CoNLL-Merge: Efficient Harmonization of Concurrent Tokenization and Textual Variation

Christian Chiarcos
Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany
http://www.acoli.informatik.uni-frankfurt.de/
chiarcos@informatik.uni-frankfurt.de

Niko Schenk
Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany
schenk@informatik.uni-frankfurt.de

Abstract

The proper detection of tokens in of running text represents the initial processing step in modular NLP pipelines. But strategies for defining these minimal units can differ, and conflicting analyses of the *same* text seriously limit the integration of subsequent linguistic annotations into a shared representation. As a solution, we introduce *CoNLL Merge*, a practical tool for harmonizing TSV-related data models, as they occur, e.g., in multi-layer corpora with non-sequential, concurrent tokenizations, but also in ensemble combinations in Natural Language Processing. *CoNLL Merge* works unsupervised, requires no manual intervention or external data sources, and comes with a flexible API for fully automated merging routines, validity and sanity checks. Users can chose from several merging strategies, and either preserve a reference tokenization (with possible losses of annotation granularity), create a common tokenization layer consisting of minimal shared subtokens (loss-less in terms of annotation granularity, destructive against a reference tokenization), or present tokenization clashes (loss-less and non-destructive, but introducing empty tokens as place-holders for unaligned elements). We demonstrate the applicability of the tool on two use cases from natural language processing and computational philology.

2012 ACM Subject Classification Applied computing → Format and notation; Applied computing → Document management and text processing; Applied computing → Annotation; Software and its engineering → Interoperability

Keywords and phrases data heterogeneity, tokenization, tab-separated values (TSV) format, linguistic annotation, merging

Digital Object Identifier 10.4230/OASIcs.LDK.2019.7

Supplement Material https://github.com/acoli-repo/conll

Funding The research described in this paper was supported by the project *Linked Open Dictionaries* (LiODi, 2015-2020), funded by the German Ministry for Education and Research (BMBF) and the project *Machine Translation and Automated Analysis of Cuneiform Languages* which is generously funded by the German Research Foundation (DFG), the Canadian Social Sciences and Humanities Research Council, and the National Endowment for the Humanities through the T-AP Digging into Data Challenge.

1 Motivation

Linguistic annotations of running text exhibit a great diversity and comprise, among others, part-of-speech tags, phrasal chunks, syntactic parses, semantic roles, or discourse relations. *Tokenization* as the initial pre-processing step is concerned with the proper detection and segmentation of application-specific, minimal units, i.e. *tokens*, and represents the basis for subsequent annotations. Tokens can be typified by words (lexemes or morphemes) or other methodologically-informed symbols (numbers, alpha-numerics and punctuation), and

have various properties: In many applications, they constitute the basis for "word" distance measurements, e.g., Normalized Levenshtein [29] and related similarity tasks involving word embeddings [17]. Beyond that, in many annotation tools and their corresponding formats, the order of tokens provides a timeline for the sequential order of structural elements [18].

Similarly, multi-layer formats refer to tokens in order to define the absolute position of annotation elements, and only by reference to a single privileged token layer (or an alternative base segmentation), annotations from different layers can be put into relation with each other [3]. On the single privileged token layer, tokens are organized in a total order, they cover the full annotated portion of the primary data, and represent the smallest unit of annotation. This aspect is especially important for the study of richly annotated syntactic and semantic resources, as an integration and serialization of linguistic annotations produced by *different* tools is usually established by reference to the token layer. Unfortunately, different annotation routines on the *same* texts oftentimes rely on *concurrent* tokenization schemes, which crucially requires efforts for harmonization. This is of particular relevance for NLP tools which need to draw on multiple linguistic annotations but for which concurrent information (potentially stored in alignment-incompatible, distinct data formats) heavily complicate their development process.

Our Contribution. Based on these observations, we argue that with the availability of robust conversion and flexible merging routines for standardized CoNLL and TSV-related data models, complex NLP tools that rely on a multitude of linguistic annotations can be realized in a more straightforward way. To this end, we introduce *CoNLL Merge*, a fully automated, application-independent merging routine for linguistic annotations based on different underlying tokenizations of the same text. The theoretical basis for our approach is described in [22], however, while [22] build their implementation of a highly complex standoff XML formalism with limited use in natural language processing and the language sciences, our implementation focuses on one-word-per-line (OWPL) tab separated value (TSV) formats, a simple formalism with wide application in corpus linguistics, lexicography and natural language processing, most famously associated with the long series of CoNLL shared tasks.

We are aware that project- or application-specific solutions for automated tokenization harmonization do exist (cf. Sect. 5). In opposition to these, *CoNLL Merge* provides a generic solution which does not only allows to merge files in *any* OWPL TSV format without manual interference, but which also allows to define the merging strategy – depending on whether the user prefers reversibility or keeping/enforcing a particular tokenization. We illustrate the practicability of our approach on a collection of annotated texts from the Wall Street Journal-based corpora that are in the intersection of several corpora with independent manual annotations, frequently with concurrent tokenizations. A second experiment on historical texts demonstrates the robustness of CoNLL Merge against textual variation *beyond* tokenization mismatches.

The paper is structured as follows: Section 2 describes alignment strategies for plain (tokenized) text, Section 3 describes the merging of the associated annotations, and Section 4 provides an evaluation.

2 Aligning Tokenizations

Among both efforts to manually create annotations for linguistics and philology and NLP tools to automatize such annotations, we find a remarkable band-width and variation even within a single language. If multiple annotators (manual or automated) are applied the

same piece of text, they choose (or require) a specific tokenization strategy – and this may deviate greatly from the tokenization adopted by another. Tokenization strategies differ with respect to the research question or application of interest (e.g., tagging, parsing, information extraction), and can be divided into morphosyntactic, full syntactic, and morphology-based analyses. For instance, tokenizations can drastically disagree as in the examples to the right for the text *the attorney general's office* [6]:

1. [attorney] [general's] British National Corpus [2]
2. [attorney] [general][']s] Tnt Tagger [1]
3. [attorney] [general]['s] Penn TreeBank [14]
4. [attorney general]['s] Protein Name Tagger [28]

Crucially, when dealing with multiple linguistic annotations on top of concurrent tokenizations of the *same* text, efforts for harmonization are required. Here, we focus on strategies for their automated alignment. The handling of associated annotations is subject to Sect. 3.

2.1 Identity-Based Alignment

The primary strategy for aligning concurrent tokenizations is based on string identity between different variants of the same text: Even if token boundaries have been inserted and whitespaces have been normalized, we can normally assume that textual content remains untouched.[1]

For string alignment, we build on existing diff implementations, most notably Myer's Diff [19, 15].[2] The scope of our implementation differs from standard Un*x functionalities in that the basis of comparison is the token rather than the line. In default alignment, tokenization mismatches are described by insertions and deletions of tokens. Thus, concerning the alignment between two files, `FILE1` and `FILE2`, three cases have to be distinguished, which we handle as follows:

1. 1:1 alignment
2. 1:0 alignment: For $n:0$ alignment, spell out n lines (tokens) with 1:0 alignments.
3. 0:1 alignment: For $0:m$ alignment, spell out m lines (tokens) with 0:1 alignments.

An $n:m$ alignment will thus be represented by a sequence of $n:0$ (1:0) and $0:m$ (0:1) alignments. In addition to this default merging, we support two merging strategies based on string identity:

forced. Enforce a 1:1 (or 1:0) alignment by concatenating the last 1:1-aligned token from `FILE2` (and its annotations) with those of the following 0:m alignments.

split. Enforce a dense alignment based on maximal common substrings: After default alignment, define an alignment window as a sequence of tokens that start with 1:1 alignment, followed by a sequence of 0:1 and 1:0 alignments. Within that window, perform a character-level (rather than a token-level) diff and aggregate consecutive sequences of 1:1 character alignments into maximal common substrings.

[1] In fact, this is often not true, as annotation tools may replace reserved characters with special symbols or escape sequences, enforce different character encodings or drop, for example, diacritics. However, our implementation has been proven robust against such changes.

[2] At the moment, we employ the implementation from Java *diff utils* by Dmitry Naumenko, `https://github.com/dnaumenko/java-diff-utils`.

It should be noted that the split strategy does not guarantee to arrive at 1:1 alignments in case of character insertions or deletions.[3] In those cases, another force alignment can be applied to eliminate $0:1$ alignments. Likewise, split alignment can be applied after force alignment to reduce the number of $1:0$ alignments. In either case, another challenge is the treatment of the associated annotations.

2.2 Similarity-Based Alignment

CoNLL-Merge was originally intended to cope with conflicting annotations of the same text. However, initial experiments showed that it is also directly applicable as a collator, i.e., a tool that identifies corresponding and deviating text passages and merges them into a common representation. Collators are frequently used in various branches of computational philology, e.g., to identify patterns of re-use and adaptation among different textual fragments (intertextual relations) or manuscript and edition genealogy (stemmatology). Taking CollateX (see Sect. 4.2 below) as an example, it resembles CoNLL Merge in building on existing diff implementations, it exceeds plain diff functionalities in providing convenient user interfaces and visualizations. Unlike CoNLL Merge, CollateX is restricted to plain text and does not provide a way for harmonizing annotations.

Initial experiments for applying CoNLL Merge to philological data have been performed against a small collection of medieval manuscripts written in different orthographies in Middle Low German (cf. Sect. 4.2 below). CoNLL Merge successfully achieved an alignment despite the fact that these texts deviated in their choice of words and in editorial changes such as insertions and deletions of large portions of texts. However, the method was obviously not sufficiently robust against deviating orthographies (a common problem in medieval texts). In order to improve its usability in Digital Humanities contexts, we provide an additional merging strategy based on string similarity rather than identity, based on minimal edit distance, resp., Levenshtein distance [29]. Like force and split, Levenshtein alignment is applied after default alignment was applied to determine alignment windows (non-aligned tokens preceded and/or followed by identity-aligned tokens).

levenshtein. Within the alignment window, determine the source and target token pair with minimum Levenshtein distance. Accept this as alignment and create novel alignment windows before and after the aligned words. Iterate until no more $n:m$ alignments (i.e., sequences of $n:0$ and $0:m$ alignments) remain.

As our application of Levenshtein alignment does not support crossing edges, it does often not produce a 1:1 alignment.

3 Merging Annotations

For merging annotations in one-word-per-line formats, we focus on tabular formats using tab-separated values (TSV), as widely used in corpus linguistics and lexicography, but also in natural language processing (most notably in the context of the long series of CoNLL Shared Tasks).

[3] For identical text, apparent insertions or deletions can arise from different escaping strategies, e.g., the replacement of double quotes with two single quotes, or encoding differences, e.g., the direct encoding of UTF-8 characters or their representation as XML entities.

3.1 The CoNLL Format Family

Since 1999, the Conference on Natural Language Learning[4] (CoNLL) has established a tradition of annual shared tasks in which training and test data is provided by the organizers, thereby facilitating the systematic evaluation of participating tools.[5] With their continuous progression in terms of linguistic complexity, the shared tasks reflect the maturation of statistical NLP, the dominating paradigm of computational linguistics in the 2000s. In many cases, successful participants established reference tools, and – as it allowed for comparative evaluation – the underlying, standardized formats continued to be supported by succeeding NLP tools, which in fact has reinforced the global importance of the CoNLL format family within the language processing community and which thus represents the core basis for the merging routine described in this paper.

3.2 CoNLL Merge

We offer a lightweight Java package for sanity checks, format testing, producing, manipulating and – most notably – merging of TSV files. On the one hand, `CoNLLChecks` can be applied for selected validity checks on a set of CoNLL files.[6] This is particularly useful as a preprocessing step before the actual merging routine. On the other hand, for merging on token and subtoken level itself, most importantly, `CoNLLAlign` establishes the core interface to the implementation. It takes two files to be merged as input (`FILE1 FILE2`) allowing for the following options:

- default
  1:1 alignment: write the `FILE1` line, write a tabulartor, write the `FILE2` line.
  1:0 alignment: write the `FILE1` line, fill up missing `FILE2` columns with the placeholder (`?`).
  0:1 alignment: create an "empty" token (∗`RETOK`∗-<`FORM`>, where `FORM` is replaced by the token string of the `FILE2` line), append placeholder characters (`?`) for the annotations expected from `FILE1`, append the corresponding `FILE2` lines.
- `-f` forced merge: mismatching `FILE2` tokens are merged with last `FILE1` token. This flag suppresses ∗`RETOK`∗ nodes, thus keeping the original token sequence intact. Annotations of merged lines are concatenated, using + as a separator.
- `-split` (boolean): `false` merges two CoNLL files and adopts the tokenization of the first. Tokenization mismatches from the second are represented by empty artificial tokens, i.e. words prefixed with ∗`RETOK`∗-... – `true` splits tokens from both files into maximal common substrings. From a split line, all annotations are copied to the lines of the subtokens. In order to mark the scope of a particular annotation, we use the I(O)BE(S) schema: Split annotations at the line of the first subtoken are prefixed by `B-` (begin), split annotations at the line of the last subtoken are prefixed by `E-` (end), and intermediate annotations are prefixed by `I-`. The flag `-split` is a shorthand for `-split=true`.

[4] `http://www.conll.org/`
[5] In the context of CoNLL shared tasks, one-word-per-line TSV formats have been applied to the following phenomena: shallow parsing (1999-2001), lexical semantics (2002-2003), dependency syntax (2006-2009, 2017-2018), semantic role labeling (2004-2005, 2008-2009), coreference (2011-2012), discourse parsing (2015-2016), inflectional morphology (2017-2018), applied NLP (2010, 2013-2014). Some recent shared tasks on highly abstract levels of linguistic analysis involved JSON formats along with the classical TSV format (discourse parsing, 2015-2016), or in their place (semantic parsing, 2019).
[6] In particular, on the number of columns, mismatches between parentheses, IOBES statements, cells without content and checks on comments.

- `-lev` perform Levenshtein alignment
- `-drop none` keep both versions of the merged column (by default, the `FILE2` column is dropped).

Additionally, we provide merge scripts for multiple (iterated) pair-wise alignments and final merging of multiple documents, as well as test cases for validity checks on the resulting CoNLL output.

4 Evaluation

We evaluate CoNLL Merge against two use cases: Alignment and annotation merging for multi-layer corpora, and alignment between deviating textual variants.

4.1 Same Text – Different Annotations

In order to evaluate the usefulness and practicability of *CoNLL Merge*, we describe a workflow of semantic annotation integration of a diverse collection of well-established, standard data sets which are all grounded on the *same* base texts taken from the Wall Street Journal (WSJ) data of the Penn Treebank [13, PTB]. We focus on the latest Penn Treebank version with syntactic phrase structure annotations [14, PTB3], PropBank [20, verbal predicate argument structure], NomBank [16, nominal semantic roles], OntoNotes [11, coreference], the Penn Discourse Treebank [21, PDTB2/shallow discourse structure], the RST Discourse Treebank [4, RSTDTB/hierarchical discourse structure], and the Discourse Graphbank [27, PDGB]. Crucially, not every resource provides annotations for every document in the PTB.

Figure 1 shows the (relative) number of corpora that a PTB file occurs in, averaged over WSJ sections. This information can be easily acquired by running *CoNLL Merge* on the distinct data sets as a preprocessing step. Ideally, a 100% bar in the chart would signify that each document of the respective section is annotated by all resources. Top sections range between 45–50% which indicates that an average document is found in half of the corpora.

It is important to note that all aforementioned resources come with partly conflicting, i.e. *varying* underlying tokenizations. The urgent need to make use of such multiple, distinct linguistic annotations in a *joint* learning framework (e.g., for discourse parsing based on syntactic dependencies, or semantic roles that are part of a coreference chain) has been the focus of a number of recent successful computational approaches [25, 26, 24, 23]. Figure 2 (left) illustrates our approach to harmonize concurrent tokenizations and to merge their annotation as part of a semantic annotation workflow that merges *all* levels of annotations provided for WSJ data.

In the first step, corpora with their original idiosyncratic tokenizations and linguistic annotations are converted to a CoNLL or TSV format. Some data sets provide CoNLL formats by default, for most others, converters are available. For the more exotic formats (PDGB, PDTB, RST-DTB), we provide CoNLL converters as part of the CoNLL Merge release. Sanity checks are performed by `CoNLLChecks`. Then, *pairwise* merges between two CoNLL files are produced (`CoNLLAlign`). Finally, *full* merges are generated resulting in the global data structure that shares the content of all base resources. In total, our method encounters 5,542 tokenization mitmatches out of which on average 98.7% are resolved and successfully merged with the different flag options.[7]

[7] Rare issues are encountered for cases in which subsequent tokenization conflicts appear immediately adjacent. Moreover, since merging can be easily parallelized, our routine runs reasonably fast (< 3 mins

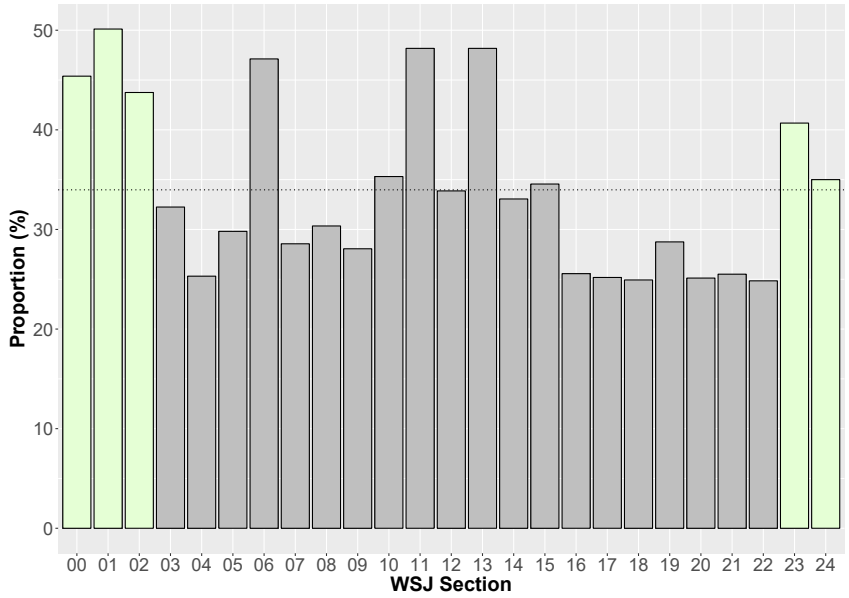

Figure 1 Relative number of corpora that contain a particular PTB document, averaged over WSJ sections.

Table 1 Alignment of OntoNotes (ON) parse files with other annotations, file wsj_0655, 992 (ON) tokens.

		PTB	RST	PDGB	PDTB*
default	1:1 ON alignment	959 (97%)	811 (82%)	834 (84%)	609 (61%)
alignment	1:0	33 (3%)	181 (18%)	158 (16%)	383 (39%)
	0:1	11 (1%)	113 (11%)	141 (14%)	51 (5%)
force	1:1 (no merge)	968 (98%)	839 (85%)	852 (86%)	643 (65%)
alignment	1:0	23 (2%)	114 (11%)	91 (9%)	340 (34%)
(-f)	0:1	0 (0%)	0 (0%)	0 (0%)	2 (0%)
	merged annotations	1 (0%)	39 (4%)	49 (5%)	9 (1%)
split	1:1 (no split)	959 (97%)	811 (82%)	834 (84%)	609 (61%)
alignment	1:0	0 (0%)	92 (9%)	91 (12%)	311 (31%)
(-split)	0:1	0 (0%)	32 (3%)	25 (9%)	12 (1%)
	split annotations	33 (3%)	50 (5%)	118 (12%)	98 (11%)
-split -f	1:1 (no split/merge)	959 (97%)	811 (82%)	834 (84%)	609 (61%)
	1:0	0 (0%)	72 (7%)	74 (7%)	270 (27%)
	0:1	0 (0%)	0 (0%)	0 (0%)	0 (0%)
	split/merged annotations	33 (3%)	62 (6%)	118 (12%)	98 (10%)

* contains text fragments only

Table 1 illustrates the extent of tokenization differences and the effect of the merging strategies for file `wsj_0655`, one of only seven files contained in the intersection of OntoNotes, Penn Treebank, RST Discourse Treebank, Penn Discourse Graphbank and Penn Discourse Treebank.[8] For illustration, we use the tokenization of OntoNotes parses as primary tokenization and match all other annotations against it. The PTB provides a slightly older version of the *same* annotations and tokenization, nevertheless deviating in 3% of the tokens. One source of deviation is in the treatment of hyphenized words and multi-word expressions. Note that these annotations do not just tokenize the original text. In addition to text tokens, empty tokens are inserted to represent syntactic movement operations. The RST edition provides untokenized text plus markup for paragraph boundaries. Similarly, the PDGB edition uses untokenized text. In the RST, PDGB and PDTB converters we provide, a TSV representation is created by treating every white-space separated string as a token. The PDTB edition is very different in that it is a standoff format which does not provide the full text, but only the text of the annotated spans, and their character offsets in the original text file. The standoff mechanism is defined with reference to a PTB version (similar but not identical to the OntoNotes version used here) or against the original plain text (which none of these corpora provide). Our converter does *not* attempt to resolve standoff references. Instead, we use the character offsets and the text provided in the span to *reconstruct* the plain text. As the spans contain only about 60% of the original text, this reconstruction is incomplete, its TSV representation can nevertheless be successfully aligned against OntoNotes (or any other full-text corpus).

As the table shows, the merging strategies have the following characteristics:

default is fully reversible, in that original token boundaries and the original annotations are preserved. The empty elements introduced for $0:m$ alignments, however, do interrupt the original sequence of tokens from `FILE1`.

-f enforces the tokenization of `FILE1` onto `FILE2`. `FILE2` annotations and tokenizations can be altered in an irreversible fashion, in that annotations are concatenated without the possibility to align them with their original token boundaries.

-split is fully reversible, in that the original token boundaries and the extent of the original can be recovered. Interruptions by $0:m$ alignments are minimized, but the original `FILE1` tokenization is altered. A tool expecting `FILE1` tokenization should not be applied to the output of this merging operation.

-split -f also enforces `FILE1` tokenization, but distributes `FILE2` annotations over multiple `FILE1` tokens (where applicable).

Three main objectives can be pursued: annotation reversibility (default), the establishment of a tokenization based on maximal shared substrings (-split, reduces both $n:0$ and $0:m$ alignments) or adoption of a privileged tokenization (-force, i.e., strictly enforce 1:1 or 1:0 alignments).

As an example, consider the following phrase from `wsj_0655`: *Lawmakers haven't publicly raised the possibility of renewing military aid[...]* Figure 2 (right) shows the result of three pairwise merges between the Penn Discourse Treebank as primary source and the annotations in PropBank. For illustration purposes, we first highlight the different tokenization outputs.

---

for the complete PTB with standard CPU).

[8] For demonstration purposes, the different versions of this file are included in the associated software distribution. However, for reasons of copyright, the archive is encrypted archive and the password must be requested from the first author. Alternatively, access to the different versions of the file can be requested from LDC, `https://catalog.ldc.upenn.edu`.

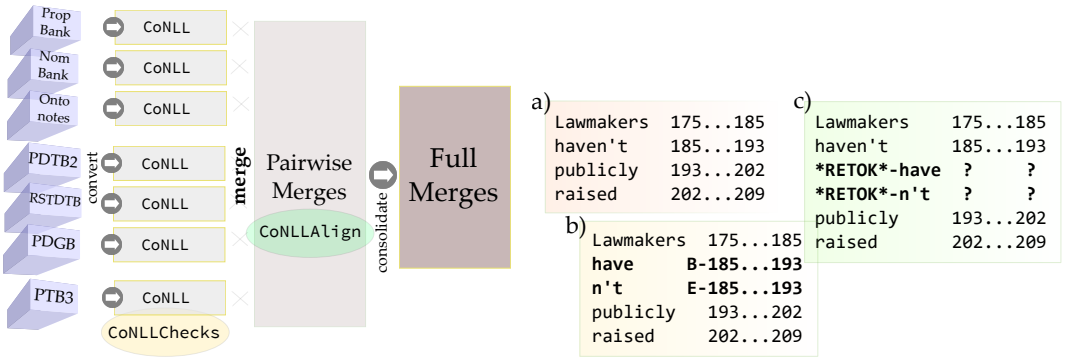

Figure 2

Left: Harmonizing PTB corpora (conversion & merging of ling. annotations) into one CoNLL output.

Right: Merged tokenizations between PDTB2 and PropBank: `-f` (a), `-split=true` (b), default (c).

Merging was performed with a combination of flag options (forced and default merging). The latter adopts the tokenization of the primary source, and inserts ∗RETOK∗ tokens, whenever alternatives are encountered within PropBank. With the `-split` option set to `true`, spans are equipped with underspecified beginning and end indices. The reason for having varying tokenizations across the two resources is due to the requirements of their idiosyncratic linguistic annotations. In PropBank, for instance, it is necessary to assign an individual modifier role to the negation (`ARGM-NEG`), therefore requiring a distinct token (*n't*) to be split from the orthographically combined auxiliary verb. In contrast, in the discourse setting of the PDTB2, only larger (shallow) spans are considered which dispense with the need for such a fine-grained segmentation. However, *CoNLL Merge* allows for a fruitful combination of both types of complementing linguistic annotations into one shared layer: Fig. 4 in Appendix A shows the combined information including discourse aspects as well as predicative argument structure (semantic role annotations) into one harmonized CoNLL token layer.

4.2 Same Source – Different Text

Beyond comparing and merging annotations of the same texts, CoNLL Merge can also be used as a collator for the alignment of different versions of the same text, and thus, for projecting annotations from one text to another. In philology, collation is the process of determining the differences between two or more variants of a text (e.g., different editions of a book, or different manuscripts of a particular text).

Designated tools for the purpose exist, e.g., CollateX [8], but they do currently not support the alignment of annotated text nor the projection of annotations from one textual variant to another. Instead, they focus on applications in stemmatology and the study of intertextual relations and provide graphical interfaces for the purpose. CollateX reads multiple plain text versions of a text, performs tokenization on each version, performs an alignment similar to that of CoNLL Merge and returns alignment results for further processing, for instance for producing a critical apparatus.

CoNLL Merge does provide a similar functionality, albeit with a focus on annotation rather than stemmatology and intertextuality:[9] We experimented with several merging routines on different Middle Low German editions of the *Interrogatio Sancti Anselmi de Passione Domini*. For reasons of copyright, we cannot ship the sample data, so we provide a script

[9] Scripts and data set available under `https://github.com/acoli-repo/conll/tree/master/data_phil`.

Table 2 Alignment of Anselmus ms. D with 6 other mss., 7263 tokens (D).

		D2	HA1521	Kh	O	SP	StA1495
default	1:1	6085 (84%)	4231 (58%)	4581 (63%)	4505 (62%)	4412 (61%)	3801 (52%)
alignment	1:0	993 (14%)	2856 (39%)	2506 (35%)	2575 (35%)	2675 (37%)	3286 (45%)
	0:1	1079 (15%)	2336 (36%)	2422 (35%)	2646 (37%)	2556 (37%)	2540 (40%)
Levenshtein	1:1	6817 (94%)	5453 (75%)	5831 (80%)	6070 (84%)	5900 (81%)	5186 (71%)
alignment	1:0	261 (4%)	1634 (22%)	1256 (17%)	1010 (14%)	1187 (16%)	1901 (26%)
(-lev)	0:1	347 (5%)	1114 (17%)	1172 (17%)	1081 (15%)	1068 (15%)	1155 (18%)

for downloading the original PDFs, for text extraction and for creating a CoNLL-compliant TSV file with three columns:

- We perform whitespace tokenization. Punctuation signs are not separated from grammatical words.
- The first columns contains the original string value, including punctuation signs.
- The second columns contains a normalized representation of the string value, with a number of orthographical conventions being harmonized (punctuation removed, lowercase, removal of diacritics). This normalization is not language-specific, but presupposes a Latin-based orthography.
- The third column contains a language-specific normalization of the normalized string. For instance, many medieval orthographies use *w*, *u* and *f* interchangeably with *v* (for different phonemes), so that *w,u,f* are all normalized to *v*.

For seven Middle Low German Anselmus manuscripts, we performed alignment (collation) over the third column. As gold data for the alignment of these texts is currently not available, we only report the number of successfully aligned tokens. Note that mis-alignment results in a low likelihood that subsequent tokens will be aligned, so that 1:1 alignments for more than 50% in default mode generally indicate alignment success. For random samples, this has been manually confirmed by the authors. However, the manuscripts differ considerable, both in their orthography and formulations, but also in additions and omissions. For instance, manuscripts D and D2 share a prolog of 235 tokens which is absent from the other manuscripts. Table 2 shows the alignment results for Anselmus ms. D. with six other manuscripts.

Figure 3 illustrates collation/alignment results for the second to fourth shared sentence of Anselmus mss. D, D2 and HA1521. This example illustrates typical alignment errors: Default alignment frequently fails to identify orthographic variants of the same word. These errors are inherited by force alignment, but not by Levenshtein alignment.

5 Summary

In this paper, we have described *CoNLL Merge*, a fully automated merging routine for harmonizing linguistic annotations in multi-layer corpora which are based on *concurrent tokenizations* of the same text.

To our best knowledge, CoNLL Mergeis the first system to perform this task in a generic fashion for one of the most popular corpus formalisms: one-word-per-line tab-separated values, as used in the CoNLL shared tasks, in popular corpus information systems [10] or for digital lexicography [12]. We are aware of existing implementations for handling conflicting tokenizations: Solutions based on hand-crafted rules [9] suffer from a lack of genericity. A number of libraries for merging CoNLL files do already exist, however, these are restricted to

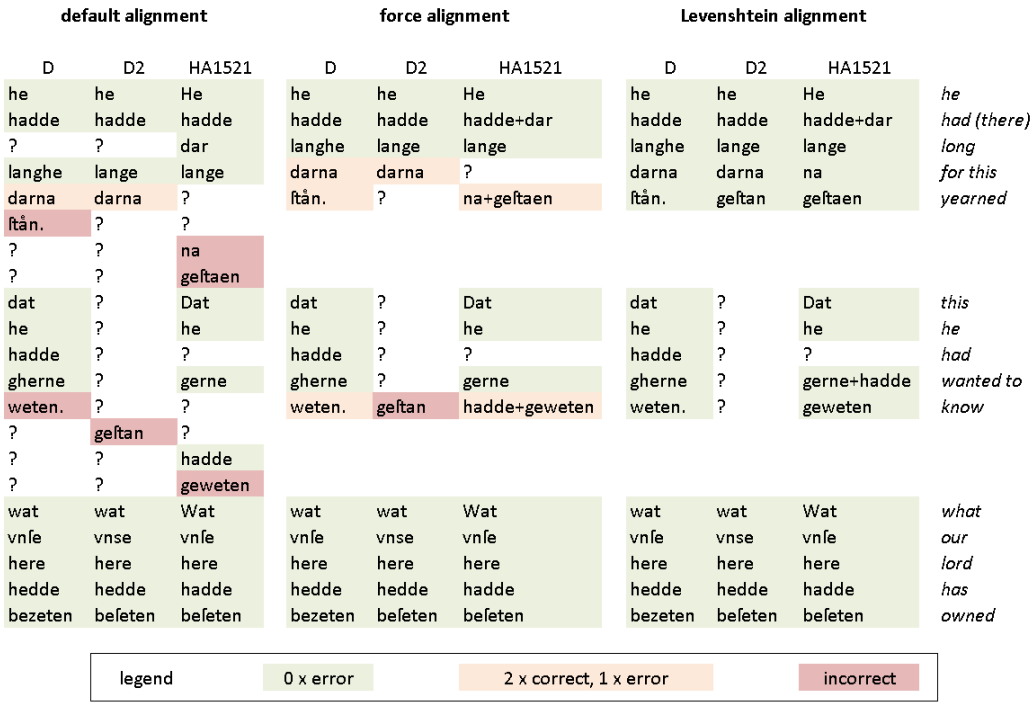

Figure 3 Collation results for Anselmus ms. D, D2 and HA1521, second to fourth shared sentence: *He yearned for this for a long time. He wanted to know: What did our lord own?* Question marks indicate the absence of an alignable token.

individual CoNLL dialects such as CoNLL-U[10] or CoNLL-X,[11] whereas our implementation is fully generic in that it allows the user to configure what column(s) to take as the basis for comparison. Our own earlier work on the automated resolution of tokenization mismatches [5] basically implemented the same functionality as CoNLL Merge, but this was closely tied to a highly complex standoff annotation formalism, not directly applicable to common corpus formats – and, effectively, forgotten. Finally, designated modules included in a number of NLP pipeline systems provide heuristic components for the resolution of tokenization mismatches, e.g., the Illinois NLP Curator [7] implements a maximum common substring strategy. These suffer from a similar limitation, i.e., these components are tightly integrated with a particular implementation, and not applicable to annotations in general. Moreover, we are not aware of any such module which allows the user to select among possible merging strategies.

Beyond this, we have shown that our implementation is applicable to texts with variation far beyond tokenization differences. In fact, CoNLL Merge can be used as a collator. Unlike other state-of-the-art collators, however, CoNLL Merge allows to perform collation with annotated texts and supports the projection of annotations from one text variant to another. For applications in NLP, this demonstrates that CoNLL Merge can also be applied for alignment of and annotation projection between paraphrases.

CoNLL Mergeis an efficient implementation of string-based comparisons, it works unsupervised, and does not require manual interference or any external resources. We demonstrated its applicability to successfully combine distinct linguistic annotations by connecting inform-

[10] `https://www.npmjs.com/package/conllu`
[11] `https://github.com/danieldk/conllx-utils`

ation from language resources which by default come with incompatible token layers. The augmented data obtained in this way enable improved insight into the interplay of annotations provided by distinct linguistic frameworks, allow for advanced NLP tool development, and due to its generic functionality could easily be extended to merging of morphologically more complex languages.

References

1 Thorsten Brants. TnT: A Statistical Part-of-speech Tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, pages 224–231, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi:10.3115/974147.974178.

2 Lou Burnard. Reference guide for the British national corpus (XML Edition), 2007. URL: http://www.natcorp.ox.ac.uk/XMLedition/URGbnctags.html.

3 Jean Carletta, Stefan Evert, Ulrich Heid, Jonathan Kilgour, Judy Robertson, and Holger Voormann. The NITE XML Toolkit: Flexible annotation for multimodal language data. *Behavior Research Methods, Instruments, & Computers*, 35(3):353–363, 2003. doi:10.3758/BF03195511.

4 Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. RST Discourse Treebank, 2002. LDC Catalog No.: LDC2002T07, ISBN, 1-58563-223-6.

5 Christian Chiarcos, Julia Ritz, and Manfred Stede. By all these lovely tokens... Merging Conflicting Tokenizations. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 35–43, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL: http://www.aclweb.org/anthology/W/W09/W09-3005.

6 Christian Chiarcos, Julia Ritz, and Manfred Stede. By all these lovely tokens... Merging conflicting tokenizations. *Language resources and evaluation*, 46(1):53–74, 2012.

7 James Clarke, Vivek Srikumar, Mark Sammons, and Dan Roth. An NLP Curator (or: How I Learned to Stop Worrying and Love NLP Pipelines). In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC-2012)*, Istanbul, Turkey, 2012. ELRA.

8 R. H. Dekker and G. Middell. Computer-Supported Collation with CollateX: Managing Textual Variance in an Environment with Varying Requirements. In *2nd Conference on Supporting Digital Humanities 2011 (SDH-2011)*, University of Copenhagen, Denmark, 2011.

9 Rebecca Dridan and Stephan Oepen. 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, July 2012. Association for Computational Linguistics. URL: https://www.aclweb.org/anthology/P12-2074.

10 Stefan Evert and Andrew Hardie. Twenty-first century Corpus Workbench: Updating a query architecture for the new millennium. In *Proceedings of Corpus Linguistics 2011 (CL2011)*, University of Birmingham, 2011.

11 Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. OntoNotes: The 90% Solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL: http://dl.acm.org/citation.cfm?id=1614049.1614064.

12 Adam Kilgarriff, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlỳ, and Vít Suchomel. The Sketch Engine: Ten years on. *Lexicography*, 1(1):7–36, 2014.

13 Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. doi:10.3115/1075812.1075835.

14 Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. Treebank-3, 1999. LDC Catalog No.: LDC99T42, ISBN, 1-58563-163-9.

15 Edward M. McCreight. A Space-Economical Suffix Tree Construction Algorithm. *J. ACM*, 23(2):262–272, 1976. doi:10.1145/321941.321946.

16 Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. The NomBank Project: An Interim Report. In *In Proceedings of the NAACL/HLT Workshop on Frontiers in Corpus Annotation*, 2004.

17 T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at International Conference on Learning Representations*, 2013.

18 Christoph Müller and Michael Strube. Multi-level annotation of linguistic data with MMAX2. In Sabine Braun, Kurt Kohn, and Joybrato Mukherjee, editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a.M., Germany, 2006.

19 Eugene W. Myers. AnO(ND) difference algorithm and its variations. *Algorithmica*, 1(1):251–266, 1986. doi:10.1007/BF01840446.

20 Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Comput. Linguist.*, 31(1):71–106, March 2005. doi:10.1162/0891201053630264.

21 Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. The Penn Discourse TreeBank 2.0. In *Proceedings, 6th International Conference on Language Resources and Evaluation*, pages 2961–2968, Marrakech, Morocco, 2008.

22 James Pustejovsky, Adam Meyers, Martha Palmer, and Massimo Poesio. *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*, chapter Merging PropBank, NomBank, TimeBank, Penn Discourse Treebank and Coreference, pages 5–12. Association for Computational Linguistics, 2005. URL: http://aclweb.org/anthology/W05-0302.

23 Michael Roth. Role Semantics for Better Models of Implicit Discourse Relations. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*, 2017. URL: http://aclweb.org/anthology/W17-6934.

24 Niko Schenk, Christian Chiarcos, Kathrin Donandt, Samuel Rönnqvist, Evgeny Stepanov, and Giuseppe Riccardi. Do We Really Need All Those Rich Linguistic Features? A Neural Network-Based Approach to Implicit Sense Labeling. In *Proceedings of the CoNLL-16 shared task*, pages 41–49. Association for Computational Linguistics, 2016. doi:10.18653/v1/K16-2005.

25 Carina Silberer and Anette Frank. Casting Implicit Role Linking as an Anaphora Resolution Task. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 1–10, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL: http://dl.acm.org/citation.cfm?id=2387636.2387638.

26 Mihai Surdeanu, Tom Hicks, and Marco Antonio Valenzuela-Escárcega. Two Practical Rhetorical Structure Theory Parsers. In *HLT-NAACL*, pages 1–5. The Association for Computational Linguistics, 2015.

27 Florian Wolf, Edward Gibson, Amy Fisher, and Meredith Knight. Discourse Graphbank, 2005. LDC Catalog No.: LDC2005T08, ISBN, 1-58563-320-8.

28 Kaoru Yamamoto, Taku Kudo, Akihiko Konagaya, and Yuji Matsumoto. Protein Name Tagging for Biomedical Annotation in Text. In Sophia Ananiadou and Jun'ichi Tsujii, editors, *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, pages 65–72, 2003.

29 Li Yujian and Liu Bo. A Normalized Levenshtein Distance Metric. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1091–1095, June 2007. doi:10.1109/TPAMI.2007.1078.

A Merging Discourse and SRL Structure – Sample CoNLL Merge

The following figure illustrates the merging of discourse and verbal predicate argument structure annotations of the same texts from two distinct resources. The resulting CoNLL format contains columns for tokens, character begin and end offsets, discourse meta data (blue), phrase structure (green), semantic roles (purple). Note, that *haven't* is treated as a single token in the PDTB2. The resulting output below contains *two* tokens with partial production rules (green) assigned to them.

```
Lawmakers  175...185  1:Arg1  (Explicit and,  Expansion.Conjunction);
                      3:Arg1  (Explicit but,  Comparison.Contrast.Juxtaposition)
                      NNS ( (S (S (NP-SBJ *)  _ ARG0 _ ARG0

have      B-185...193 B-1:Arg1 (Explicit and, Expansion.Conjunction);
                      3:Arg1   (Explicit but, Comparison.Contrast.Juxtaposition)
                      S-VBP (VP *             _ _ _ _

n't       E-185...193 E-1:Arg1 (Explicit and, Expansion.Conjunction);
                      3:Arg1   (Explicit but, Comparison.Contrast.Juxtaposition)
                      S-RB *                  _ S-ARGM-NEG _ _

publicly   193...202  1:Arg1   (Explicit and, Expansion.Conjunction);
                      3:Arg1   (Explicit but, Comparison.Contrast.Juxtaposition)
                      RB (VP ( ADV-MNR *)     _ ARGM-MNR _ _

raised     202...209  1:Arg2   (Explicit and, Expansion.Conjunction);
                      3:Arg1   (Explicit but, Comparison.Contrast.Juxtaposition)
                      VBN *                   raise.v.01 rel _ _
...
                                                        merged CoNLL
```

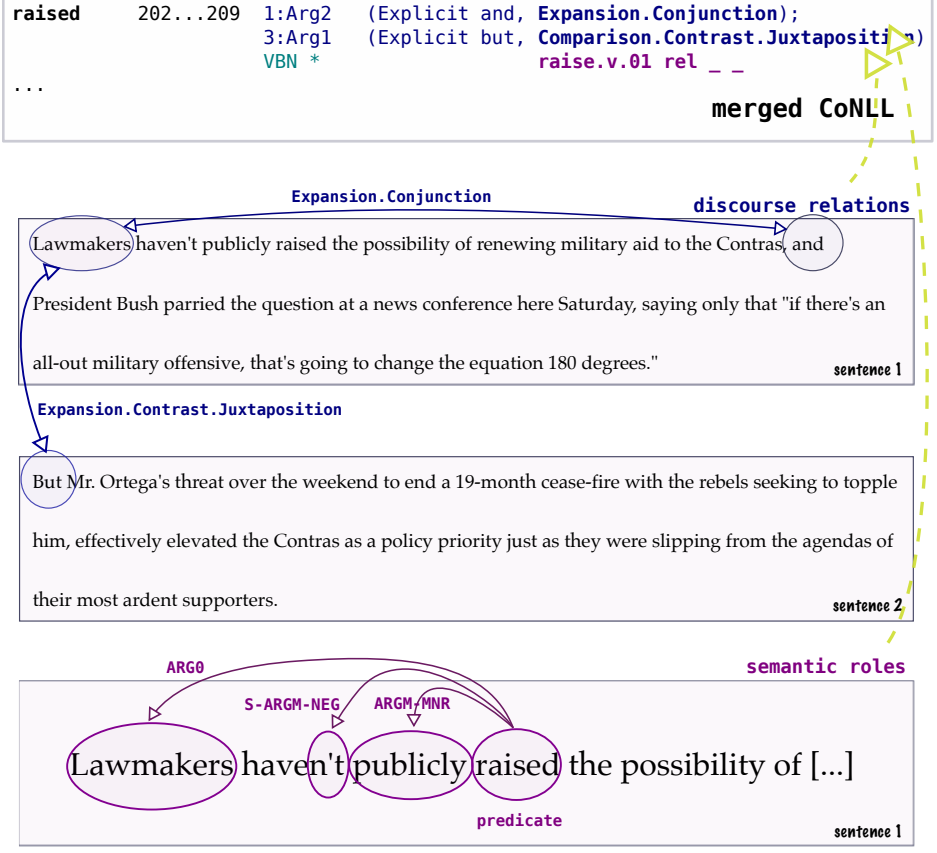

Figure 4 Merged CoNLL annotations (top) between PTB base source, PDTB2, and PropBank. The resulting output combines both phrase structure, **discourse structure** (blue) and **semantic roles** (purple). One word per line format expanded for better legibility.