An Untold Story of Preprocessing Task Evaluation: An Alignment-based Joint Evaluation Approach

Eunkyul Leah Jo[1,2*] Angela Yoonseo Park[3*] Grace Zhang[1*]
Izia Xiaoxiao Wang[4*] Junrui Wang[3*] MingJia Mao[3*] Jungyeul Park[3]
[1]Department of Computer Science, The University of British Columbia, Canada
[2]Faculté des Sciences et Ingénierie, Sorbonne Université, France
[3]Department of Linguistics, The University of British Columbia, Canada
[4]Linguistik Zentrum Zürich, Universität Zürich, Schweiz
jungyeul@mail.ubc.ca

Abstract

A preprocessing task such as tokenization and sentence boundary detection (SBD) has commonly been considered as NLP challenges that have already been solved. This perception is due to their generally good performance and the presence of pre-tokenized data. However, it's important to note that the low error rates of current methods are mainly specific to certain tasks, and rule-based tokenization can be difficult to use across different systems. Despite being subtle, these limitations are significant in the context of the NLP pipeline. In this paper, we introduce a novel evaluation algorithm for the preprocessing task, including both tokenization and SBD results. This algorithm aims to enhance the reliability of evaluations by reevaluating the counts of true positive cases for F1 measures in both preprocessing tasks jointly. It achieves this through an alignment-based approach inspired by sentence and word alignments used in machine translation. Our evaluation algorithm not only allows for precise counting of true positive tokens and sentence boundaries but also combines these two evaluation tasks into a single organized pipeline. To illustrate and clarify the intricacies of this calculation and integration, we provide detailed pseudo-code configurations for implementation. Additionally, we offer empirical evidence demonstrating how sentence and word alignment can improve evaluation reliability and present case studies to further support our approach.

Keywords: Tokenization, sentence boundary detection (SBD), alignment-based evaluation algorithm

1. Introduction

Recognizing and adjusting for errors is crucial to achieve dependable outcomes in most computer-based language tasks. However, this step often does not get the attention it deserves because there is a common practice of accepting some level of errors. This lack of focus becomes even more problematic during the initial stages of text processing, particularly when dealing with tasks like tokenization and sentence boundary detection (SBD). Given that most commonly used preprocessing methods are generally recognized as well-established and can be easily reused, people tend to think that challenges like tokenization and SBD have already been completely solved. However, what often goes unnoticed is the fact that there has been a longstanding lack of attention to the evaluation process, which has been underestimated despite the apparent success of these tools.

In light of these points, there have been efforts in the literature to address and improve the limitations of tokenization and SBD. These efforts involve extensive comparisons and detailed reevaluations, as seen in studies like Dridan and Oepen (2012) for tokenization and Read et al. (2012) for SBD. While their reevaluation results are comprehensive and convincing, they often lack clear and adaptable configurations that others can use for reimplementation.

When you examine these efforts side by side, it becomes evident that the tasks of sentence and word segmentation are closely related and complement each other more than they are typically considered as separate steps. In response to this, some innovative sequence labeling methods, as discussed in Evang et al. (2013), have been proposed, showing promising results in terms of reproducibility across various domains. However, there are still noticeable challenges related to sentence boundaries that may remain unresolved in the evaluation process.

In this paper, we introduce a new evaluation algorithm for preprocessing tasks, which we call the **jp-algorithm** (a **j**oint evaluation algorithm for **p**reprocessing). This algorithm utilizes an alignment-based approach to enhance the accuracy of evaluation by addressing the previously mentioned issues, which we refer to as "mismatches" in preprocessing. Drawing inspiration from alignment techniques used in machine translation (MT), our algorithm simplifies the extensive evaluation process for both preprocessing tasks, making it more efficient and accurate. This integration allows us to reevaluate key edge cases, refining precision and recall for F1 measures in the final evaluation results. These calculations are

[*]Equally contributed authors.

based on adjusted true positive counts after re-alignment.

To emphasize the importance of reevaluating pre-processing tasks, we begin by addressing certain aspects that have been overlooked in tokenization and SBD. We showcase the effectiveness of our alignment method and illustrate the core components of our approach with adjusted examples to highlight the evaluation challenges faced in previous works (Section 2). To demonstrate the implementation of the jp-algorithm, we provide pseudocode and detailed explanations (Section 3). To confirm the reliability and usefulness of our algorithm, we carry out case studies. In these case studies (Section 4), we assess the results obtained from various existing preprocessing tools. This evaluation helps us establish the effectiveness of our approach before we summarize our findings and conclude the paper (Section 5).

2. Mismatches in Preprocessing

2.1. Tokenization

The primary challenge in tokenization, as demonstrated by several methods, revolves around handling the ambiguity related to sentence periods and contractions (Dridan and Oepen, 2012). For instance, consider the sentence *When No. 1 Isn't the Best*[1]. Different tokenization schemes, such as `tok.sed`[2], `Moses` (Koehn et al., 2007), and `CoreNLP` (Manning et al., 2014), tokenize the contraction and abbreviation of common words in this sentence differently.

`tok.sed`	When No␣. 1 is␣n't the Best
`Moses`	When No. 1 isn␣'t the Best
`CoreNLP`	When No. 1 is␣n't the Best

where ␣ is a symbol for a token delimiter. Typically, periods are treated as individual tokens in text processing. However, due to the common use of word-final periods in conjunction with abbreviations, acronyms, and named entities, there is a lack of consistency in how these cases are handled. This inconsistency results in variations in tokenization approaches for these words. While there are various tokenization methods available, `CoreNLP` results suggest that, in practice, the terminal leaf nodes in the Penn treebank (Marcus et al., 1993; Taylor et al., 2003) have become a widely accepted standard tokenization scheme for English.

[1] https://www.washingtonpost.com/archive/sports/2004/05/26/when-no-1-isnt-the-best/fa34156f-881a-4181-b0fe-4447e2f36f0e/

[2] ftp://ftp.cis.upenn.edu/pub/treebank/public_html/tokenization.html

2.2. Sentence boundary detection

The task of sentence boundary detection involves identifying the points where sentences start and end. In many prior studies (Palmer and Hearst, 1997; Reynar and Ratnaparkhi, 1997; Kiss and Strunk, 2006; Gillick, 2009; Lu and Ng, 2010), sentence boundary disambiguation has been treated as a classification problem. It is important to note that sentence boundary disambiguation is distinct from sentence boundary detection. The former requires the use of punctuation marks for classification, while the latter does not necessarily rely on punctuation to determine sentence boundaries. A significant challenge in sentence boundary disambiguation arises when dealing with text that lacks consistent punctuation marks. To tackle this issue, Evang et al. (2013) proposed a method involving supervised character-level sequence labeling for both sentence and word segmentation. Additionally, Lim and Park (2024) annotated a dataset with rich linguistic information to improve sentence boundary detection. However, many common sentence boundary detection systems, such as `splitta` (Gillick, 2009), `CoreNLP` (Manning et al., 2014), and `Elephant` (Evang et al., 2013), often fail to correctly identify sentence boundaries in cases where there are no punctuation marks between two sentences, even when their methods are not designed to rely on punctuation marks. For instance, this issue can be observed between *... session* and *I ...* in the following text extracted from the Europarl parallel corpus (Koehn, 2005):

```
Opening of the session I declare resumed
the 2000-2001 session of the European
Parliament.
```

Additionally, textual content often exhibits improper formatting, including issues such as missing or incorrect punctuation, inconsistent casing, and the presence of unnecessary symbols. For instance, in social media text, individuals may write in all uppercase or with incorrect punctuation and casing, or entirely omit punctuation. In the domain of automatic speech recognition (ASR) where there are no punctuation marks, several SBD-related works have been proposed (Treviso, Shulby, and Aluísio, 2017; González-Gallardo and Torres-Moreno, 2018). Punctuation restoration in ASR has also been explored (Fu et al., 2021; Alam et al., 2020; Poláček et al., 2023).

2.3. Evaluation

Methods for evaluating sentence boundary disambiguation and tokenization can be broadly categorized into three main approaches:

Accuracy and error rate This method assesses how accurately the boundaries are detected or tokens are separated, and quantifies the rate of errors made.

Precision and recall for F1 Measure This approach focuses on measuring the precision (how many of the detected boundaries or tokens are correct) and recall (how many of the actual boundaries or tokens are detected). The F1 measure is then calculated to balance these two factors.

Levenshtein distance This method involves calculating the Levenshtein distance, which represents the number of edits (insertions, deletions, or substitutions) needed to transform the detected boundaries or tokens into the correct ones.

These three evaluation approaches help assess the performance and accuracy of sentence boundary detection and tokenization methods.
In previous SBD tasks, such as the one mentioned in Dridan and Oepen (2012), each character position is considered a potential boundary point. If a gold-standard boundary is missed, it is counted as a false negative. This imbalance between negative and positive cases makes precision and recall for F1 evaluation metrics more informative than accuracy or error rate metrics. To address these issues, character-level methods aim to facilitate the integration of evaluations between the two tasks, as proposed by Evang et al. (2013). However, even at the character level, the imbalance problem persists, especially in cases where there are partial matches, which can lead to miscounted true positive cases. For example, in Figure 1, within the context of tokenization, a perfect alignment between the system output and the reference gold standard is observed, yielding an F1 measure of 1.0 (100%). However, when it comes to sentence boundary detection (SBD), there's a difference. Using previous methods, *Click here* and *To view it.* are identified as two separate sentences in SBD, with two "S" labels marking sentence boundaries under *Click* and *To*, as shown in Figure 1. However, the first sentence boundary marked by *Click* is essentially a partial match, while the second part is separated into another sentence by the second sentence boundary marked by *To*.

```
      Click here To view it.
sys   SIIIIOTIIIOSIOTIIIOTIT
gold  SIIIIOTIIIOTIOTIIIOTIT
```

Figure 1: BIO-style evaluation in previous work where partial matched sentences (e.g *Click here*) could be considered as `true positive`.

The proposed evaluation using the alignment algorithm in this paper aims to prevent such partially matched sentences from being incorrectly counted during evaluation. This is achieved by incorporating sentence alignment methods borrowed from machine translation (MT), addressing various preprocessing issues along the way.

3. Alignment-based Evaluation

We begin by employing a fundamental algorithm that leverages sentence and word alignment techniques typically used in MT. In MT and other cross-language applications, sentence and word alignment have been crucial sub-tasks. They are often employed when working with parallel corpora, where sentences in one language need to be aligned with their translations in another language. This necessity arises from the fact that when two sentences are direct translations of each other, they can differ significantly in terms of word order, length, and structure. Therefore, to enable downstream cross-language tasks, it's often essential to perform comprehensive structural adjustments by establishing sentence and word alignments between the source and target languages. These alignment methods, used to address inter-lingual differences, shed light on the challenges we discuss in Section 2 regarding evaluation.
However, when dealing with monolingual inputs from both the system and gold results, there are no inter-lingual differences between sequences on each side (L and R). Instead, these sequences are essentially identical sentences, differing only in terms of token and sentence boundaries. Nevertheless, when evaluating tokenization and SBD tasks, they share a commonality with cross-language tasks: for a corresponding sentence pair from the system and gold results, even if they are identical at the character level, they can still vary in length across lines due to differences in tokenization and SBD results, as demonstrated in the `system result` and `gold file` in Figure 2. This is where the alignment approach used in MT becomes useful in the evaluation algorithm we propose. Figure 2 provides a simplified example of how our proposed evaluation, facilitated by the alignment process, would resolve the partial match issue discussed in the previous section.
While Figure 2 presents sentence alignment for the evaluation of sentence boundary detection, a same approach will be employed for tokenization evaluation, utilizing word alignment.

3.1. Algorithm description

Notations To describe our proposed algorithms more concisely and clearly, we'll use the following notations: $\mathcal{L}$ and $\mathcal{R}$ represent the system output and the gold files for preprocessing, respectively. We assume `len`($\mathcal{L}_{\not\sqcup}$) `==` `len`($\mathcal{R}_{\not\sqcup}$), where $\mathcal{L}_{\not\sqcup}$ and

input file:

```
Click here To view it. He makes some good ⊓
observations on a few of the picture's. ⊓
```

system result:

```
Click here ⊓
To view it⊔. ⊓
He makes some good observations on a few of the picture⊔'s⊔. ⊓
```

gold file:

```
Click here To view it⊔. ⊓
He makes some good observations on a few of the picture⊔'s⊔. ⊓
```

sentence-aligned result for preprocessing evaluation:

`Click here ∼∼∼ To view it⊔. ⊓`	`Click here To view it⊔. ⊓`
`He makes some good observations on a few of the picture⊔'s⊔. ⊓`	`He makes some good observations on a few of the picture⊔'s⊔. ⊓`

Figure 2: Example of intermediate results of the evaluation by alignment algorithm where *Click here* and *To view it.* In the system result are the realigned, produced by merging after the implementation of sentence alignment: ⊔ is a symbol for tokenization, ⊓ is a symbol for SBD, and $\sim\sim\sim$ is a symbol for merged sentences by sentence alignment.

$\mathcal{R}_{\not\sqcup}$ are sequences of characters with all spaces removed from $\mathcal{L}$ and $\mathcal{R}$. L and R denote the current sentence pair from the system output and the gold files. The following notations are also used interchangeably for both L and R:

$\mathcal{C}_{sb}(\mathcal{L})$ and $\mathcal{C}_{tk}(\mathcal{L})$: These represent the total number of sentence boundaries (sb) and the total number of tokens (tk) in $\mathcal{L}$.

$\texttt{TP}_{\texttt{sb}}$ and $\texttt{TP}_{\texttt{tk}}$: These denote the number of true positives (tp) for sentence boundaries and tokens, respectively.

$L_{\sqcup}$: This represents L where spaces between tokens have not been removed.

$L_{\not\sqcup}$: This represents L where spaces between tokens have been removed.

L_i: It stands for the *i*th token in $L_{\sqcup}$.

Evaluation measures To calculate an F1 score for SBD, we need to have the total counts of sentence boundaries in both the system output and the gold standard ($\mathcal{C}_{sb}(\mathcal{L})$ and $\mathcal{C}_{sb}(\mathcal{R})$, respectively), as well as the number of true positives for sentence boundaries ($\texttt{TP}_{\texttt{sb}}$). Precision and recall, for example, are defined as follows, considering sentence boundary detection:

$$\begin{aligned}\text{precision} &= \frac{\text{relevant \# of sb} \cap \text{retrieved \# of sb}}{\text{retrieved \# of sb}} \\ &= \frac{\mathcal{C}_{sb}(\mathcal{L}) \cap \mathcal{C}_{sb}(\mathcal{R})}{\mathcal{C}_{sb}(\mathcal{L})} = \frac{\texttt{TP}_{\texttt{sb}}}{\mathcal{C}_{sb}(\mathcal{L})} \\ \text{recall} &= \frac{\text{relevant \# of sb} \cap \text{retrieved \# of sb}}{\text{relevant \# of sb}} \\ &= \frac{\mathcal{C}_{sb}(\mathcal{L}) \cap \mathcal{C}_{sb}(\mathcal{R})}{\mathcal{C}_{sb}(\mathcal{R})} = \frac{\texttt{TP}_{\texttt{sb}}}{\mathcal{C}_{sb}(\mathcal{R})}\end{aligned}$$

To apply in the algorithm, 'relevant # of sb ∩ retrieved # of sb' represents $\texttt{TP}_{\texttt{sb}}$, 'retrieved # of sb' is $\mathcal{C}_{sb}(\mathcal{L})$ by a system result, and 'relevant # of sb' is $\mathcal{C}_{sb}(\mathcal{R})$ by a gold file. The same precision and recall can be defined for tokenization using $\mathcal{C}_{tk}(\mathcal{L})$, $\mathcal{C}_{tk}(\mathcal{R})$ and $\texttt{TP}_{\texttt{tk}}$.

Sentence alignment for SBD evaluation In MT, sentence alignment involves identifying corresponding sentences in two or more languages and linking sentences from one language to their corresponding counterparts in another. Sentence alignment has been a subject of study for many years, leading to the development of various algorithms. Early research in this area relied on statistical methods that used bilingual corpora to create models capturing the lexical equivalence between words in different languages. For instance, the Gale-Church algorithm, based on sentence length, was one such approach (Gale and Church, 1993). `Bleualign` introduced a more advanced iterative bootstrapping approach building on length-based methods (Sennrich and Volk, 2011). Earlier approaches also aimed to enhance sentence alignment methodologies by incorporating lexical correspondences, as seen in `hunalign` (Varga et al., 2005). Some attempts involved the integration of linguistic knowledge, heuristics, and various scoring methods to improve efficiency, as demonstrated by `vecalign` (Thompson and Koehn, 2019).

The proposed alignment algorithm processes $\mathcal{L}$ and $\mathcal{R}$ and increments the $\texttt{TP}_{sb}$ count if $L_{\not\sqcup}$ is equal to $R_{\not\sqcup}$. However, when they do not match, the algorithm merges sentences in L' and R' for mismatched sentence pairs until L' and R' are identical (practically, this is done until the following $L_{\not\sqcup}$ and the following $R_{\not\sqcup}$ are identical instead of verifying L' and R' being identical). After this merging process, the algorithm continues to increment $\texttt{TP}_{sb}$ if $L_{\not\sqcup}$ is equal to $R_{\not\sqcup}$.

Word alignment for tokenization evaluation In MT, word alignment methodologies are employed to establish correspondences between words in one language and their direct translations in another. Widely used IBM models, along with tools like `giza++` (Brown et al., 1993; Och and Ney, 2000) or `BerkeleyAligner` (Liang et al., 2006; DeNero and Klein, 2007), are capable of aligning words.
In the context of our evaluation algorithm, we increment the $\texttt{TP}_{tk}$ count in the word-aligned $L_{\sqcup}$ and $R_{\sqcup}$ pair if L_i is equal to R_j, where L_i and R_j represent the *i*-th and *j*-th tokens in $L_{\sqcup}$ and $R_{\sqcup}$, respectively. While IBM word alignment can be adapted for token alignment within our evaluation approach, our focus lies in aligning tokens within the specific sentence pair expected to be identical, rather than traversing the entire document as IBM models are designed to find the proper lexical translation pair between sentence pairs. For our evaluation algorithm, token alignment only needs to traverse each aligned sentence pair within $L_{\sqcup}$ and $R_{\sqcup}$, directly assessing token equivalence within the corresponding sentence pair.

Summary Drawing an analogy from sentence and word alignment, we introduced an alignment-based algorithm that significantly enhances the efficiency of evaluating sentence boundary detection (SBD) and tokenization results. This algorithm allows us to evaluate both tasks in a single pass through the input. Through the alignment of sentences and tokens, we can streamline the evaluation process, resulting in more precise counts of true positive cases for both tokenization and SBD results.

3.2. Pseudo-codes

Alignment algorithm The alignment algorithm is employed in two key scenarios: (a) to align sentences between the system and the gold results, and (b) to align tokens within the current sentence pair. Once the alignment is established, we can conduct a side-by-side comparison between the system and gold results. This comparison allows us to discern and integrate all the intertwined matched and mismatched cases between sentence boundary detection (SBD) and tokenization. Algorithm 1 provides a generic representation of the alignment algorithm, which we can apply to both sentence boundary and tokenization evaluation. The inner `while` statement, where mismatches are addressed, is part of an iterative process controlled by the outer `while` loop, which iterates through each element. The time complexity of this algorithm is $\mathcal{O}(N + N')$, where N and N' represent the lengths of the left and right sentences (L and R), respectively. Additionally, this algorithm is applicable to tokens.

Algorithm 1: Alignment algorithm

```
while $\mathcal{L}$ and $\mathcal{R}$ do
    if $L_i$ == $R_j$ then
        TP++;
    else
        $L'$ += $L_i$;
        $R'$ += $R_j$;
        while $L_{i+1} \neq R_{j+1}$ do
            if len($L'$) > len($R'$) then
                $j$++;
                $R'$ += $R_j$;
            else
                $i$++;
                $L'$ += $L_i$;
            end
        end
    end
    $i$++;
    $j$++;
end
```

Implementations We provide two comprehensive implementations of the algorithm: one as a basic algorithm (Algorithm 2) and another as a joint algorithm (Algorithm 3). In Algorithm 2, the evaluation procedure distinguishes between sentence boundary detection (SBD) and tokenization. Its time complexity is $\mathcal{O}(2N + 4NM)$, where N represents the number of sentences and M denotes the number of tokens in each sentence. We assume that $N \simeq N'$ and $M \simeq M'$ for the left (L) and right (R) sentences, respectively. This version simplifies notations by removing indexes i and j. During the alignment of sentences, we obtain `TPsb`. Subsequently, in the second outer `while` loop, we obtain `TPtk` during word alignment, where $L'_{i,i'}$ represents the i'-th token in the i-th sentence of L'. Both iteration processes resemble those in Algorithm 1. In contrast, Algorithm 3 processes SBD and tokenization jointly, reducing the time complexity to $\mathcal{O}(4NM)$. We maintain the assumption that $N \simeq N'$ and $M \simeq M'$ for L and R.

Examples Figure 3 provides examples to illustrate the jp-algorithm in different scenarios. In the first case, where $L_{\not\sqcup}$ matches $R_{\not\sqcup}$, we obtain true positive instances for sentence boundaries, denoted as $\texttt{TP}_{\texttt{sb}}$. The second case depicts conditions where true positive sentence boundaries are not present. In such instances, the algorithm proceeds to build L' and R' if a sentence is segmented and requires merging (which we consider sentence alignment). After merging, these sentences should align correctly, and we identify the true positives among these pairs. To count the

Algorithm 2: Basic algorithm

```
Data: $\mathcal{L}$, $\mathcal{R}$
Result: $\mathcal{C}_{sb}(\mathcal{L})$, $\mathcal{C}_{tk}(\mathcal{L})$, $\mathcal{C}_{sb}(\mathcal{R})$, $\mathcal{C}_{tk}(\mathcal{R})$, $\texttt{TP}_{\texttt{sb}}$,
        $\texttt{TP}_{\texttt{tk}}$
Obtain $\mathcal{C}_{sb}(\mathcal{L})$, $\mathcal{C}_{tk}(\mathcal{L})$ from $\mathcal{L}$ and $\mathcal{C}_{sb}(\mathcal{R})$,
 $\mathcal{C}_{tk}(\mathcal{R})$ from $\mathcal{R}$;
while $\mathcal{L}$ and $\mathcal{R}$ do
    /** Obtain $\texttt{TP}_{\texttt{sb}}$ **/
    if $L_{i\not\sqcup}$ == $R_{j\not\sqcup}$ then
        $\texttt{TP}_{\texttt{sb}}$++;
        $\mathcal{L}'$, $\mathcal{R}'$ += $L_i, R_j$;
    else
        while $L_{i+1\not\sqcup} \neq R_{j+1\not\sqcup}$ do
            $L'$ += $L_i$;
            $R'$ += $R_j$;
        end
        $\mathcal{L}'$, $\mathcal{R}'$ += $L', R'$
    end
end
while $\mathcal{L}'$ and $\mathcal{R}'$ do
    /** Obtain $\texttt{TP}_{\texttt{tk}}$ **/ if $L_{i\sqcup} == R_{j\sqcup}$ then
        $\texttt{TP}_{\texttt{tk}}$ += len($Li\sqcup$);
    else
        while $L_{i\sqcup}$ and $R_{j\sqcup}$ do
            $\texttt{TP}_{\texttt{tk}}$++ if $L_{i,i'} == R_{j,j'}$;
        end
    end
end
```

Algorithm 3: Joint algorithm

```
Data: $\mathcal{L}$, $\mathcal{R}$
Result: $\mathcal{C}_{sb}(\mathcal{L})$, $\mathcal{C}_{tk}(\mathcal{L})$, $\mathcal{C}_{sb}(\mathcal{R})$, $\mathcal{C}_{tk}(\mathcal{R})$, $\texttt{TP}_{\texttt{sb}}$,
        $\texttt{TP}_{\texttt{tk}}$
Obtain $\mathcal{C}_{sb}(\mathcal{L})$, $\mathcal{C}_{tk}(\mathcal{L})$ from $\mathcal{L}$ and $\mathcal{C}_{sb}(\mathcal{R})$,
 $\mathcal{C}_{tk}(\mathcal{R})$ from $\mathcal{R}$;
while $\mathcal{L}$ and $\mathcal{R}$ do
    /** Obtain $\texttt{TP}_{\texttt{sb}}$ **/
    if $L_{i\not\sqcup}$ == $R_{j\not\sqcup}$ then
        $\texttt{TP}_{\texttt{sb}}$++;
        /** Obtain $\texttt{TP}_{\texttt{tk}}$ **/
        if $L_{i\sqcup} == R_{j\sqcup}$ then
            $\texttt{TP}_{\texttt{tk}}$ += len($L$);
        else
            while $L_{i\sqcup}$ and $R_{j\sqcup}$ do
                $\texttt{TP}_{\texttt{tk}}$++ if $L_{i,i'} == R_{j,j'}$;
            end
        end
    else
        while $L_{i+1\not\sqcup} \neq R_{j+1\not\sqcup}$ do
            $L'$ += $L_i$;
            $R'$ += $R_j$;
        end
        while $L'_{\sqcup}$ and $R'_{\sqcup}$ do
            $\texttt{TP}_{\texttt{tk}}$++ if $L'_{i'} == R'_{j'}$;
        end
    end
end
```

number of true positives for tokens between L and R, we obtain $\texttt{TP}_{\texttt{tk}}$ if $L_i == R_j$. Figure 4 provides a detailed breakdown of how to evaluate and count the number of true positives ($\texttt{TP}$) for tokens. This is done using the same alignment method as previously explained for sentence alignment.

Discussion The length-based sentence alignment algorithms, like the one described by Gale and Church (1993, p.83), typically consider matches in the ratios of 1:0, 0:1, 1:1, 2:1, 1:2, and 2:2. However, we need to account for cases where the system segments a sentence into more than two sentences or where gold sentence boundaries are segmented into more than two sentences. In other words, even after using sentence alignment, the segmented results of $L_{\not\sqcup}$ and $R_{\not\sqcup}$ may still differ from each other if we apply pre-existing sentence alignment algorithms from MT. To address this, we accumulate and merge L and R together until their characters match, resulting in $L' == R'$ instead of using MT's alignment algorithm. As described in Algorithm 3, the 'else' condition (when $L_{\not\sqcup} \neq R_{\not\sqcup}$) entails aggregating L' and R' to form a matched sentence pair between L and R. This process allows for the accumulation of pairs such as m:1, 1:n, or m:n sentence segments.

3.3. Proof

We refer a perfect sentence alignment pair to $L_{\not\sqcup} = R_{\not\sqcup}$. We also consider $L'_{\not\sqcup} = R'_{\not\sqcup}$ as another perfect matched pair by accumulating unmatched lines of L and R if $L_{\not\sqcup} \neq R_{\not\sqcup}$. We investigate such alignment implementations as it can achieve L' and R' where ${L_{\not\sqcup}}' == {R_{\not\sqcup}}'$.

3.3.1. Soundness

The algorithm is sound when each premise is true and can return a true answer, resulting in a tautology. By the Soundness Theorem, the Algorithm A is sound through the following three cases. Let $\mathcal{L}, \mathcal{R}, L', R' \in A$.

Case 1 *Algorithm 3 is sound if* $L_{\not\sqcup} \vdash L$ *implies* $L_{\not\sqcup} \models_{\textit{taut}} L$ *and* $R_{\not\sqcup} \vdash R$ *implies* $R_{\not\sqcup} \models_{\textit{taut}} R$.

Case 2 *Algorithm 3 is sound if* $(L_{\not\sqcup}\&L_{\sqcup}) \vdash L$ *implies* $(L_{\not\sqcup}\&L_{\sqcup}) \models_{\textit{taut}} L$ *and* $(R_{\not\sqcup}\&R_{\sqcup}) \vdash R$ *implies* $(R_{\not\sqcup}\&R_{\sqcup}) \models_{\textit{taut}} R$.

Case 3 *Algorithm 3 is sound if* $L_{\not\sqcup} \vdash L'$ *implies* $L_{\not\sqcup} \models_{\textit{taut}} L'$ *and* $R_{\not\sqcup} \vdash R'$ *implies* $R_{\not\sqcup} \models_{\textit{taut}} R'$.

3.3.2. Correctness

To prove the correctness of $L' == R'$, let $L' = L_i...L_n$ and $R' = R_j...R_m$ for some $i, j, n, m \in \mathbb{Z}^{\geq 0}$.

	$\mathcal{L}$ (system)	$\mathcal{R}$ (gold)	
$L_{\not\sqcap}$ == $R_{\not\sqcap}$	`When No. 1 Isn 't the Best ⊓`	`When No. 1 Is n't the Best ⊓`	TP
$L_{\not\sqcap} \neq R_{\not\sqcap}$ (before alignment)	`Mike McConnell 07/06/2000 14:57 John , Hello from South America . ⊓`	`Mike McConnell ⊓` `07/06/2000 14:57 ⊓` `John , Hello from South America . ⊓`	
(after alignment)	`Mike McConnell 07/06/2000 14:57 John , Hello from South America . ⊓`	`Mike McConnell ∼∼∼ 07/06/2000 14:57 ∼∼∼ John , Hello from South America. ⊓`	-

Figure 3: Examples with the jp-algorithm in Algorithm 3 for `true positive` of sentences. If not, merging sentence boundaries.

$\mathcal{L}$ (system)	`When`	`No.`	`1`	`Isn ∼∼∼ 't`	`the`	`Best`
$\mathcal{R}$ (gold)	`When`	`No.`	`1`	`Is ∼∼∼ n't`	`the`	`Best`
	`TP`	`TP`	`TP`	-	`TP`	`TP`

Figure 4: Examples for `true positive` of tokens. If not, merging tokens.

Case 4 $L_{i+1} == R_{j+1}$. *For any i and j, we know that $i+1, j+1 \geq 0$ such that $i, j \in \mathbb{Z}^{\geq 0}$. Hence, for L_{i+1} and R_{j+1} to have the same value, i and j will remain in the domain. This will allow L_{i+1} and R_{j+1} to accumulate together.*

Case 5 $L_{n+1} == R_{m+1}$. *We know for any n and m, $n+1$ and $m+1$ are positives. Since $n, m \in \mathbb{Z}^{\geq 0}$, L_{n+1} and R_{m+1} are able to accumulate together.*

Therefore, the statement $L' == R'$ is true.

4. Experiments and Results

4.1. Case study on English corpora

For our case studies, we conduct preprocessing on five raw input files sourced from English Universal Dependencies (Nivre et al., 2016, 2020). These files include: (1) Universal Dependencies syntax annotations from the `GUM` corpus. (2) A multilingual parallel treebank known as `ParTUT`, developed at the University of Turin. (3) A gold standard Universal Dependencies corpus for English, constructed using the source material of the English Web Treebank (`EWT`). (4) The English portion of the parallel Universal Dependencies (`PUD`) treebanks. (5) The English half of the `LinES` Parallel Treebank. (6) A dataset specifically created for pronoun identification (`Pronouns`).
We employ the `nltk` library (Loper and Bird, 2002; Bird et al., 2009) along with its `word_tokenizer` and `sent_tokenizer` for both sentence boundary detection (SBD) and tokenization tasks. We also utilize the `stanza` toolkit (Qi et al., 2020), which is a natural language processing toolkit based on Dozat's biaffine attention dependency parser (Dozat and Manning, 2017). This toolkit includes a standard tokenizer with built-in sentence boundary detection, enabling us to generate text and CoNLL-U format[3] outputs. We use these outputs to evaluate preprocessing results using our alignment-based evaluation algorithm. Furthermore, we juxtapose our evaluation findings with those of prior methods, such as the evaluation script utilized in the CoNLL 2018 Shared Task (Zeman et al., 2018), employing the CoNLL-U format outputs generated by `stanza`.
Both `nltk` and `stanza` handles tokenization and sentence boundary detection similarly in some aspects. For instance, they both split words like *gift's* into two tokens, *gift* and *'s*, and treat periods as separate tokens. However, they differ in their treatment of certain contentious areas, such as words containing dashes. For example, in the case of *search-engine* found in EWT, `nltk` considers it as one token, while `stanza` separates it into three tokens: *search␣-␣engine*. Another difference arises in how they identify sentence boundaries, especially in cases where the text lacks capitalized beginnings or period endings. This ambiguity is more pronounced in sentences without clear markers. Table 1 presents the results of case studies, including the numbers of true positives (TP), false positives (FP), and false negatives (FN) for both sentence boundaries and tokens. These results were obtained using the proposed alignment-based evaluation algorithm with both `nltk` and `stanza`. Additionally, we provide the results of the CoNLL evaluation script, which was applied to the SBD and tokenization results produced by `stanza` and formatted in CoNLL-U format.
The previous evaluation method also utilized precision and recall for F1 measures as evaluation metrics, relying on true positives (TP), false positives (FP), and false negatives (FN). In the CoNLL 2018 Shared Task evaluation script,[4] both tokens and sentences are treated as spans. In the case of a character-level mismatch in the positions of spans between the system output and the gold file, the script adjusts by skipping to the next token in the file with the smaller start value until

[3]https://universaldependencies.org/format.html

[4]https://universaldependencies.org/conll18/conll18_ud_eval.py

		GUM			ParTUT			EWT			PUD			LinES			Pronouns		
		TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
sbd	`nltk & jp`	845	137	251	149	2	4	1084	349	993	976	26	24	865	141	170	285	0	0
	`stanza & jp`	1038	38	58	144	6	9	1805	160	272	998	4	2	912	128	123	285	0	0
	`stanza`* & conllu	1038	38	58	144	6	9	1805	160	272	998	4	2	912	128	123	285	0	0
tk	`nltk & jp`	19443	351	462	3345	27	63	24176	1109	918	20733	251	443	17571	56	104	1673	16	32
	`stanza & jp`	19791	118	114	3381	20	27	24724	286	370	21162	18	14	17517	380	158	1649	28	56
	`stanza`* & conllu	19791	118	114	3381	20	27	24724	286	370	21162	18	14	17517	380	158	1649	28	56

Table 1: Numbers of TP (true positive), FP (false positive) and FN (false negative) using `nltk` and `stanza` for sentence boundaries and tokens. The `stanza*` line provides result numbers by the CoNLL-U evaluation script.

the positions align. This process is also applied to sentence boundaries. The start and end values of sentence spans are compared between the system and the gold file, with matching values incrementing the count of correctly matched sentences (true positive sentence boundaries). However, in our alignment-based evaluation method, this process is limited to the aligned sentence pair, ensuring accuracy in the counts. Any miscounted true positives can significantly impact the evaluation results negatively. Therefore, this paper addresses these issues in the existing evaluation scripts, highlighting them as sources of mismatches, and suggests adjusted alternatives to enhance the reliability of sentence preprocessing evaluation.

Due to the differing characters in these representations, our character-level evaluation of both `nltk` and `stanza` preprocessing results may not capture such cases accurately. The lack of consensus on tokenization across different corpora, including Universal Dependencies, contributes to the mismatch issue. Notably, EWT tokenizes *can't* as *ca* and *n't*, while ParTUT tokenizes it as *can* and *not*. We identified variations in the representation of contractions like *can't* and *ain't*. These contractions can be expressed in multiple ways, where EWT tokenizes *can't* as *ca* and *n't*, while ParTUT tokenizes it as *can* and *not*. The same issue can arise when converting "starting quotes" (``` `` ```) and "ending quotes" (`''`) in the corpus into straight quotes (`"`) in `nltk`, resulting in discrepancies. Since the number of contractions and symbols to convert, such as quotes, in a language is limited, we have created an exception list for our system to capture such cases in English. During the final stages of our implementation, we cross-check against the exception list to ensure that every case can be correctly handled by the proposed algorithm. As a result, the algorithm effectively addresses the preprocessing mismatches discussed in Section 2, which could otherwise disrupt our evaluation procedures.

4.2. Case study on Multilingual corpora

While we have addressed tokenization mismatches, such as the representation of contractions, other tokenization issues may arise from morphological segmentation or analysis, where additional morphemes can be introduced during the analysis rather than through mechanical token segmentation. Table 2 presents the results of case studies using seven GSD corpora (McDonald et al., 2013) in UD, which have been provided by Google. Our results closely align with those suggested by the CoNLL UD evaluation script, except for French. In French GSD, tokenization often combines several units into a single token. For example, the expression *de 1 000 mètres* ('of 1,000 meters') is treated as three tokens instead of four, as spaces are used to separate the words. Notably, this discrepancy is not a limitation of the proposed algorithm but rather stems from differences in tokenization conventions between plain text and the *rich* CoNLL-U format. Even when the input text is *de 1 000 mètres*, with *1* and *000* separated, it is tokenized as separated in plain text. Such examples occur across various treebanks. For example, in `UD_Kurmanji-MG`, phrases like *dagir kiriye* ('occupied') are tokenized as a single unit.

4.3. Discussion and limitation

The effectiveness of the proposed word alignment approach would remain unaffected even in the presence of significant morphological mismatches. For example, we trace back to the sentence in Hebrew described in Tsarfaty et al. (2012) as a word mismatch example caused by morphological analyses:

gold	^{0}B	$^{1.0}H \sim\sim\sim {}^{1.1}CL$	^{2}FL	^{3}HM	$^{4.0}H \sim\sim\sim {}^{4.1}NEIM$
	'in'	'the' 'shadow'	'of'	'them'	'the' 'pleasant'
sys	^{0}B	^{1}CL	^{2}FL	^{3}HM	$^{4}HNEIM$
	'in'	'shadow'	'of'	'them'	'made-pleasant'

Pairs of {$^{1.0}H$ $^{1.1}CL$, ^{1}CL} ('the shadow') and {$^{4.0}H$ $^{4.1}NEIM$, $^{4}HNEIM$} ('the pleasant') are word-aligned using the proposed algorithm, and we can obtain 4/5 and 4/7 for precision and recall using the proposed method. The CoNLL evaluation script is unable to assess such mismatches because *the concatenation of tokens in gold file and in system file differ.*[5] Unfortunately, we were unable to find real-world examples of morphological mismatches from GSD treebanks. Since `stanza` has been trained on UDs, it would produce UD-friendly

[5]This is an actual `UDError` message.

		DE			ES			FR			ID			JA			KO			PT		
		TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
sbd	`stanza & jp`	787	110	190	373	35	53	393	23	23	515	35	42	541	5	2	629	143	360	1161	71	39
	`stanza`* & conllu	787	110	190	373	35	53	393	23	23	515	35	42	541	5	2	629	143	360	1161	71	39
tk	`stanza & jp`	16172	61	52	11705	33	28	9714	17	22	11523	9	18	12750	361	284	11332	184	345	29311	52	50
	`stanza`* & conllu	16172	61	52	11705	33	28	9710	19	23	11523	9	18	12750	361	284	11332	184	345	29311	52	50

Table 2: Numbers of TP , FP and FN using `stanza` for sentence boundaries and tokens for multilingual case studies using `UD_*-GSD` where $*$ is German, Spanish, French, Indonesian, Japanese, Korean, and Portuguese.

results without discrepancies. Therefore, we view this area as a potential subject for future investigation.

Given the absence of consensus evaluation methods for tokenization and sentence boundary detection, as well as the lack of direct approaches to evaluate preprocessing outcomes, a comprehensive comparison with previous work is unfeasible. Instead of utilizing the plain text format, where conventional preprocessing tools are typically applied, we opt to perform comparisons using the CoNLL-U evaluation script in the CoNLL-U format, which offers representational advantages over plain text, such as representing tokenization results like *1 000* as a single token.

We have expanded our alignment-based evaluation approach to include other tasks, such as evaluating constituency parsing results. The widely used `evalb` script has traditionally been employed for evaluating the accuracy of constituency parsing results, albeit with the requirement for consistent tokenization and sentence boundaries. We align sentences and words when discrepancies arise to to overcome several known issues associated with `evalb` by utilizing the 'jointly preprocessed alignment-based method (Jo et al., 2024). The proposed approach will also be applicable to various sentence-based evaluation metrics, including POS tagging, machine translation, and grammatical error correction.

5. Conclusion

While most existing tokenization and sentence boundary detection (SBD) implementations are generally considered suitable for direct re-implementation, it is important to note that when they are applied to new use cases, many miscounted true positives are likely to be overlooked. As a result, these inaccuracies remain hidden and not immediately apparent in the intermediate preprocessing results. However, these text segmentation tasks play a fundamental role in sentence processing. Any unnoticed inaccuracies at these early stages can potentially be magnified in downstream NLP tasks, significantly affecting the entire NLP pipeline. This issue of miscounted true positives is a largely unacknowledged aspect of sentence preprocessing. By introducing sentence and word alignments into the proposed pipeline, we can better identify and reassess such hidden but prevalent inaccuracies in the foundational preprocessing steps. Through the jp-algorithm, we can focus on addressing mismatches that occur during crucial preprocessing procedures and accurately count the true positives during evaluation. All codes and results from the case studies can be accessed at `https://github.com/jungyeul/jp-preprocessing/`.

6. Ethics Statement

We would like to affirm that our research or project has been conducted in accordance with ethical guidelines and standards. We have thoroughly reviewed and addressed all relevant ethical considerations, and we can confidently state that we have no ethical concerns associated with our work.

7. Acknowledgements

This research is based upon work partially supported by *Students as Partners* for Eunkyul Leah Jo and Grace Zhang.

8. Bibliographical References

Tanvirul Alam, Akib Khan, and Firoj Alam. 2020. Punctuation Restoration using Transformer Models for High-and Low-Resource Languages. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 132–142, Online. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Newton, Massachusetts, United States.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

John DeNero and Dan Klein. 2007. Tailoring Word Alignments to Syntactic Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages

17–24, Prague, Czech Republic. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, Toulon, France. The International Conference on Learning Representations (ICLR).

Rebecca Dridan and Stephan Oepen. 2012. Tokenization: Returning to a Long Solved Problem — A Survey, Contrastive Experiment, Recommendations, and Toolkit —. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 378–382, Jeju Island, Korea. Association for Computational Linguistics.

Kilian Evang, Valerio Basile, Grzegorz Chrupała, and Johan Bos. 2013. Elephant: Sequence Labeling for Word and Sentence Segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1426, Seattle, Washington, USA. Association for Computational Linguistics.

Xue-Yong Fu, Cheng Chen, Md Tahmid Rahman Laskar, Shashi Bhushan, and Simon Corston-Oliver. 2021. Improving Punctuation Restoration for Speech Transcripts via External Data. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 168–174, Online. Association for Computational Linguistics.

William A. Gale and Kenneth W. Church. 1993. A Program for Aligning Sentences in Bilingual Corpora. *Computational Linguistics*, 19(1):75–102.

Dan Gillick. 2009. Sentence Boundary Detection and the Problem with the U.S. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 241–244, Boulder, Colorado. Association for Computational Linguistics.

Carlos-Emiliano González-Gallardo and Juan-Manuel Torres-Moreno. 2018. WiSeBE: Window-based Sentence Boundary Evaluation. In *Advances in Computational Intelligence: Proceedings of the17th Mexican International Conference on Artificial Intelligence (Part II), MICAI 2018*, pages 119–131, Guadalajara, Mexico. Springer International Publishing.

Eunkyul Leah Jo, Angela Yoonseo Park, and Jungyeul Park. 2024. A Novel Alignment-based Approach for PARSEVAL Measures. *Computational Linguistics*, pages 1–10.

Tibor Kiss and Jan Strunk. 2006. Unsupervised Multilingual Sentence Boundary Detection. *Computational Linguistics*, 32(4):485–525.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of The Tenth Machine Translation Summit X*, pages 79–86, Phuket, Thailand. International Association for Machine Translation.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement. In *Proceedings of the Human Language Technology Conference of the {NAACL}, Main Conference*, pages 104–111, New York City, USA. Association for Computational Linguistics.

Kyungtae Lim and Jungyeul Park. 2024. Real-world Sentence Boundary Detection using Multi-Task Learning: A Case Study on French. *Natural Language Engineering*, 30(1):150–170.

Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Wei Lu and Hwee Tou Ng. 2010. Better Punctuation Prediction with Dynamic Conditional Random Fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 177–186, Cambridge, MA. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal Dependency Annotation for Multilingual Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, page 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hong Kong. Association for Computational Linguistics.

David D. Palmer and Marti A. Hearst. 1997. Adaptive Multilingual Sentence Boundary Disambiguation. *Computational Linguistics*, 23(2):241–267.

Martin Poláček, Petr Červa, Jindřich Žďánský, and Lenka Weingartová. 2023. Online Punctuation Restoration using ELECTRA Model for streaming ASR Systems. In *Proceedings of INTERSPEECH 2023*, pages 446–450, Dublin, Ireland. International Speech Communication Association.

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.

Jonathon Read, Rebecca Dridan, Stephan Oepen, and Lars Jørgen Solberg. 2012. Sentence Boundary Detection: A Long Solved Problem? In *Proceedings of COLING 2012: Posters*, pages 985–994, Mumbai, India. The COLING 2012 Organizing Committee.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, Washington, DC, USA. Association for Computational Linguistics.

Rico Sennrich and Martin Volk. 2011. Iterative, MT-based Sentence Alignment of Parallel Texts. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA 2011)*, pages 175–182, Riga, Latvia. Northern European Association for Language Technology (NEALT).

Ann Taylor, Mitchell Marcus, and Beatrice Santorini. 2003. The Penn Treebank: An Overview. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 5–22. Springer Netherlands, Dordrecht.

Brian Thompson and Philipp Koehn. 2019. Vecalign: Improved Sentence Alignment in Linear Time and Space. 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 1342–1348, Hong Kong, China. Association for Computational Linguistics.

Marcos Treviso, Christopher Shulby, and Sandra Aluísio. 2017. Evaluating Word Embeddings for Sentence Boundary Detection in Speech Transcripts. In *Proceedings of the 11th Brazilian Symposium in Information and Human Language Technology*, pages 151–160, Uberlândia, Brazil. Sociedade Brasileira de Computação.

Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2012. Joint Evaluation of Morphological Segmentation and Syntactic Parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 6–10, Jeju Island, Korea. Association for Computational Linguistics.

Dániel Varga, Lázló Németh, Péter Halácsy, András Kornai, Viktor Trón, and Viktor Nagy. 2005. Parallel corpora for medium density languages. In *Proceedings of the RANLP (Recent Advances in Natural Language Processing)*, pages 590–596, Borovets, Bulgaria.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.