# LLpro: A Literary Language Processing Pipeline for German Narrative Texts

**Anton Ehrmanntraut** and **Leonard Konle** and **Fotis Jannidis**
Julius-Maximilans-Universität Würzburg
{anton.ehrmanntraut,leonard.konle,fotis.jannidis}@uni-wuerzburg.de

## Abstract

In this paper, we motivate, describe, and evaluate LLpro[1], a novel NLP pipeline for German with the goal of laying the foundation for computational analysis of literary fiction. Our work is strongly inspired by BookNLP[2], which has a similar goal for English texts and has already shown its relevance through application in various research (e.g. Milli and Bamman, 2016). The pipeline consists of *fundamental NLP tasks* (tokenization, POS tagging, etc.) and *literary tasks* more tailored to narrative texts (e.g. scene segmentation, character recognition, detection of speech, thought, and writing representation, etc.). Building on the work of Ortmann et al. (2019) we present an updated evaluation of the *fundamental NLP tasks* and combine the most appropriate approaches with partially improved models for the *literary tasks* to create a rich representation of narrative fiction.

## 1 Introduction

'Distant Reading' (Moretti), the computational analysis of large collections of literary texts, has made progress in recent years, but is yet the province of the happy few who have literary expertise and sufficient knowledge about natural language processing and do know how to explore and analyze quantitative data. Especially the NLP part proves to be challenging, because the fast moving research in this field uses very modern techniques which are often hard to apply for someone not close to the rapid developments. At the same time, recent years saw a series of proposals how to extract specific features from literary texts, not only character references but events, scenes, speech renditions and more. The automatic detection of features like these is not part of generic pipelines like spaCy or Stanza. Our goal was to provide a pipeline which covers linguistic tasks like POS tagging, and tasks specific for narrative texts, like scene annotation.

We choose to build this on the basis of spaCy's pipeline framework and to use a Docker image to reduce the complexity of installing the components. The application covers *fundamental NLP tasks*: tokenization, lemmatization normalization, POS tagging morphological analysis and dependency parsing. For performance reasons we decided not to rely on spaCy even for these basic NLP tasks but integrate the best and the fastest solutions available for German texts.

Additionally, we integrate NLP applications that we specifically intend to use for literary analysis, performing the following linguistic tasks: named entity recognition, character mentions detection, coreference resolution, event classification, classification of speech, thought and writing representation, and scene segmentation. In the course of this paper, we will refer to these linguistic tasks as *literary tasks* to contrast them with the usually more simple and more widespread *fundamental tasks*. Concerning the *literary tasks*, we either incorporated published solutions, or improved on them by providing a LLM which has been adapted to the literary domain, or by re-implementing them.

## 2 Related Work

The NLP pipeline framework spaCy[3] (Honnibal et al., 2023) can be considered the de-facto default Python NLP processing pipeline for German text, also being one of the first to provide an integrated pipeline to process German text at all. The spaCy models were continuously improved, incorporating Transformer-based pipelines since 2021, and thus making state-of-the-art accuracies available in a simple and accessible interface. In the course of this paper, we will refer to version v3.5.2 (April 12, 2023) of spaCy, and to the German models `de_core_news_lg-3.5.0` based on word embeddings and focusing on speed,

[1] https://github.com/cophi-wue/LLpro
[2] https://github.com/booknlp/booknlp
[3] https://spacy.io



*Proceedings of the 19th Conference on Natural Language Processing (KONVENS 2023)*, pages 28–39
September 18–22, 2023. 

and `de_dep_news_trf-3.5.0`, based on Transformers.

SpaCy's default Transformer-based implementation for German `de_dep_news_trf-3.5.0` implements precisely the tasks we denote in this paper as *fundamental NLP tasks*, and hence we will particularly discuss LLpro's performance on these tasks with spaCy and other pipelines. Architecture-wise, the `de_dep_news_trf` pipeline, like all Transformer-based spaCy pipelines, consists of a single Transformer model to embed each token of the document into a contextualized vector representation, which has been fine-tuned to perform *multiple* tasks, implementing a multi-task learning.

As already sketched, we built upon spaCy's broad and tested APIs for components, pipeline architecture, and data structures, to implement LLpro, profiting from spaCy's easy and flexible extensibility.

While the alternative Python NLP processing toolkit Stanza[4] (Qi et al., 2020) is also designed to perform the *fundamental NLP tasks*, we found Stanza hard to extend to our purposes. First, extensions to Stanza are nontrivial to implement, and secondly, Stanza, by design, focuses on a language-agnostic modeling, building upon the Universal Dependencies formalism. This formalism distinguishes Stanza from other German NLP tools (usually following a German-specific grammar, not UD), which would have caused further difficulties in adapting tools.

The best-known example of the combination of *fundamental NLP tasks* with components specifically targeted at literary texts in one pipeline is BookNLP.[5] Besides the *fundamental NLP tasks*, BookNLP provides NER, coreference resolution, speaker identification, supersense tagging, event tagging and referential gender inference.[6] LLpro covers most of the functionality of BookNLP, only supersense[7] and speaker detection are missing, but also introduces new tasks (scene segmentation, classification of speech, thought and writing representation).

In its current state, BookNLP can only process English language; a further development to a multilingual tool, including support for German, is planned, but not yet available. BookNLP, like LLpro, is built on spaCy infrastructure, so transferring or exchanging modules between pipelines will be facilitated once a German BookNLP version is available.

Finally, the Python NLP pipeline MONAPipe[8] (Dönicke et al., 2022) also extend spaCy to more specialized tasks in the analysis of German literary texts.

While both the MONAPipe and the presented LLpro are intended for the literary texts, the choice of *literary tasks* the respective pipelines perform, are different. As MONAPipe particularly focuses on modes of narration and attribution, it performs a dictionary-based semantic analysis of phrases to enrich a feature set intended to identify the narrative mode 'comment' (in contrast to the narrative modes 'description', 'report' and 'speech'). From the same set of features, MONAPipe attributes each clause to one of 'character', 'author', and/or 'narrator'. (Cf. Weimer et al., 2022; Dönicke et al., 2022) Like Stanza, MONAPipe decided to build upon Universal Dependencies (and in particular trained a new UD-based spaCy parser), since some of its downstream modules require UD parses.

In contrast, LLpro's exclusive components focus around literary characters, and in particular includes a coreference resolution model with state-of-the-art performance, much stronger and more scalable than the one included in MONAPipe, and is the only one of the discussed pipelines that can perform a segmentation into scenes, and can recognize references to literary characters. Furthermore, concerning the *fundamental NLP tasks*, MONAPipe relies on the provided spaCy models, unlike LLpro which provides wrappers for other NLP tools performing *fundamental tasks*. Finally, since MONAPipe is based on spaCy v2.3, it is unable to use the more accurate Transformer-based spaCy models, and can only run the less accurate word-embedding-based models for the *fundamental tasks*.

[4] `https://stanfordnlp.github.io/stanza/`. We refer to version v1.5.0, March 14, 2023.

[5] `https://github.com/booknlp/booknlp`; we refer to version v1.0.7, commit 2b42ccdk, December 4, 2021.

[6] Gender inference is a postprocessing step, which maps the usage of pronouns to coreference clusters. Certainly useful, we decided, partly due to a lack of evaluation data, to leave this postprocessing to users

[7] The supersense detection component builds upon WordNet (Fellbaum, 2005; `https://wordnet.princeton.edu/`). While GermaNet (Hamp and Feldweg, 1997; `https://uni-tuebingen.de/en/142806`) mirrors WordNet for German, it is still much smaller and differs in its supersense ontology. After a review, we conclude that the direct benefit of the supersenses present in GermaNet, without further refinement, for the analysis of literary texts has yet to be tested more thoroughly.

[8] `https://gitlab.gwdg.de/mona/pipy-public`. We will refer to version v3.2.

Finally, we want to remark that *locally* executed pipelines (such as spaCy, LLpro, Stanza, or BookNLP) are not the only option to design NLP pipelines. For instance, WebLicht[9] (Hinrichs et al., 2010) follows a service-oriented architecture, chaining together multiple distributed and independet web services, each performing an individual *fundamental* NLP task. These services are hosted online by different providers and not locally, enabling pipeline composition and execution through a browser-based interface. This makes usage far more accessible.

However, in light of the increasing computational effort associated with some tasks (in particular the *literary tasks*), such architecture also has limitations. Extending, e.g., WebLicht's functionality requires independent and reliable hosting of additional services. Moreover, scaling to larger corpora may be challenging as it relies on external providers' compute power, which could have limitations or usage restrictions.

## 3 Architecture and Pipeline Components

As already outlined, LLpro is built on top of the open-source spaCy (v3.5.2) API using Python. SpaCy provides a programming interface and trained models to individually compose a language processing pipeline for one's use case, building on top of their provided data structures that manages the document, the tokens and the annotations on these objects. Invoked on an input document, spaCy first calls the specified tokenizer that segments the input text into tokens, converting the text to a document object, consisting of all the token objects. Then, in the subsequent steps, the document object is processed by the specified components of the pipeline, each enriching the document object with information that is annotated on the individual token objects, on spans of tokens, or on the entire document.

Now, LLpro's key contribution consists of implementations of pipeline components for the spaCy API, providing wrappers of already existing NLP tools designed to process German text. In particular, LLpro, for one, provides alternative components for the previously mentioned *fundamental tasks* that spaCy (and Stanza) can already do, but with higher accuracy and/or speed. We primarily grounded our choice of tools in the previously mentioned study by Ortmann et al. (2019), selecting the most promising ones.

Secondly, LLpro contributes by implementing new pipeline components that provide access to novel NLP tools that perform specific NLP tasks useful in the field of literary analysis. Table 1 provides an overview of the implemented components, which are discussed below.

Notice, moreover, that while the default pipeline implemented by LLpro can perform all of its tasks without any of spaCy's models or components, the modular structure of spaCy's API allows all components to be replaced or omitted in a custom pipeline, if desired. For instance, instead of the probabilistic parser presented here, it is possible, in a correspondingly custom-programmed pipeline, to switch back to the Transformer-based parser trained by spaCy.

In the remainder of this section, we briefly describe each component LLpro implements.

### 3.1 Preprocessing and Basic Components

With the **SoMaJoTokenizer** we wrapped the rule-based tokenizer / sentence splitter SoMaJo[10] (Proisl and Uhrig, 2016) as component for spaCy. Additionally, we implemented a simple normalization to correct for historic characters, which otherwise would cause wrong inferences in the successive components. We replace the historic notation of umlauted vowels (superscript E) with contemporary notation (with diaeresis), followed by NFKC Unicode normalization. This has also the effect that long S characters get converted to short S characters. Note that this simple form of normalization does not address for orthographic differences, for instance *selbstthätig*, *seyn* (vs. *selbsttätig*, *sein*).

The **SoMeWeTaTagger** invokes the part-of-speech tagger SoMeWeTa[11] (Proisl, 2018). For LLpro, we use the 'newspaper' model based on the TIGER corpus.[12] Next to the predicted tags (as defined by the TIGER variant of the German STTS tagset, cf. Smith, 2003), the component also provides an automatic table-based conversion[13] to the Universal Dependencies v2 POS tagset (de Marneffe et al., 2021).[14]

[9] https://weblicht.sfs.uni-tuebingen.de/. We want to thank the anonymous reviewer to bring WebLicht to our attention.

[10] https://github.com/tsproisl/SoMaJo

[11] https://github.com/tsproisl/SoMeWeTa

[12] https://corpora.linguistik.uni-erlangen.de/someweta/german_newspaper_2020-05-28.model

[13] https://universaldependencies.org/tagset-conversion/de-stts-uposf.html

[14] See also https://universaldependencies.

| Component | Task, Tagset if applicable | Reference(s) | Version |
|---|---|---|---|
| | **Fundamental Tasks** | | |
| • SoMaJoTokenizer | Tokenization, Normalization, Sentence Splitting | Proisl and Uhrig, 2016 | 2.2.3 |
| • SoMeWeTaTagger | Part-of-speech tagging;<br>TIGER variant of the STTS tagset | Proisl, 2018 | 1.8.1; model May 28, 2020 |
| • RNNTagger | Morphological analysis;<br>Universal features inventory | Schmid, 2019 | 1.3 |
| • RNNLemmatizer | Lemmatization | Schmid, 2019 | 1.3 |
| • ParZuParser | Dependency Parsing; HDT tagset | Sennrich et al., 2009 | Feb 11, 2022 |
| | **Literary Tasks** | | |
| FLERTNERTagger | Named entity recognition;<br>PER, ORG, MISC, LOC | Schweter and Akbik, 2021 | 0.12.2 |
| CorefTagger | Coreference Resolution | Schröder et al., 2021 | Aug 31, 2021 |
| EventClassifier | Annotates *event types* to verbal phrases; differentiates between *non-event*, *stative* event, *process* event, and *change of state* event | Vauth et al., 2021 | 0.2 |
| ∗ RedewiedergabeTagger | Tagging of German speech, thought and writing representation (STWR); recognizes direct, indirect, reported and free indirect speech<br>cf. Brunner et al., 2020 | Schweter and Akbik, 2021 | — |
| ∗ CharacterRecognizer | Recognizes references to literary characters<br>cf. Krug et al., 2017 | Schweter and Akbik, 2021 | — |
| ∗ SceneSegmenter | Segmentation of literary text into *scenes* and *non-scenes*, cf. Zehe et al. (2021a,b) | Kurfalı and Wirén, 2021 | — |

Table 1: Overview of LLpro's components. Each component marked with • provides a replacement for a spaCy component performing the same task. Each component marked with ∗ has been (re-)implemented and (re-)trained from scratch.

The **RNNTagger** and **RNNLemmatizer** provide a wrapper for the RNNTagger[15] tool (Schmid, 2019) to perform a morphological analysis and lemmatize the tokens. To be consistent with spaCy's API, we convert the output of the tagger into an equivalent form consisting of Universal Dependencies v2 features (de Marneffe et al., 2021).[16]

The **ParZuParser** is a wrapper for the Prolog-based probabilistic dependency parser ParZu[17] (Sennrich et al., 2009; Sennrich and Kunz, 2014). For each token, the component predicts the head token and the respective relation as specified by the grammar of the Hamburg Dependency Treebank (Foth, 2014). Note that this labeling scheme of relations differs from the one used by spaCy's default models, which is trained on a (semi-automatically derived) dataset based on the TIGER corpus/tagset (Smith, 2003). SpaCy's API and subsequent components that build on top of relation labels have been configured accordingly to match the changed labeling scheme.

### 3.2 Components for Literary Analysis

The following subsection discusses the remainder of LLpro's components, i.e. the *literary NLP tasks*, which particularly perform tasks intended for literary analysis. Since many of the the tasks resp. annotations are not represented in spaCy's data structures, we use the provided "extension attributes" to store the components' results. A full specification of the exposed extension attributes is provided in LLpro's documentation.

In some instances, we (re-)implemented and (re-)trained models to adapt them to our domain. For this, we have domain-adapted the `deepset/gbert-large` BERT model (Chan et al., 2020) with literary texts to obtain `fiction-gbert-large`, which we make available. Details are presented in Sec. A.1 in the Appendix.

The **FLERTNERTagger** invokes the NER tagger FLERT from the Flair[18] framework,

org/u/pos/all.html

[15] https://www.cis.uni-muenchen.de/~schmid/tools/RNNTagger/

[16] See also https://universaldependencies.org/u/feat/all.html

[17] https://github.com/rsennrich/ParZu

[18] https://github.com/flairNLP/flair

which builds upon a BERT-based sequence tagging (Schweter and Akbik, 2021). For LLpro, we use the publicly available Flair model `ner-german-large`.[19] Note that while some models of spaCy include a NER tagger, spaCy misses a Transformer-based one like FLERT. The tagger annotates non-overlapping named entity spans as one of the four CoNLL-03 classes (PER, LOC, ORG, MISC; cf. Tjong Kim Sang and De Meulder, 2003). While the tagger has issues in recognizing *characters* in literary texts (see below), we keep the FLERTNERTagger primarily to recognize locations and organizations.

The **CharacterRecognizer** attempts to resolve a conceptional issue arising with determining mentions of characters in literary texts. In literary texts, character references to an entity appear not only as (1) proper nouns (e.g., *Alice*, *Effi Briest*), but also as (2) nominal phrases, e.g. *gardener, mother, Earl, Lieutenant, idiot, beauty, ....*

While the mention of type (1) are theoretically *named entities* in the sense of an NER tagger, mentions of type (2) are not, therefore not recognized by the FLERTNERTagger. Furthermore, the NER tagger was primarily trained on newspaper articles, implying another domain gap (cf. Krug et al., 2017).

To resolve this, we trained a tagger that recognizes character mention spans of both type (1) and (2), using the DROC corpus (Krug et al., 2017) which annotated character references in German novels, employing the same Transformer-Linear architecture as used in the FLERTNERTagger, fine-tuning our custom BERT model `fiction-gbert-large`. The tagger makes no distinctions between these two types, thus recognizes combined variants such as *Ritterschaftsrätin von Padden* (*knighthood councilor von Padden*).

The **CorefTagger** provides coreference resolution by invoking the neural tagger developed by Schröder et al. (2021).[20] Most importantly, the tool implements an incremental entity-based approach that scales to very long documents such as the literary works we want to process. Also, the model is adapted to our literary domain, as it is fine-tuned and tested on the literary DROC (Krug et al., 2017) dataset. For LLpro, we use the publicly available model `droc_incremental_no_segment_distance`.[21]

The **EventClassifier** invokes a neural sequence classifier developed by Vauth et al. (2021).[22] The authors model the event structure of literary texts using narratological event concepts, and their classifier automatically recognizes these events. In particular, they opt to model events as only occurring in verbal phrases. Their model then categorizes each of the phrases as either 'changes of state', 'process event', 'stative event' or 'non-event'.

To automatically recognize these event types, their proposed classifier automatically extracts verbal phrases from the text using the syntactic structure inferred by a parser (in their case: spaCy's parser), and then classifies phrases using a Transformer-based architecture. For LLpro, we use their publicly available model.[23] We incorporate this tagger by instead re-using the syntactic structure predicted by the previously mentioned ParZuParser, and then invoking the Transformer model for classification on the extracted verbal phrases.

The **RedewiedergabeTagger** is a re-implementation of four taggers proposed by Brunner et al. (2021) that use neural representations to recognize four different types of speech, thought and writing representation (STWR) for German texts. The four types of STWR are 'direct', 'indirect', 'free indirect', and 'reported'. They approach this kind of classification by developing four different sequence taggers for each STWR type, each effectively performing a binary classification for each token in the sequence, building on a BiLSTM-CRF architecture on top of a chosen language embedding derived from Transformer models.

For LLpro, we re-implemented these models, and specifically fine-tuned the aforementioned `fiction-gbert-large` on the respective tasks using the same REDEWIEDERGABE corpus. (Brunner et al., 2020) As proposed by Schweter and Akbik (2021), we omit the additional LSTM/CRF and predict the respective STWR type from the token encoding in the final Transformer layer alone, following the Transformer-Linear variant that is also used in the NER tagging of above FLERTNERTagger.

[19] https://huggingface.co/flair/ner-german-large

[20] https://github.com/uhh-lt/neural-coref

[21] https://github.com/uhh-lt/neural-coref/releases/tag/konvens

[22] https://github.com/uhh-lt/event-classification

[23] https://github.com/uhh-lt/event-classification/releases/tag/v0.2

| Pipeline | Tokens | Sents | POS | UPOS | Lemmas | Morph | Deps |
|---|---|---|---|---|---|---|---|
| spaCy, `de_core_news_lg-3.5` | 0.9953 | 0.9091 | 0.9465 | 0.9270 | 0.9062 | 0.9149 | 0.6942 |
| spaCy, `de_dep_news_trf-3.5` | 0.9953 | 0.8936 | **0.9635** | 0.9320 | 0.9181 | **0.9508** | 0.7573 |
| Stanza 1.5 | 0.9975 | 0.9784 | 0.9433 | 0.9144 | 0.8778 | 0.9045 | **0.7578** |
| LLpro | **1.0000** | **1.0000** | 0.9458 | **0.9610** | **0.9188** | 0.9372 | 0.7425 |

Table 2: Evaluation of different NLP pipelines on the *fundamental NLP tasks* using the adapted evaluation system by Ortmann et al. (2019) against the gold annotations of the *novelette* text. For columns *Tokens* and *Sents*, metric is F1, comparing the output from raw text input with the gold tokenization/sentencization. In all other columns, metric is accuracy, comparing the output from (gold) pre-tokenized input. Evaluation only run on the novelette text. The column UPOS refers to the universal dependencies POS tags, which are predicted alongside the fine-grained POS tagging in each pipeline.

The **SceneSegmenter** is a re-implementation of a tool by Kurfalı and Wirén (2021) that recognizes contiguous and non-overlapping *scenes* resp. *non-scenes*. In short, a *scene* is "a segment of a text where the story time and the discourse time are more or less equal, the narration focuses on one action and space and character constellations stay the same" (Zehe et al., 2021a), whereas a *non-scene* refers to a non-scenic bridge between scenes like reflections of the narrator or accelerated speed of narration. See the shared task description resp. formal definition (Zehe et al., 2021b,a) for details on scene segmentation task. The model by Kurfalı and Wirén showed best performance in the shared task Track 1 that evaluated on gold annotations in dime novels. The tool adapted the sequential sentence classification system proposed by Cohan et al. (2019) to the scene segmentation task. Similar to the previously mentioned RedewiedergabeTagger, we re-trained the model on our domain-adapted custom BERT model.

We will discuss the results of this re-implementation and re-training of the three preceding components in Section 4.2. Details on the training of each of the models is provided in the Appendix, as well as links to the model weights.

## 4 Evaluation

Concerning the *fundamental tasks*, we focus on a comparative discussion of LLpro's components with the equivalent components of spaCy and Stanza. For this, firstly, we compare the annotation *accuracies* using human-labeled data provided by Ortmann et al. (2019), and secondly, measure and compare the runtimes of these components to estimate their (computational) *efficiency*.

Concerning the *literary tasks*, we are unable to compare their accuracies with respect to other NLP pipelines, since, in most cases, they are not implemented in any pipeline system. Therefore, we restrict ourselves to a qualitative analysis, discussing the performance of the underlying NLP systems that LLpro's components wrap around. Besides this, we provide quantitative results on the effect of our re-implementing/re-training on the CharacterRecognizer, RedewiedergabeTagger, SceneSegmenter building on the `fiction-gbert-large` model.

### 4.1 Accuracy on Fundamental Tasks

In order to compare the accuracies of the respective components, we opted to follow the evaluation system developed by Ortmann et al. (2019) that was specifically designed to compare NLP tools performing the NLP tasks tokenization, POS tagging, lemmatization, morphological analysis, and dependency parsing.

The evaluation system consists, in the first part, of five human-labeled documents from different registers. In the second part, the evaluation system consists of a comparison procedure that evaluates a tool's output with the gold label, and in particular, accounts for different naming/annotation schemes between different NLP tools.

For our evaluation, we take over this comparison procedure, but will primarily focus on the one human-labeled *novelette* document (1588 tokens), which was chosen by Ortmann et al. as representative for the literary register. Note that in the original evaluation, pipelines like spaCy were not evaluated as a whole, but only the individual components. For instance, the spaCy dependency parser was provided with (gold) POS annotations as input in the evaluation.

We deviate from this and want to compare the different pipelines in a way that imitates an end-to-end use. To this end, we performed two experiments: first, to compare the different tokenizers,

we compare the tokenizers' outputs from raw text with the gold tokenization. Second, to compare all the other downstream components of the components, but controlling for potentially incorrect tokenization, we compare the pipelines' outputs derived from (gold) pre-tokenized input with the gold labels. This means that, e.g., we evaluate how LLpro's parser performs even when given inaccurate POS tagged text from LLpro's POS tagger.

Table 2 gives the evaluation results on the LLpro, spaCy and Stanza pipelines on the *novelette* text. Columns *Tokens*, *Sents* refer to the accuracy on the first experiments; the subsequent columns refer to the second experiment. (See Table 6 in the Appendix for the aggregated results on all five texts.)

Concerning the accuracy of LLpro, we observe that LLpro is competitive with contemporary Stanza and Transformer-based spaCy models, and even slightly outperforming these pipelines in some tasks.

### 4.2 Accuracy on Literary Tasks

LLpro's components perform the *literary NLP tasks*, either by wrapping around previously developed systems, all of which can be generally considered state of the art in their respective fields, or by running our custom fine-tuned models for the tasks.

Concerning NER tagging, the model used in LLpro's component FLERTNERTagger is published with a reported CoNLL-F1 of 0.92 on the CoNLL-03 German revisited test set. With this high accuracy, we do not expect significant improvements by fine-tuning from our domain-adapted BERT model, and consider the task practically solved for our use-case with this NER tagger. As a comparison, the alternative NER tagger provided by Stanza showed worse performance than Flair's tagger (CoNLL-F1 81.9).

Coreference resolution, as it is done by the CorefTagger by invoking a model by Schröder et al. (2021), is known to be a notoriously hard task. With this background, the (not very impressive-looking) performance of their model on the literary DROC dataset (CoNLL-F1 0.65) can be considered extremely strong. See also the survey and experiments by Dönicke et al. (2022) concerning coreference resolution in German.

The remaining *literary NLP tasks* – event classification, tagging of STWR, tagging of character references, and segmentation into scenes – were introduced only very recently, and in all cases, almost no other models appear to approach the respective tasks, making a comparative analysis impossible in most cases. Concerning the component for the first task (EventClassifier), we remark that the classifier designed by Vauth et al. (2021), which LLpro invokes for event classification, should explicitly only be understood as a qualitative indicator: in particular, the tests performed by Vauth et al. to evaluate their model with respect to unseen documents was primarily *visual*, comparing the resulting "narrativity graphs" between predicted event spans and gold-annotated event spans. These graphs can be understood as smoothed time series of "narrativity" assigned to each type of event. In total, the authors observe a sufficient match between the predicted and the gold-derived narrativity graphs, and conclude applicability of their model in corpus analysis.

Concerning the other tasks, we have opted to re-implement and re-train models for each task on our domain-adapted BERT model `fiction-gbert-large`.

For the character recognition (CharacterRecognizer), our simple Transformer-based model resulted in an F1-score of 0.91 on a held-out test dataset from the DROC corpus. With this accuracy, we find this model sufficient for our use-case. Additionally, no other model that performs such tasks is known to us.

| Model | Direct | Ind. | Rep. | Fr. Ind. |
|---|---|---|---|---|
| Brunner et al. (2021) | 0.84 | 0.76 | 0.58 | 0.57 |
| RedewiedergabeTagger | 0.91 | 0.79 | 0.70 | 0.58 |

Table 3: Scores on STWR recognition on the held-out test set of the REDEWIEDERGABE corpus. F1 in all cases.

| Model | Prec. | Rec. | F1 |
|---|---|---|---|
| Track 1 | | | |
| Kurfalı and Wirén (2021) | 0.29 | 0.51 | 0.37 |
| SceneSegmenter | 0.37 | 0.44 | 0.40 |
| Track 2 | | | |
| Kurfalı and Wirén (2021) | 0.14 | 0.22 | 0.17 |
| SceneSegmenter | 0.32 | 0.40 | 0.35 |

Table 4: Scores on the Shared Task on Scene Segmentation (not publicly released) test set, Tracks 1 (dime novels) and Track 2 (out-of-domain high-brow literature). Results for Kurfalı and Wirén (2021) cited from the task report (Zehe et al., 2021b). Results for our model are reported by the task organizers.

| | Cores | | | |
|---|---|---|---|---|
| **Pipeline** | **4** | **8** | **16** | **32** |
| spaCy, `de_dep_news_trf-3.5` | 62.59 | 57.18 | 48.43 | 36.55 |
| Stanza 1.5 | 156.4 | 109.7 | 55.91 | 23.04 |
| LLpro, *fundamental tasks* only | 151.3 | 73.00 | 48.54 | 20.27 |
| LLpro, all tasks | 5.025 | 3.152 | 2.959 | 1.540 |

Table 5: Number of tokens processed per core in one second, under different intra-op parallelizations.

For the STWR recognition (RedewiedergabeTagger), our fine-tuning was able to increase the models' accuracies on the STWR task, except for the *free indirect* STWR type. See Table 3 for an overview and a comparison to the models by Brunner et al. (2021). Note that the increase is most likely explained by the different model architecture and fine-tuning procedure, which is missing in the original models.

Concerning the scene segmentation, note again that we based our SceneSegmenter on a re-implementation of a sequential sequence classification model by Kurfalı and Wirén (2021), which was the best-performing contribution in the scene segmentation shared task. Our re-implementation directly takes over this architecture and ports it to the contemporary Pytorch/Transformers API (Wolf et al., 2020) with minimal modifications. To evaluate our model, the organizers of the Shared Task on Scene Segmentation (Zehe et al., 2021b) evaluated our model on the test datasets, on both Track 1 (dime novels) and Track 2 (out-of-domain high-brow novels). The fine-tuning was able to increase the model's F1 score by few percentage points in Track 1, with respect to the original model published by Kurfalı and Wirén. See Table 4 for an overview. Also, our model appears to generalize much better to the out-of-domain Track 2. Note that both our model and the one by Kurfalı and Wirén build upon the 'large' variant of the BERT model, hence the difference in performance can be attributed to the domain-adaption of our `fiction-gbert-large` model.

### 4.3 Computational Efficiency

As we intend to process a large corpus of literary texts with LLpro, we are also interested in their computational efficiency, next to accuracy. In our CPU-only setup with many cores, it is immediately clear that the computational effort required by LLpro will be dominated by the slow Transformer-based components, performing the *literary NLP tasks*. Even with this in consideration, we will briefly discuss our experiments concerning the computational efficiency of our pipeline. In our case, we are particularly interested in *throughput* – the number of tokens we can process per second *and per core*. This delimits our investigation to previous studies, like already mentioned one by Ortmann et al. (2019), that were focused on *latency*, keeping the computational setup fixed.

Table 5 shows the measured throughput of the different pipelines, all restricted to performing the *fundamental tasks* only. In the case of LLpro, we additionally provide the throughput of the full pipeline, including the (computationally much more expensive) *literary tasks*. Measurements were performed by repeated trial runs on Intel Xeon Gold 6148 cores, varying number of cores, and varying length of input documents.

While the results confirm what we already assumed – LLpro with all components is slow in CPU-only setups – we can take away two things from these measurements: first, we see that the tools we use for the *fundamental NLP pipelines* are in some setups much more efficient overall than those (Transformer-based) models of spaCy, while performing equally well accuracy-wise. Second, the experiment indicates that for maximum efficiency of Transformer-based pipelines like LLpro (running all tasks), the appropriate parallelization and partitioning of the available CPU cores still remains an important ingredient, potentially increasing throughput a factor of 3.

## 5 Conclusion and Future Work

In this report, we present LLpro, a custom spaCy pipeline that provides components for the linguistic and literary analysis of German texts. On the side of linguistic analysis, LLpro provides wrappers to alternative NLP tools that perform tokenization, part-of-speech tagging, morphological analysis, lemmatization, and dependency parsing (*fundamental NLP tasks*). On the side of literary anal-

ysis (*literary NLP tasks*), LLpro implements several components that perform novel tasks currently not found in spaCy or other comparable pipelines: coreference resolution, named entity recognition, event classification, tagging of speech, thought and writing representation types, character reference recognition, and segmentation into scenes.

For the first part of components, the *fundamental NLP tasks*, our evaluation shows that our alternative models are, accuracy-wise, competitive with current spaCy, and in some setups, perform their tasks more efficient, particularly when bulk processing many texts. This comparative analysis also continues a research direction started by Ortmann et al. (2019) who evaluated many off-the-shelf NLP tools performing the *fundamental NLP tasks*, effectively giving an update of their evaluation with respect to the contemporary Transformer-based German spaCy model. While spaCy made significant improvement since the last evaluation by Ortmann et al. in 2019, our experiments showed that spaCy (and Stanza) still do not significantly outperform some specialized NLP tools. Furthermore, our analysis broadens the analysis of these NLP tools in terms of their computational efficiency. In total, our evaluation points out that for many simple linguistic NLP tasks, more lightweight models might be a suitable alternative to larger Transformer-based models, being more efficient without sacrificing accuracy.

For the *literary NLP tasks*, LLpro provides an accessible pipeline to perform automatic literary analysis by incorporating specialized Transformer-based models, reaching accuracies that make LLpro a novel basis for quantitative literary analysis on many texts. We can conceive that the outputs of the pipeline can be combined to investigate specific questions, for instance combining the scene segmentation and the character recognizer to carry out a fine-grained variant of a character network analysis. Or, use the coreference resolution, combined with the character recognizer and the parse trees, to collect attributes and adjectives that describe a particular character, or character's actions. LLPro thus provides a a robust basis for the automatic analysis of collections of German fiction.

## Limitations

The most obvious limitation of LLpro is the restriction to German language. But since one motivation for this work is the limitation of BookNLP to English, we already consider LLpro as a step towards multilinguality in the analysis of literary texts. This is further highlighted by the plans to extend BookNLP to other languages and the spaCy architecture as the backbone of both systems.

The second restriction refers to the domain for which LLpro can be applied. We focus on narrative texts (novels, short stories, etc.) and thus exclude other literary genres (e.g. plays, poems). Since we offer only a very basal orthographic normalization, a drastic performance loss is to be expected when processing older texts. However, the analysis of large corpora over long periods of time is a central concern of Computational Literary Studies. Therefore normalization is a requirement we need to address in future work. Especially in light of the short novelette text used in our reported experiments, a larger evaluation corpus for all tasks would be mandatory for accurate in-domain evaluation, as well as further experimentation to improve the components.

Thirdly it is plausible that improvements in spaCy's Transformer-based pipeline could significantly outperform our *fundamental* NLP components in the near future, due to its capability to exploit multi-task learning, while relying on a single Transformer model. This Transformer model is fine-tuned to a multitude of NLP tasks, allowing, for one, faster inference as the embedding needs to be computed only once. For another, as soon as better Transformer models for German are released, instant performance gains are to be expected. To address these developments while mitigating the dependence on GPU resources for fast inference, we plan to make LLpro Adapter-based (Pfeiffer et al., 2021; Hu et al., 2021). This should at least drastically reduce the computational effort for the *literary NLP tasks*, ensure SOTA competing performance on *fundamental tasks* and enable more lightweight domain adaptation. Still however, like all NLP pipelines, LLpro faces the challenge of potential tool obsolescence and the need for sustainable maintenance and ongoing development, in order to maintain long-term viability and competitiveness.

## Ethics Statement

We do not see any conflict of our work with the principles set out in the ACL Ethics Policy[24]. The

[24] https://www.aclweb.org/portal/content/acl-code-ethics

purpose of LLpro is to create a rich representation of literary texts. These texts may contain structural discrimination, which is therefore also present in the output of LLpro. That is not a problem, but an opportunity to systematically uncover and investigate them.

However, such a research perspective requires that the components of the pipeline operate without bias. We are not aware of any anecdotal evidence of biased behavior, but since this has not been systematically investigated for any of the modules, there is at least a possibility that e.g. coreference clusters of female characters are resolved less accurate.

## References


Annelen Brunner, Stefan Engelberg, Fotis Jannidis, Ngoc Duyen Tanja Tu, and Lukas Weimer. 2020. Corpus REDEWIEDERGABE. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 803–812, Marseille, France. European Language Resources Association.

Annelen Brunner, Ngoc Duyen Tanja Tu, Lukas Weimer, and Fotis Jannidis. 2021. To BERT or not to BERT – Comparing contextual embeddings in a deep learning architecture for the automatic recognition of four types of speech, thought and writing representation. In *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS)*, volume 2624 of *CEUR Workshop Proceedings*, Zurich, Switzerland.

Branden Chan, Stefan Schweter, and Timo Möller. 2020. German's next language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6788–6796, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Arman Cohan, Iz Beltagy, Daniel King, Bhavana Dalvi, and Dan Weld. 2019. Pretrained language models for sequential sentence classification. 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 3693–3699, Hong Kong, China. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Tillman Dönicke, Florian Barth, Hanna Varachkina, and Caroline Sporleder. 2022. MONAPipe: Modes of narration and attribution pipeline for German computational literary studies and language analysis in spaCy. In *Proceedings of the 18th Conference on Natural Language Processing (KONVENS 2022)*, pages 8–15, Potsdam, Germany. KONVENS 2022 Organizers.

Tillmann Dönicke, Hanna Varachkina, Anna Mareike Weimer, Luisa Gödeke, Florian Barth, Benjamin Gittel, Anke Holler, and Caroline Sporleder. 2022. Modelling speaker attribution in narrative texts with biased and bias-adjustable neural networks. *Frontiers in Artificial Intelligence*, 4.

Christiane Fellbaum. 2005. Wordnet(s). In Keith Brown, editor, *Encyclopedia of Language and Linguistics*, second edition, pages 665–670. Elsevier.

Kilian A. Foth. 2014. Eine umfassende Constraint-Dependenz-Grammatik des Deutschen. Universität Hamburg.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Birgit Hamp and Helmut Feldweg. 1997. GermaNet - a lexical-semantic net for German. In *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*.

Erhard Hinrichs, Marie Hinrichs, and Thomas Zastrow. 2010. WebLicht: Web-based LRT services for German. In *Proceedings of the ACL 2010 System Demonstrations*, pages 25–29, Uppsala, Sweden. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2023. spaCy: Industrial-strength Natural Language Processing in Python. Supplement to `https://github.com/explosion/spaCy/tree/v3.5.2`.

Edward Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. ArXiv:2106.09685.

Markus Krug, Lukas Weimer, Isabella Reger, Luisa Macharowsky, Stephan Feldhaus, Frank Puppe, and Fotis Jannidis. 2017. Description of a corpus of character references in German novels – DROC [Deutsches ROman Corpus]. DARIAH-DE Working Papers 27.

Murathan Kurfalı and Mats Wirén. 2021. Breaking the narrative: Scene segmentation through sequential sentence classification. In *Proceedings of the Shared Task on Scene Segmentation*, volume 3001 of *CEUR Workshop Proceedings*, pages 49–53, Düsseldorf, Germany.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019.

RoBERTa: A robustly optimized BERT pretraining approach. ArXiv:1907.11692.

Smitha Milli and David Bamman. 2016. Beyond canonical texts: A computational analysis of fanfiction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2048–2053, Austin, Texas. Association for Computational Linguistics.

Katrin Ortmann, Adam Roussel, and Stefanie Dipper. 2019. Evaluating off-the-shelf NLP tools for German. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)*, pages 212–222, Erlangen, Germany. German Society for Computational Linguistics & Language Technology.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Thomas Proisl. 2018. SoMeWeTa: A part-of-speech tagger for German social media and web texts. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 665–670, Miyazaki, Japan. European Language Resources Association ELRA.

Thomas Proisl and Peter Uhrig. 2016. SoMaJo: State-of-the-art tokenization for German web and social media texts. In *Proceedings of the 10th Web as Corpus Workshop*, pages 57–62, Berlin. Association for Computational Linguistics.

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

Helmut Schmid. 2019. Deep learning-based morphological taggers and lemmatizers for annotating historical texts. In *DATeCH, Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, pages 133–137, Brussels, Belgium. Association for Computing Machinery.

Fynn Schröder, Hans Ole Hatzel, and Chris Biemann. 2021. Neural end-to-end coreference resolution for German in different domains. In *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)*, pages 170–181, Düsseldorf, Germany. KONVENS 2021 Organizers.

Stefan Schweter and Alan Akbik. 2021. FLERT: Document-level features for named entity recognition. arXiv: 2011.06993.

Rico Sennrich and Beat Kunz. 2014. Zmorge: A German morphological lexicon extracted from Wiktionary. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*, pages 1063–1067, Reykjavik, Iceland. European Language Resources Association (ELRA).

Rico Sennrich, Gerold Schneider, Martin Volk, and Martin Warin. 2009. A new hybrid dependency parser for German. In *Proceedings of the 2009 GSCL Conference*, pages 115–124, Tübingen, Germany.

George Smith. 2003. A brief introduction to the TIGER treebank, version 1. Universität Stuttgart.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, page 142–147, Edmonton, Canada. Association for Computational Linguistics.

Michael Vauth, Hans Ole Hatzel, Evelyn Gius, and Chris Biemann. 2021. Automated event annotation in literary texts. In *Proceedings of the Conference on Computational Humanities Research 2021*, volume 2989 of *CEUR Workshop Proceedings*, pages 333–345, Amsterdam, the Netherlands.

Anna Mareike Weimer, Florian Barth, Tillmann Dönicke, Luisa Gödeke, Hanna Varachkina, Anke Holler, Caroline Sproleder, and Benjamin Gittel. 2022. The (in-)consistency of literary concepts. Operationalising, annotating and detecting literary comment. *Journal of Computational Literary Studies*, 1(1).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Albin Zehe, Leonard Konle, Lea Katharina Dümpelmann, Evelyn Gius, Andreas Hotho, Fotis Jannidis, Lucas Kaufmann, Markus Krug, Frank Puppe, Nils Reiter, Annekea Schreiber, and Nathalie Wiedmer. 2021a. Detecting scenes in fiction: A new segmentation task. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3167–3177, Online. Association for Computational Linguistics.

Albin Zehe, Leonard Konle, Svenja Guhr, Lea Dümpelmann, Evelyn Gius, Andreas Hotho, Fotis Jannidis, Lucas Kaufmann, Markus Krug, Frank

| Pipeline | Tokens | Sents | POS | UPOS | Lemmas | Morph | Deps |
|---|---|---|---|---|---|---|---|
| spaCy, `de_core_news_lg-3.5` | 0.9960 | 0.9220 | 0.9407 | 0.9263 | 0.9321 | 0.9084 | 0.6977 |
| spaCy, `de_dep_news_trf-3.5` | 0.9960 | **0.9378** | **0.9563** | 0.9343 | **0.9409** | **0.9436** | **0.7591** |
| Stanza 1.5 | 0.9920 | 0.8998 | 0.9390 | 0.9058 | 0.9075 | 0.9050 | 0.7489 |
| LLpro | **0.9971** | 0.8996 | 0.9406 | **0.9563** | **0.9409** | 0.9251 | 0.7444 |

Table 6: Evaluation of different NLP pipelines on the *fundamental NLP tasks* using the adapted evaluation system by Ortmann et al. (2019) against the gold annotations of the entire evaluation corpus (*wikipedia*, *novelette*, *sermon*, *TED*, *movie*). For columns *Tokens* and *Sents*, metric is F1, comparing the output from raw text input with the gold tokenization/sentencization. In all other columns, metric is accuracy, comparing the output from (gold) pre-tokenized input. Evaluation only run on the novelette text. The column UPOS refers to the universal dependencies POS tags, which are predicted alongside the fine-grained POS tagging in each pipeline.

Puppe, Nils Reiter, and Annekea Schreiber. 2021b. Shared task on scene segmentation @ KONVENS 2021. In *Proceedings of the Shared Task on Scene Segmentation*, volume 3001 of *CEUR Workshop Proceedings*, pages 1–21, Düsseldorf, Germany.

# A Appendix

## A.1 Model `fiction-gbert-large`

The foundation of our domain adaptation attempt is the RoBERTa-style (Liu et al., 2019) model `deepset/gbert-large` published by Chan et al. (2020). It is the best performing German model of its size, only competing with `deepset/glectra-large`, introduced in the same paper. Following Gururangan et al. (2020) we gathered a collection of in-domain texts and continued the models pre-training task with it. The training is performed over 10 epochs on 2.3 GB of narrative fiction with a learning rate of $1 \times 10^{-4}$ (linear decrease) and a batch size of 512. The model is available at `https://huggingface.co/lkonle/fiction-gbert-large`.

## A.2 Model `droc-character-recognizer`

We use the DROC corpus (August 11, 2022) for training. Since the DROC dataset does not define a train/val/test split on its own, we split the documents ourselves, approximating a 80/10/10 split. From the annotated DROC corpus we derive labeled sequences (in BIO format). The precise split and derivation algorithm is provided in the training code included in LLpro. Each input sequence is a concatenation of sentences, maximally filling BERT's input window. Following Flair's training procedure, training of the sequence tagger is performed over 30 epochs with an initial learning rate of $5 \times 10^{-6}$, a batch size of 4, annealing the leaning rate by factor 0.5 when micro-F1 on the evaluation set does not increase for three epochs. We take the best overall model with respect to the validation set, and report the results on the held-out test set. The model is available at `https://huggingface.co/aehrm/droc-character-recognizer`.

## A.3 Model `redewiedergabe-direct`, `-indirect`, `-reported`, `-freeindirect`

We use the identical REDEWIEDERGABE train/val/test split as used for the publication of the original taggers by Brunner et al. (2021).[25] Each binary sequence tagger (one for every STWR type) is identically trained, selected, and evaluated, following the same training procedure as for the `droc-character-recognizer`. The models are available at `https://huggingface.co/aehrm/redewiedergabe-direct`, resp. `-indirect`, `-reported`, `-freeindirect`.

## A.4 Model `stss-scene-segmenter`

We use the annotated training data provided by the Shared Task organizers.[26] A single document is held out for validation. We follow the same training procedure as the original model. For the input sequences, we set a threshold of at most 25 sentences per input sequence, and each sentence is truncated to at most 100 tokens. The training is performed over 20 epochs with a learning rate of $5 \times 10^{-6}$ (linear decrease) and batch size of 4. We take the best overall model with respect to the Shared Task evaluation score on the validation document, and report the results on the held-out test set. The model is available at `https://huggingface.co/aehrm/stss-scene-segmenter`.

[25] `https://github.com/redewiedergabe/corpus/blob/master/resources/docs/data_konvens-paper-2020.md`

[26] `http://lsx-events.informatik.uni-wuerzburg.de/stss-2021/task.html`