Fine-Grained Error Analysis and Fair Evaluation of Labeled Spans

Katrin Ortmann
Department of Linguistics
Fakultät für Philologie
Ruhr-Universität Bochum
ortmann@linguistics.rub.de

Abstract
The traditional evaluation of labeled spans with precision, recall, and $F_1$-score has undesirable effects due to double penalties. Annotations with incorrect label or boundaries count as two errors instead of one, despite being closer to the target annotation than false positives or false negatives. In this paper, new error types are introduced, which more accurately reflect true annotation quality and ensure that every annotation counts only once. An algorithm for error identification in flat and multi-level annotations is presented and complemented with a proposal on how to calculate meaningful precision, recall, and $F_1$-scores based on the more fine-grained error types. The exemplary application to three different annotation tasks (NER, chunking, parsing) shows that the suggested procedure not only prevents double penalties but also allows for a more detailed error analysis, thereby providing more insight into the actual weaknesses of a system.

Keywords: Evaluation, $F$-score, labeled spans, double penalties, error analysis

1. Introduction

Evaluation in NLP serves two main purposes: (i) determining how good a system is at a given task and comparing its performance to other systems, and (ii) analyzing the errors a system makes to be able to improve it. For 1:1 mapping tasks like POS tagging, the procedure is clear-cut. Every token receives exactly one tag, and the number of correctly assigned tags is compared to the incorrect tags and reported as accuracy, possibly accompanied by a confusion matrix.

For tasks that include the annotation of spans or do not require every token to receive an annotation, e.g., tokenization, named entity recognition (NER), or chunking, the incorrect annotations can be further divided into false positives ($FP$, i.e., superfluous annotations) and false negatives ($FN$, i.e., missing annotations). System performance, in this case, is measured by comparing the two types of incorrect annotations with the number of true positives ($TP$), and the results are reported as recall (Eq. [1] ‘how many annotations that should be present are actually there’) and precision (Eq. [2] ‘how many of the annotations that are present are actually correct’). Usually, there is a trade-off between precision and recall because improving one likely worsens the other.

\[
Recall = \frac{TP}{TP + FN} \quad (1)
\]

\[
Precision = \frac{TP}{TP + FP} \quad (2)
\]

The harmonic mean of precision and recall, better known as $F_1$-score, is consulted to compare different systems based on a single number (Eq. [3]).

\[
F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3)
\]

There are certain issues with these evaluation metrics, though (cf., e.g., Shao et al. (2017) for the task of word tokenization). The focus of this paper will be on the yet unsolved problem of double penalties for the annotation of labeled spans. When labeled spans are evaluated in the traditional way, in trivial cases as displayed in example [1], one (missing) annotation counts as one true positive or one error, respectively.

\[
\begin{array}{c|c|c|c}
Target & A & A & - \\
System  & A & _ & A \\
\end{array}
\]

However, if a system annotates a span that overlaps with the correct annotation but is not identical to it, one annotation is counted as two errors as in example [2] because the target annotation is missing ($FN$), while another annotation is present ($FP$).

\[
\begin{array}{c|c|c|c}
Target & A & A & - \\
System  & A & _ & A \\
\end{array}
\]

This phenomenon is especially undesirable since the annotations in [2] are closer to the target annotation than completely missing or superfluous annotations, i.e., $FN$s and $FP$s, as in [1]. Optimizing a system based on these metrics could thus encourage the system to skip difficult or uncertain cases because missing an annotation ($FN$) is punished less than getting it almost right ($FN+FP$). Intuitively, these close-to-correct errors should be punished equally or maybe even less than the errors in [1], and not vice versa.

Also, the traditional evaluation with only two error categories does not provide information about the actual weaknesses of a system, which is critically important for improving performance.
2018] Manning, 2011). Instead, a manual error analysis would be necessary to distinguish between the two very different error types in example (2).

As most researchers are likely aware of this problem (cf., e.g., Jurafsky and Martin (2021)), there have been various attempts to deal with it, e.g., by performing qualitative error analyses (Braı̂oveanu et al., 2018] Manning, 2011), counting overlapping tokens or characters (Potthast et al., 2010), or introducing partial annotation scores or relaxed evaluation metrics (Röder et al., 2018; Ji and Nothman, 2016). However, there is no universal solution yet, and the traditional metrics are still widely used for evaluating labeled spans despite their drawbacks.

This paper suggests an approach to a more fair evaluation of labeled spans that prevents double penalties for a single annotation and, at the same time, allows for a more fine-grained error analysis. First, Section 2.1 introduces new error types that help to distinguish between different kinds of overlapping spans. Section 2.2 then discusses ways to calculate precision, recall, and F1-scores based on these error types. Afterwards, Section 5 presents an algorithm for the identification of the different error types in flat and multi-level annotations. Finally, in Section 6 the results of the traditional evaluation method are compared to the fair evaluation for different types of annotations. The paper concludes with a discussion in Section 7.

2. Fair evaluation

The enterprise of this paper was inspired by Manning (2006], who explicitly brings up the problem of double penalties in NER evaluation. Similar to the remarks above, he argues that one should not optimize NER systems for F1 because the metric is dysfunctional for sparse annotations. Although he focuses on named entity recognition, the same also holds for other types of labeled spans, e.g., chunks or syntactic constituents. As an alternative, Manning (2006] suggests the distinction of different error types, which will be picked up and expanded upon in the next section 2.1. From his considerations, it remains unclear, though, how these error types should be used to compare different NLP systems, which is the topic of Section 2.2.

2.1. Error types

The traditional evaluation only considers true positives (TP), false positives (FP), and false negatives (FN). However, example (2) already pointed out that a restriction to the latter two error types does not reflect the actual annotation quality in the case of overlapping spans. FPs and FNs should therefore be used exclusively to refer to 1:0 and 0:1 mappings as displayed in example (1). For cases in which the system annotation overlaps with the target annotation but is not identical to it, Manning (2006] suggests the distinction of three additional error types:

**LE (labeling error):** Identical span, different label

**BE (boundary error):** Identical label, different (overlapping) span

**LBE (labeling-boundary error):** Different label, different (overlapping) span

The three additional error types are illustrated in example (3) As intended, their application resolves the problem of double penalties because one annotation now counts as one error instead of two. Moreover, they enable a more detailed error analysis and allow to distinguish between entirely missing or superfluous annotations and almost correct annotations, which are often more frequent than actual FPs and FNs (Manning, 2006; Ortmann, 2021a; Ortmann, 2021b).

As intended, their application resolves the problem of double penalties because one annotation now counts as one error instead of two. Moreover, they enable a more detailed error analysis and allow to distinguish between entirely missing or superfluous annotations and almost correct annotations, which are often more frequent than actual FPs and FNs (Manning, 2006; Ortmann, 2021a; Ortmann, 2021b).

In the case of boundary errors, it is possible to make the evaluation even more fine-grained by distinguishing whether the system’s annotation is smaller (BEb) or larger (BEl) than the target span or whether it overlaps with it (BEo). Example (4) displays the three sub-types of boundary errors, which provide even more details on a system’s weaknesses, indicating possible starting points for improvement.

Annotations that overlap with two (or more) spans, at least one of which has the same label, should be counted as BE and not LBE. In total, for n target annotations and m system annotations, the number of true positives plus errors always lies between max(n, m) and n + m. Both examples in (5) should thus yield three errors.

In the literature, even more error types have been introduced. While some of them are only relevant to a specific annotation type (e.g., [Braı̂oveanu et al. (2018] with an error taxonomy for Named Entity Linking), other categories like errors in the gold standard (Manning, 2011] can only be recognized with a manual analysis. For practical reasons, these error types are not discussed further in this paper. But if their frequency is known for a given data set, they could be integrated into the analysis and calculation of metrics similar to the error types presented here.

Depending on the intended application, it would also be possible to distinguish whether one of the system boundaries, left or right, is identical to the target boundary to provide even more insight into the actual errors. The same distinctions could be made for labeling-boundary errors, but they would not provide much additional information since label and span of the system annotation both differ from the target. Therefore, LBE sub-types are not considered here.

1 In the literature, even more error types have been introduced. While some of them are only relevant to a specific annotation type (e.g., Braı̂oveanu et al. (2018] with an error taxonomy for Named Entity Linking), other categories like errors in the gold standard (Manning, 2011] can only be recognized with a manual analysis. For practical reasons, these error types are not discussed further in this paper. But if their frequency is known for a given data set, they could be integrated into the analysis and calculation of metrics similar to the error types presented here.

2 Depending on the intended application, it would also be possible to distinguish whether one of the system boundaries, left or right, is identical to the target boundary to provide even more insight into the actual errors. The same distinctions could be made for labeling-boundary errors, but they would not provide much additional information since label and span of the system annotation both differ from the target. Therefore, LBE sub-types are not considered here.
2.2. Precision, Recall, F$_1$-score

The fine-grained distinction of error types as described in Section 2.1 solves the problem of double penalties and enables a more detailed error analysis. However, the raw number of errors is unsuitable for comparing different systems, especially across different data sets. Instead, it would be desirable to include these error types in the calculation of precision, recall, and F$_1$-score. In Ortmann (2021b), I argued that the additional error types refer to an existing annotation and should therefore count as false positives for the calculation of F$_1$-scores. Read et al. (2012), instead, count these kinds of errors as false negatives to prevent double penalties. For the resulting F$_1$-score, the decision makes no difference since, mathematically, it only depends on the number of true positives and errors (cf. Eq. 5).

\[
F_1 = \frac{2 \times \text{Prec} \times \text{Rec}}{\text{Prec} + \text{Rec}} = \frac{2 \times TP}{2 \times TP + \text{errors}} \quad (4)
\]

However, counting the overlapping errors as either FP or FN makes recall and precision values hard to interpret in a meaningful way. As each of the error types indicates a (partly) missing target annotation and, at the same time, a (partly) incorrect system annotation, it seems more appropriate to count the new error types as half FP and half FN (cf. Eq. 5).

\[
1LE = 1BE = 1LBE = 0.5FP + 0.5FN \quad (5)
\]

As explained above, this does not change the F$_1$-score, but it renders precision and recall values more meaningful again.

**Weighted Evaluation** Depending on the application, it could also be useful to make the evaluation more nuanced by introducing specific weights for different error types. For example, boundary errors could be considered less severe, e.g., in a search context because the target span is still found by the system. In this case, BEs could, for example, be counted as 50% true positives as in equation (6).

\[
1BE = 0.5TP + 0.25FP + 0.25FN \quad (6)
\]

When different types of boundary errors are distinguished, the evaluation could be even more differentiated (cf., e.g., Eq. 7) to more precisely reflect true annotation quality in precision and recall. It is important to note, though, that contrary to equation (5), the weighting in equations (6) and (7) also affects F$_1$-scores because it increases the total number of TPs.

\[
1BE_s = 0.5TP + 0.5FN
\]

\[
1BE_l = 0.5TP + 0.5FP \quad (7)
\]

\[
1BE_e = 0.5TP + 0.25FP + 0.25FN
\]

3. Algorithm

For traditional evaluation with only two error categories (false positives and false negatives), the algorithm to identify error types is simple. If a target span was recognized by the system, it counts as TP. Spans only present in the system output are FPs, and target spans missing in the system annotation are FNs (cf. Algorithm 1). The different categories can be identified for individual labels or all labels overall.

**Algorithm 1:** Traditional error type identification

**Input:** A set of target spans $T$ and system spans $S$.

**Spans are triples of label $l$, begin $b$, and end $e$**

**Output:** Number of TP, FP, and FN per label and overall

1. Count every span $t \in T \cap S$ as TP for $l_t$
2. Count every span $t \in T \setminus S$ as FN for $l_t$
3. Count every span $s \in S \setminus T$ as FP for $l_s$
4. Sum up TPs, FPs, and FNs across labels
5. Return results per label and overall

Identifying the fine-grained error types is more complicated. In particular, there are the following difficulties:

(i) One target span can overlap with more than one system span and vice versa. Nevertheless, the number of TPs plus errors should always lie between $\max(n, m)$ and $n + m$ for $n$ target and $m$ system annotations, i.e., every system annotation and every target annotation should count exactly once. To achieve this, spans are removed from the input list as soon as their first matching counterpart is found. To ensure that other potential counterparts are also matched to the correct span, the algorithm must keep track of the already matched tokens in each span. In combination, these two steps allow matching multiply overlapping spans to their correct counterparts without counting any span twice.

(ii) There are cases in which one span could correspond to different error types, e.g., BE and LBE, as in example (5). As described in Section 2.1 BEs should be preferred over LBEs. The algorithm will therefore proceed in four incremental steps, starting with easy to identify spans with 1:1 mappings (TPs and LEs, step 1), followed by boundary errors (step 2), and labeling-boundary errors (step 3), and finally the remaining 1:0 and 0:1 mappings (FNs and FPs, step 4).

(iii) Per-label evaluation is also less straightforward for the more fine-grained error types. There are two main problems:

1. Which error type should be assigned to multiply overlapping spans? In the following, each span is counted as the first matching error type.
**Algorithm 2**: Identification of fine-grained error types in labeled spans

**Input**: A list of target spans $T$ and system spans $S$, sorted by span length from shortest to longest. Each span is a 4-tuple of label $l$, begin $b$, end $e$, and a set of included tokens $\text{toks} (1..n)$. $b = e$ for spans of length 1.

**Output**: Number of $\text{TP}$, $\text{FP}$, $\text{LE}$, $\text{BE}_a$, $\text{BE}_t$, $\text{BE}_o$, and $\text{FN}$ per label and overall

**Function definitions:**
Let $\text{BE}_\text{type}(t, s)$ return the correct type of $\text{BE}_a$, $\text{BE}_t$, and $\text{BE}_o$ for spans $t$ and $s$.
Let $\text{get}_{\text{BE}}(t, s \in S)$ return the most similar span $s \in S$ for $t$ with $l_t = l_s$ and $|\text{toks}_t \cap \text{toks}_s| \geq 1$.
Let $\text{get}_{\text{LBE}}(t, s \in S)$ return the most similar span $s \in S$ for $t$ with $l_t \neq l_s$ and $|\text{toks}_t \cap \text{toks}_s| \geq 1$.
Let $\text{update}_{\text{toks}}(\text{toks}_t, \text{toks}_s)$ set $\text{toks}_t = \text{toks}_t \setminus \text{toks}_s$ and $\text{toks}_s = \text{toks}_t \setminus \text{toks}_s$.
Let $\text{move}(t, T \rightarrow M)$ remove $t$ from $T$ and add it to $M$.

**Step 1: Count 1:1 mappings (true positives and labeling errors)**
1. Count identical spans $t \in T = s \in S$ as $\text{TP}$ for $l_t$ and remove $t$ from $T$ and $s$ from $S$
2. Count spans with $l_t \neq l_s$, $b_t = b_s$, and $e_t = e_s$ as $\text{LE}$ for $l_t$ and remove $t$ from $T$ and $s$ from $S$

**Step 2: Count boundary errors**
3. Create empty lists $M_t$ and $M_s$ for matched spans
4. For each $t \in T$:
5. \hspace{1em} Count $\text{get}_{\text{BE}}(t, s \in S)$ as $\text{BE}_\text{type}(t, s)$ for $l_t$
6. \hspace{1em} Update matches with $\text{update}_{\text{toks}}(\text{toks}_t, \text{toks}_s)$, $\text{move}(t, T \rightarrow M_t)$, and $\text{move}(s, S \rightarrow M_s)$
7. For each $t \in T$:
8. \hspace{1em} Count $\text{get}_{\text{BE}}(t, s \in M_s)$ as $\text{BE}_\text{type}(t, s)$ for $l_t$
9. \hspace{1em} Update matches with $\text{update}_{\text{toks}}(\text{toks}_t, \text{toks}_s)$ and $\text{move}(t, T \rightarrow M_t)$
10. For each $s \in S$:
11. \hspace{1em} Count $\text{get}_{\text{BE}}(s, t \in M_t)$ as $\text{BE}_\text{type}(t, s)$ for $l_t$
12. \hspace{1em} Update matches with $\text{update}_{\text{toks}}(\text{toks}_t, \text{toks}_s)$ and $\text{move}(s, S \rightarrow M_s)$
13. Calculate $\text{BE} = \text{BE}_a + \text{BE}_t + \text{BE}_o$

**Step 3: Count labeling-boundary errors**
14. For each $t \in T$:
15. \hspace{1em} Count $\text{get}_{\text{LBE}}(t, s \in S)$ as $\text{LBE}$ for $l_t$
16. \hspace{1em} Update matches with $\text{update}_{\text{toks}}(\text{toks}_t, \text{toks}_s)$, $\text{move}(t, T \rightarrow M_t)$, and $\text{move}(s, S \rightarrow M_s)$
17. For each $t \in T$:
18. \hspace{1em} Count $\text{get}_{\text{LBE}}(t, s \in M_s)$ as $\text{LBE}$ for $l_t$
19. \hspace{1em} Update matches with $\text{update}_{\text{toks}}(\text{toks}_t, \text{toks}_s)$ and $\text{move}(t, T \rightarrow M_t)$
20. For each $s \in S$:
21. \hspace{1em} Count $\text{get}_{\text{LBE}}(s, t \in M_t)$ as $\text{LBE}$ for $l_t$
22. \hspace{1em} Update matches with $\text{update}_{\text{toks}}(\text{toks}_t, \text{toks}_s)$ and $\text{move}(s, S \rightarrow M_s)$

**Step 4: Count false positives and negatives**
23. Count every $t \in T$ as $\text{FN}$ for $l_t$
24. Count every $s \in S$ as $\text{FP}$ for $l_s$

**Step 5:** Return results per label and the overall sum across labels

2. Should $\text{LE}$ and $\text{LBE}$ count as errors for the target or the system label? One possible solution is to put the focus on the target labels, i.e., to evaluate how the target labels were annotated by the system. In this case, all errors (except false positives) count for the label of the target span they are matched to. The resulting error distribution then gives a detailed picture of how well the target spans were identified by the system. If the focus is on the system annotation, the same process could be applied to the system labels. A confusion matrix can represent both directions at the same time.

(iv) In evaluating hierarchical annotations (e.g., constituency trees), it is common practice to compare the annotated spans and labels while ignoring the hierarchical structure.\footnote{For the evaluation of tree structures, other approaches also exist that take into account the complete paths within the tree, e.g., the leaf-ancestor metric or dependency-based metrics, cf. Rehbein and van Genabith (2007). Although
For example, an NP is considered correct if it spans across the correct tokens, independently of the presence or absence of intervening nodes like adjective phrases, etc. The same also applies to other multi-level annotations, e.g., several stacked entities. Hence, the traditional evaluation from Algorithm 1 works just the same for nested spans as for flat annotations.

The identification of the fine-grained error types, specifically BEs and LBEs, in nested structures is more complicated because it is not always clear which spans should be compared with each other. While the classification is likely no problem for humans in most of the cases, an algorithm will sometimes only approximate the optimal match of system and target spans if it shouldn’t become too complex or slow. Here, two practical decisions are made:

1. It is known that systems are usually more accurate at identifying shorter spans compared to longer ones (cf. Bastings and Sima’an (2014) on constituency parsers). Therefore, in each step, the algorithm starts with the shortest span to speed up the search for the correct match of system and target annotation.

2. If one span can be matched to two (or more) other spans, the most similar one is considered first. Similarity, here, is defined as the maximum number of shared tokens and the fewest differing tokens. If multiple spans are equally similar, the shortest one is chosen. If multiple spans are still equally similar, the first one in the input is taken, which corresponds to the left-most one if sentences are read from left to right.

Based on the previous considerations, Algorithm 2 identifies the fine-grained error types from Section 2.1 in flat and hierarchical spans. The resulting error counts can be used to calculate precision, recall, and F1-score as detailed in Section 2.2. Table 2 in the next section shows an example of the algorithm’s output. A reference implementation of the algorithm as well as the data sets and detailed results from Section 4 can be found in this paper’s repository at https://github.com/rubcompling/FairEval.

4. Examples

To illustrate the application and results of the new evaluation algorithm, in this section, it is applied to different tasks that require the identification of labeled spans: NER, chunking, and topological field parsing.

1. Named Entity Recognition

   - **Annotation**: Named entities are phrases that refer to entities such as people or places by means of a proper name (Tjong Kim Sang and De Meulder, 2003). The annotation is sparse, i.e., the majority of tokens do not receive a label. Multi-level annotations are possible but not considered here.

   - **NLP tool**: The NER component of the Stanza pipeline (Qi et al., 2020) with the germeval2014 model.

   - **Data**: The test partition of the GermEval 2014 data set (Benikova et al., 2014). Since Stanza does not support multi-level annotation, only top-level entities from the four main classes are evaluated, yielding 6.178 named entities.

2. Chunking

   - **Annotation**: Chunks are non-recursive, non-overlapping constituents from a sentence’s parse tree (Sang and Buchholz, 2000). Contrary to NER, most tokens receive a label. The annotation is non-hierarchical per definition.

   - **NLP tool**: The neural sequence labeling tool NCRF++ (Yang and Zhang, 2018) with a model from Ortmann (2021b). The model was trained on the German newspaper corpus TüBa-D/Z (Telljohann et al., 2017): 80% training, 10% development data with characters, tokens, and POS tags as features and pre-trained word embeddings.

   - **Data**: The remaining 10% of the TüBa-D/Z corpus with 101.304 chunks of 16 different types.

3. Topological Field Parsing

   - **Annotation**: Topological fields are linear syntactic structures on the clause level of German sentences. Fields can be understood to form a tree structure, i.e., the annotation is hierarchical, and most tokens receive a label.

   - **NLP tool**: The unlexicalized Berkeley parser (Petrov et al., 2006) with a model from Ortmann (2020). It was trained on 80% of the TüBa-D/Z corpus (Telljohann et al., 2017).

   - **Data**: 10% of the TüBa-D/Z corpus with 63.824 fields of 13 different types.

---

*Stanza version 1.2, https://stanfordnlp.github.io/stanza/https://github.com/jiesutd/NCRFpp

*Release 11.0, chunked version

*For an overview of the topological field model, see, e.g., Cheung and Penn (2009), Ortmann (2020), or Wöllstein (2018) in German.

*https://github.com/slavpetrov/berkeleyparser

*Release 11.0, CoNLL-U Plus version

---

these metrics are more robust against differences in annotation schemes, the PARSEVAL metric (Black et al., 1997) is still commonly used for parser evaluation.
The results for traditional vs. fair evaluation of the three annotation tasks are displayed in Table 1. \( F_1 \)-scores differ between 1–3.7 percentage points. The largest difference is observed for the recognition of named entities, while the difference is smallest for chunks. Except for NER, recall values differ slightly more between evaluation methods than precision. With respect to the labels, the largest differences are found for entities of type ORG, adjective and foreign language chunks, and coordination and post-fields.

Figure 1 shows the distribution of error types for the three annotation tasks according to traditional vs. fair evaluation. For NER and chunking, the traditional evaluation identifies 44% of the errors as FP and 56% as FN, while for topological field parsing, FPs are more frequent with 54% compared to 46% FNs. However, when the more fine-grained error types are considered, the rate of actual false positives and negatives shrinks substantially. The highest proportion of actual FNs is observed for the sparse NER annotation and the highest proportion of actual FPs for the hierarchical fields. For chunking, actual FPs and FNs together make up only 2% of all errors.

On the other hand, boundary errors, which traditionally count as two errors (1 \( FP \) and 1 \( FN \)), make up between 14% (NER) and 59% (chunking) of the errors. In most of these cases, the system annotation includes the target span (57%–73%) or vice versa (27%–42%). Errors of type BEo are extremely rare. Interestingly, labeling errors occur especially for the sparse named entities: 30% of the errors are due to entities that were recognized correctly but assigned the wrong label. Labeling-boundary errors are more frequent for chunking (34%).

Table 1: Results for traditional vs. fair evaluation of the different annotation tasks in percent.

|                | Precision | Recall | \( F_1 \) |
|----------------|-----------|--------|-----------|
| **NER**        |           |        |           |
| Trad.          | 86.66     | 83.51  | 85.05     |
| Fair           | 90.42     | 87.23  | 88.80     |
| **Chunks**     |           |        |           |
| Trad.          | 97.20     | 96.39  | 96.79     |
| Fair           | 97.86     | 97.86  | 97.86     |
| **Topol. Fields** |         |        |           |
| Trad.          | 93.41     | 94.27  | 93.84     |
| Fair           | 94.78     | 95.92  | 95.35     |

The distribution of error types for the three annotation tasks according to traditional vs. fair evaluation.

Figure 2: Confusion matrices for the (main) labels of each annotation task. Only errors are included, i.e., the diagonal displays boundary errors. False positives and negatives are shown in the bottom row and the rightmost column, respectively. The remaining cells represent labeling and labeling-boundary errors.
Future studies should consider using the presented algorithm to optimize systems for sensible metrics and gain more insight into their actual weaknesses. However, the comparison has also shown that $F_1$-scores are higher according to the new evaluation method because labeling and boundary errors are no longer multiply penalized. Since the objective of the algorithm is not to make systems ‘look better’, results that are gained in this way should be reported alongside established evaluation metrics to ensure comparability.

6. Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 232722074 – SFB 1102 (Project C6). I am grateful to Stefanie Dipper, Adam Roussel, and the anonymous reviewers for their very helpful comments.

7. Bibliographical References

Bastings, J. and Sima’an, K. (2014). All fragments count in parser evaluation. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), pages 78–82, Reykjavik, Iceland, May. European Languages Resources Association (ELRA).

Beniška, D., Biemann, C., Kisselw, M., and Padó, S. (2014). Germeval 2014 named entity recognition shared task: Companion paper.

Black, E., Abney, S. P., Flickinger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingría, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M. P., Roukos, S., Santorini, B., and Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of english grammars. In Proceedings DARPA Speech and Natural Language Workshop, pages 306–311, Pacific Grove, CA.

Braşoveanu, A. M., Rizzo, G., Kuntschik, P., Weichselbraun, A., and Nixon, L. J. (2018). Framing named entity linking error types. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).

Cheung, J. C. K. and Penn, G. (2009). Topological field parsing of German. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 -

| Label | TP | FP | LE | BE_L | BE_M | BE_B | LBE | FN | Precision | Recall | $F_1$ |
|-------|----|----|----|------|------|------|-----|----|-----------|--------|------|
| LOC   | 2132 | 81 | 56 | 29   | 28   | 0    | 57  | 40 | 98         | 93.12  | 92.43 | 92.78 |
| ORG   | 1002 | 87 | 76 | 16   | 27   | 0    | 43  | 48 | 167        | 85.46  | 80.00 | 82.64 |
| OTH   | 473  | 48 | 89 | 15   | 26   | 3    | 44  | 44 | 142        | 77.60  | 67.24 | 72.05 |
| PER   | 1552 | 37 | 31 | 11   | 25   | 0    | 36  | 23 | 55         | 94.98  | 93.95 | 94.46 |
| Overall | 5159 | 253 | 252 | 71   | 106  | 3    | 180 | 155 | 462        | 90.42  | 87.23 | 88.80 |

Table 2: Raw frequencies of $TP$s and errors in the NER annotation per label and overall as output by Algorithm 2. In addition, the rightmost columns show fair precision, recall, and $F_1$ values for individual labels.
