline 5. While this is a simple feature, by reducing the effort required to report these scores, we believe we can help improve adoption of this practice. Of the papers we surveyed, only about two-thirds were clear about how many runs they used and whether their reported score was an average or a best run.

Table 9 compares the features of SeqScore against other packages for scoring and working with sequence labeling data for chunking tasks. While SeqScore is designed to include as many features as possible, there are some features it does not implement. One is partial match scoring, which is implemented in nervaluate (Segura-Bedmar et al., 2013)[7] following the MUC scoring approach (Chinchor and Sundheim, 1993). Also, SeqScore only processes CoNLL-style file formats.

NER scorers can broadly be grouped in terms of how closely they resemble the `conlleval` Perl script. There are direct re-implementations, those that score in the same spirit as `conlleval` but have additional features, and those that take a different approach to invalid labels. The set of scorers we examine is not exhaustive but covers the most widely-used ones.

[7] https://github.com/ivyleavedtoadflax/nervaluate

	SeqScore	Stanza	NCRF++	iobes	sighsmile	spyysalo	wnuteval	seqeval	conlleval
Warns for invalid label seqs.	✓			✓			✓		
begin repair	✓				✓	✓	✓	✓	✓
Discard repair	✓	✓	✓	✓				✓	
Converts label schemes	✓	✓	✓	✓					
Scoring	✓	✓	✓		✓	✓	✓	✓	✓
Aggregation across runs	✓								

Table 9: Comparison of package features

5.2 `conlleval` reimplementations

To the best of our knowledge after testing on a number of datasets and edge cases, each of these is a faithful replication of the original `conlleval` Perl script: spyysalo `conlleval.py`[8], and sighsmile `conlleval.py`[9]. The spyysalo and sighsmile re-implimentations differ from the original `conlleval` script mainly in that they have support for BIOES and are written in Python instead of Perl.

5.3 `conlleval`-style scorers

`wnuteval`[10] `wnuteval` is limited to the entity types used in the shared task it was developed for. It raises warnings about invalid transitions. However, it does not handle multiple mention encoding schemes. We also found that it does not raise any warnings or errors about uneven document lengths between system and gold files; while this seems like an unusual case to test, with the use of models set maximum sequence lengths as a decoding hyperparameter, it is common to accidentally truncate sentences when producing system output for evaluation.

seqeval seqeval (Nakayama, 2018) can score on many different label schemes. It is unique in being one of the only scorers we examined that has more than one approach to invalid label sequences, a feature added concurrently with the development of SeqScore. Seqeval refers to them as *default* and *strict* modes. *Default* is `conlleval`-style (begin), while *strict* is what we refer to as *discard*.

5.4 Internal scorers

There are also numerous internal scorers that are part of larger NLP toolkits and packages. Though there are certainly plenty of others, we examine evaluation methods found in NCRF++ and Stanza. Neither of the approaches of these internal scorers follow `conlleval`-style handling of invalid label sequences.

Stanza Stanza (Qi et al., 2020) is a collection of models and tools for NLP. It supports NER in multiple languages and includes its own scorer implementation. Stanza's scorer is similar to *discard* or seqeval's strict mode, with a few exceptions. Stanza also includes a number of tools for converting between schemes.

NCRF++ NCRF++ (Yang et al., 2018) is a framework for doing neural sequence labeling tasks in a highly configurable way. It implements its own scorer with an approach to invalid sequences using the *discard* method.

5.5 Handling invalid label transitions

Lester (2020) provides a library for parsing label schemes, identifying invalid label sequences, converting between label schemes, and enumerating the legality of possible transitions. While this library is very useful for handling label schemes and invalid transitions, it does not address what we believe is a necessary decoupling of the scorer from the handling of invalid label sequences. While it is capable of identifying invalid transitions and supporting one's own implementation to constrain or repair invalid sequences, it does not provide common methods for repairing invalid sequences.

Lignos and Kamyab (2020) demonstrate the difference that can occur when two scorers handle invalid label sequences differently. However, they do not provide any software to evaluate these differences and only test using CoNLL-03 English data with older neural models.

6 Conclusion

This paper has provided guidelines for reproducible NER research and demonstrated the importance of

[8] https://github.com/spyysalo/conlleval.py

[9] https://github.com/sighsmile/conlleval

[10] http://noisy-text.github.io/2017/files/wnuteval.py

following them, both by describing the principles behind them and demonstrating the impact on actual scores. Ultimately, researchers can choose to either accept the status quo—which for NER is non-reproducible research due to both a lack of standard practices and a lack of standard tools—or attempt to elevate the practice in the field to higher standards. We hope that by providing a software toolkit to help follow these guidelines, we have substantially reduced the barriers to performing reproducible research for this task.

While we have created a software package to accompany our recommendations—one that makes following them extremely simple—we do not claim that using SeqScore is necessary for reproducible results. Just like sacreBLEU is not the only way to produce a reproducible MT score, SeqScore is not the only reproducible way to score NER output. However, as it is actively maintained, well-tested, and it can handle multiple repair methods, we strongly encourage its use. By focusing on transparency and prioritizing supporting reproducible research in the design of SeqScore, we believe we have produced a toolkit that can have substantial positive impact on the field.

Adoption of this paper's recommendations by researchers will increase transparency in the scoring process and enable standardization of scoring methods in a field we believe is approaching a reproducibility crisis.

Acknowledgments

We would like to thank David Adelani for helping us work with the MasakhaNER data. We would also like to thank two anonymous reviewers for their useful feedback on the paper. Chester Palen-Michel was supported by the Alfred Schonwalter Graduate Computer Science Summer Fellowship through a gift from a generous donor to Brandeis University.

References

David Ifeoluwa Adelani, Jade Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, Stephen Mayhew, Israel Abebe Azime, Shamsuddeen Muhammad, Chris Chinenye Emezue, Joyce Nakatumba-Nabende, Perez Ogayo, Anuoluwapo Aremu, Catherine Gitau, Derguene Mbaye, Jesujoba Alabi, Seid Muhie Yimam, Tajuddeen Gwadabe, Ignatius Ezeani, Rubungo Andre Niyongabo, Jonathan Mukiibi, Verrah Otiende, Iroro Orife, Davis David, Samba Ngom, Tosin Adewumi, Paul Rayson, Mofetoluwa Adeyemi, Gerald Muriuki, Emmanuel Anebi, Chiamaka Chukwuneke, Nkiruka Odu, Eric Peter Wairagala, Samuel Oyerinde, Clemencia Siro, Tobius Saul Bateesa, Temilola Oloyede, Yvonne Wambui, Victor Akinode, Deborah Nabagereka, Maurice Katusiime, Ayodele Awokoya, Mouhamadane MBOUP, Dibora Gebreyohannes, Henok Tilaye, Kelechi Nwaike, Degaga Wolde, Abdoulaye Faye, Blessing Sibanda, Orevaoghene Ahia, Bonaventure F. P. Dossou, Kelechi Ogueji, Thierno Ibrahima DIOP, Abdoulaye Diallo, Adewale Akinfaderin, Tendai Marengereke, and Salomey Osei. 2021. MasakhaNER: Named entity recognition for african languages. *Transactions of the Association for Computational Linguistics (TACL)*. In press.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. 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 724–728, Minneapolis, Minnesota. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention 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 5360–5369, Hong Kong, China. Association for Computational Linguistics.

Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Padó. 2014. GermEval 2014 named entity recognition shared task: Companion paper.

Nancy Chinchor and Beth Sundheim. 1993. MUC-5 evaluation metrics. In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*.

Nancy A. Chinchor. 1998. Overview of MUC-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8440–8451, Online. Association for Computational Linguistics.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA. Association for Computational Linguistics.

Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2019. Improved differentiable architecture search for language modeling and named entity recognition. 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 3585–3590, Hong Kong, China. Association for Computational Linguistics.

Mike Kroutikov. 2019. 7776 ways to compute F1 for an NER task. `http://blog.innodatalabs.com/7776_ways_to_compute_f1_for_ner_task/`.

Brian Lester. 2020. iobes: Library for span level processing. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 115–119, Online. Association for Computational Linguistics.

Brian Lester, Daniel Pressel, Amy Hemmeter, Sagnik Ray Choudhury, and Srinivas Bangalore. 2020. Constrained decoding for computationally efficient named entity recognition taggers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1841–1848, Online. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Constantine Lignos and Marjan Kamyab. 2020. If you build your own NER scorer, non-replicable results will come. In *Proceedings of the First Workshop on Insights from Negative Results in NLP*, pages 94–99, Online. Association for Computational Linguistics.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019. GCDT: A global context enhanced deep transition architecture for sequence labeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2431–2441, Florence, Italy. Association for Computational Linguistics.

Jouni Luoma and Sampo Pyysalo. 2020. Exploring cross-sentence contexts for named entity recognition with BERT. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 904–914, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Will Radford, Xavier Carreras, and James Henderson. 2015. Named entity recognition with document-specific KB tag gazetteers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 512–517, Lisbon, Portugal. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Isabel Segura-Bedmar, Paloma Martínez, and María Herrero-Zazo. 2013. SemEval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (DDIExtraction 2013). In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 341–350, Atlanta, Georgia, USA. Association for Computational Linguistics.

Moemmur Shahzad, Ayesha Amin, Diego Esteves, and Axel-Cyrille Ngonga Ngomo. 2021. Inferner: an attentive model leveraging the sentence-level information for named entity recognition in microblogs. In *The International FLAIRS Conference Proceedings*, volume 34.

Stephanie Strassel and Jennifer Tracey. 2016. LORELEI language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3273–3280, Portorož, Slovenia. European Language Resources Association (ELRA).

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.

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 at HLT-NAACL 2003*, pages 142–147.

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*, pages 155–163, Online. Association for Computational Linguistics.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Improving named entity recognition by external context retrieving and cooperative learning. 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 1800–1812, Online. Association for Computational Linguistics.

Yiyi Wang, Emily Prud'hommeaux, Meysam Asgari, and Jill Dolata. 2020. Automated scoring of clinical expressive language evaluation tasks. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 177–185, Seattle, WA, USA → Online. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.