EEVEE: An Easy Annotation Tool for Natural Language Processing

Axel Sorensen[1] **Siyao Peng**[2,3] **Barbara Plank**[1,2,3] **Rob van der Goot**[1]
[1] Department of Computer Science, IT University of Copenhagen, Denmark
[2] Munich Center for Machine Learning (MCML), Munich, Germany
[3] MaiNLP, Center for Information and Language Processing, LMU Munich, Germany
axelsorensen.dev@gmail.com {siyaopeng,bplank}@cis.lmu.de
robv@itu.dk

Abstract

Annotation tools are the starting point for creating Natural Language Processing (NLP) datasets. There is a wide variety of tools available; setting up these tools is however a hindrance. We propose EEVEE, an annotation tool focused on simplicity, efficiency, and ease of use. It can run directly in the browser (no setup required) and uses tab-separated files (as opposed to character offsets or task-specific formats) for annotation. It allows for annotation of multiple tasks on a single dataset and supports four task-types: sequence labeling, span labeling, text classification and seq2seq.[1]

1 Introduction

Annotated datasets are of paramount importance to the Natural Language Processing (NLP) community. Their use is at the core of research, e.g. for training models, evaluating models, and analyzing trends. One of the first considerations when creating an annotated dataset is which annotation tool to choose. There is a variety of (open-source) tools readily available with extensive feature-sets. We were motivated by the following observed difficulties with existing tools when designing EEVEE:

- Most existing tools use tool-specific data formats, often with the main annotation happening on the character level. For token-based tasks, the annotator thus has to make a (tediously) precise selection of the token boundaries. Furthermore, many NLP tools expect token-level inputs (for example, for POS tagging, parsing, NER, and relation extraction). To obtain annotations on the token level, an often cumbersome conversion is necessary.
- Existing tools often require an installation which is especially problematic on constrained (organization) computers, where there might be no administrator access.
- Although many of the advanced features (like active learning) can lead to faster annotation over time, they require some setup time and more time for the annotators to get used to the tool. Time is costly in annotation; in many cases, annotators only annotate a small amount of data. Furthermore, most strategies to increase the speed of annotation (for example active learning) could lead to an additional bias signal for the annotator (Section 7).
- For many tasks, there are task-specific tools; for example for UD there is list of available annotation tools.[2] Instead, we focus on a generalizable and flexible tool. EEVEE supports a total of four task types: sequence labeling, span labeling, text classification, and sequence to sequence (Section 4).

Based on these observations, we propose EEVEE: a simple, free, and flexible annotation tool built around tab-separated files. It is written in Javascript and runs directly in the browser. It can also be saved as a desktop application and run offline. The intuitive interface allows novice users to import a dataset and set up multiple annotation tasks quickly. The graphical user interface has two main pages: the setup page (Section 2) and the annotation page (Section 3). It supports tab-separated files and raw text input (Section 4.1). We perform a case study on NER annotation with the System Usability Scale from usability engineering (Section 6). Finally, we compare EEVEE to other toolkits (Section 7).

[1] Code, README and tutorials of EEVEE are available on https://github.com/AxelSorensenDev/Eevee, demo video at https://www.youtube.com/watch?v=HsOsfckvnQo and the tool itself on https://axelsorensendev.github.io/Eevee/

[2] https://universaldependencies.org/tools.html#annotation-tools

Proceedings of The 18th Linguistic Annotation Workshop (LAW-XVIII), pages 216–221

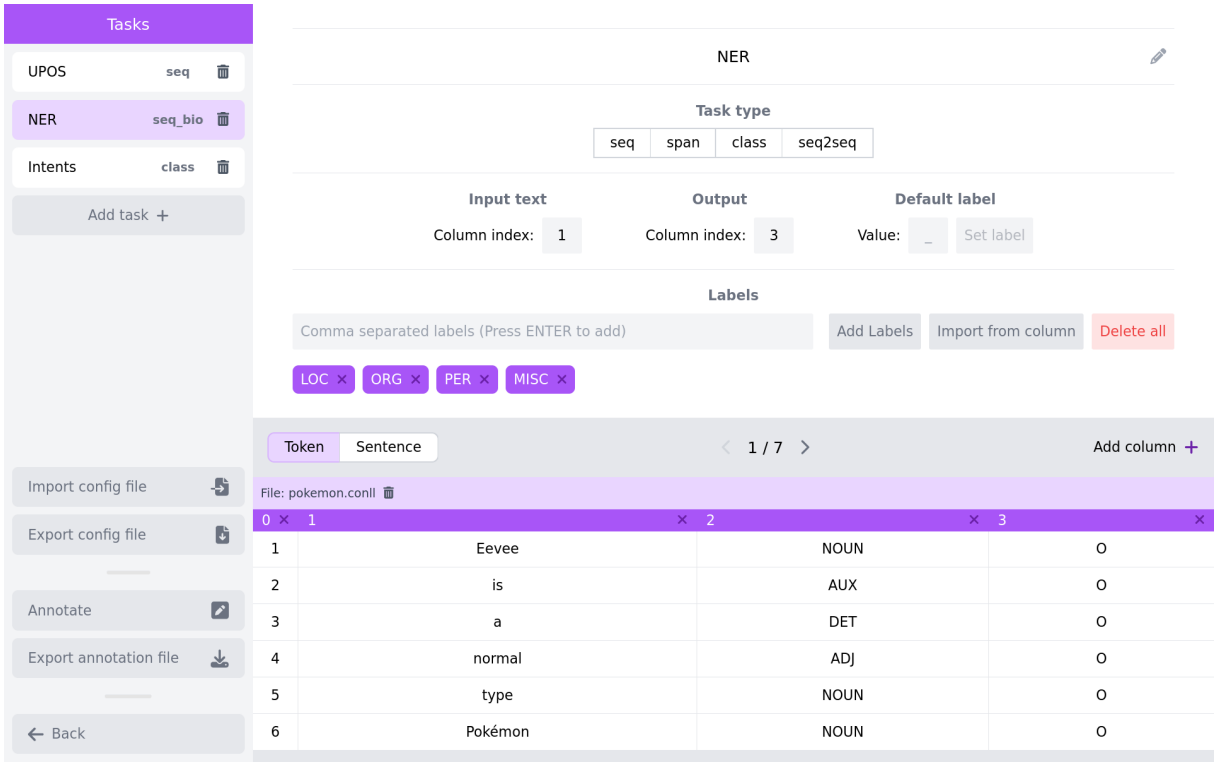

Figure 1: A screenshot of the setup page of EEVEE with multiple tasks. The user currently configures the NER task.

2 Setup page

Figure 1 illustrates the setup page where the user can define the annotation environment. Tasks can be configured in the task field (Figure 1, top right), allowing the user to specify the input column (for the input text) and output column (for the target task), as well as adding the desired labels. Labels can also be imported automatically from the annotated file (if it already contains annotations), and a default label can be set for empty annotations. For utterance-level tasks (i.e. classification), the annotation is stored in a comment above the text, in the form "# intent = inform" (see also Figure 4). To facilitate reproducibility and improve the ease of setup, the tool allows the import and export of all settings to configuration files that users can create for predefined tasks (more details in Section 4.1).

Once a dataset has been imported, the tabular data field (Figure 1, bottom right) offers a simple overview of the raw data belonging to each utterance. The user can add new columns or remove existing ones to achieve the desired result. This makes EEVEE an easy-to-use tool for extending or editing tab-separated data as well (see Section 4.1). Once the data and tasks are ready, the user simply clicks "Annotate" (Figure 1, bottom left) to continue to the "Annotation page" (see Section 3).

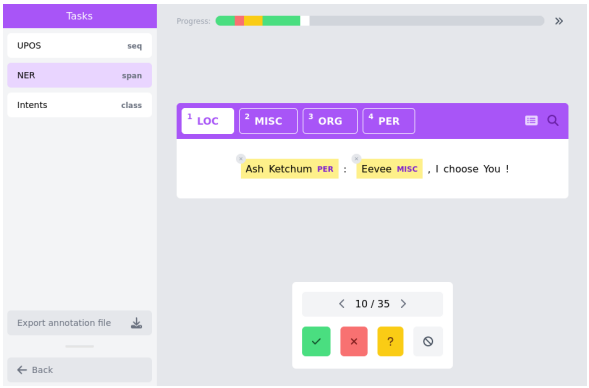

Figure 2: Annotation example with the keyboard setting.

3 Annotation page

Figure 2 illustrates an example of a NER task in the annotation interface. The user is presented with a clean, minimal annotation environment. The annotation process has been designed with efficiency in mind, enabling the user to navigate the interface also through keyboard shortcuts.

The navigation bar (Figure 2, bottom right) enables navigation between utterances and, similar to Prodigy (Montani and Honnibal, 2018), setting the status of a given task for a given utterance. The status can be set to four values: completed, wrong, unsure, and cleared (i.e. none). This overall sta-

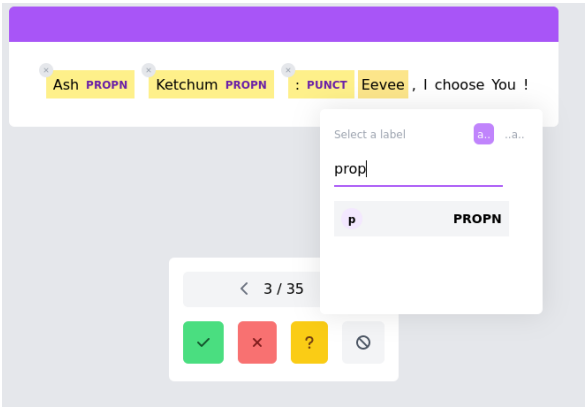

Figure 3: Searching for labels with a navigation bar.

```
# sent_id = gameboy-1
# intent = inform
1       What    PRON    O
2       ?       PUNCT   O
3       Eevee   PROPN   B-MISC
4       is      AUX     O
5       evolving        VERB    O
6       !       PUNCT   O

# sent_id = gary-1
# intent = goodbye
1       Smell   VERB    O
2       ya      PRON    O
3       later   ADV     O
4       !       PUNCT   O
```

Figure 4: Example of annotated tab-separated file with SEQ (POS in column 3), SPAN (NER in column 4), and CLASS (intent classification in the comments) tasks.

tus is reflected in the progress bar (Figure 2, top right), allowing the user to spot missing and unsure annotations easily. The progress bar is also useful when continuing annotation on a previously saved annotation file.

EEVEE provides two different annotation modes for label-based tasks: the keyboard mode and the search mode. With the keyboard mode (Figure 2), the user can use the number keys to select labels and click/select the part of the input where the label should apply (for utterance-level tasks, simply pressing the number key is sufficient). In search mode, a small pop-up appears after selecting a word or span (see Figure 3), allowing the user to find the desired label quickly. If there are more than ten labels, EEVEE defaults to search mode. Finally, the annotation file can be exported (Figure 2, bottom left). The current datetime can be appended to distinguish between different export versions.

4 Tasks

In this section, we will describe the annotation data format used by EEVEE (for import and export, importing text files is also supported), and we will discuss all the supported task types as well as the configuration files for the setups.

4.1 Data Format

There are many different data formats used in NLP, which are often task-specific. EEVEE is based on the well-established tab-separated files ubiquitously used in the NLP field. These are also sometimes called conll-like files, based on the formats used in the CoNLL shared tasks (Tjong Kim Sang and De Meulder, 2003; Buchholz and Marsi, 2006). This format (example in Figure 4) uses empty lines to separate utterances or sentences and puts one token per line. Annotations and input tokens are separated by a tab character. Comments and utterance-level information are included above the texts and are prefixed with a # character.

4.2 SEQ task-type

In sequence labeling tasks (SEQ), we annotate a single label per token, such as POS tagging or token-level language identification.

4.3 SPAN task-type

SPAN-labeling tasks are where spans are annotated as sequences of tokens (e.g. NER). Most other tools supporting this task type (e.g. Stenetorp et al., 2012; Nakayama et al., 2018) have character-level annotations, although spans normally operate on token-borders. An advantage of EEVEE is that it automatically selects the entire token if part of the token is selected, making annotation easier and faster as the annotators do not have to drag the mouse to the exact character of the token boundary. The user can simply select a label (either by clicking or pressing the corresponding number key) and then click the desired token (i.e. any character within the token) or select a span of tokens.

4.4 CLASS task-type

EEVEE also supports CLASSification tasks on the utterance level. Labels are included as a comment above the text (e.g. intents in Figure 4). The format is `# [UNIQUE NAME] = [LABEL]`, following typical meta-data format as used in conll-like formats. Usage is similar to the previous two labeling tasks, except that the user does not need to select a part of the utterance. Keyboard-only anno-

```
[{"title":"NER",
  "type":
   {"name":"seq_bio",
    "isWordLevel":true},
  "output_index":"4",
  "input_index":"1",
  "labels":["LOC","MISC","ORG","PER"],
  "id":0}]
```

Figure 5: An example of the configuration file format. The configuration file is a json file consisting of an array of tasks. Each task has a title, a type, input and output indices, and finally its corresponding labels.

tation is thus straightforward: the user can simply press a number key to select desired class labels and use the arrow keys to navigate the data.

4.5 SEQ2SEQ task-type

The SEQ2SEQ task type allows for text to text tasks (e.g. translation, question answering, summarization). This is currently the only task type without a list of provided labels; the user can directly type the target text in a text field. The annotations are utterance level and thus also saved in the comments.

4.6 Config Files

Because EEVEE runs entirely in the browser, it will not internally save the setup for the current annotation task. Therefore, it supports configuration files. These configuration files are in json format, and can thus easily be inspected by administrators, and are easy (i.e. small) to be distributed. An example of the configuration file format for named entity recognition (NER) is given in Figure 5.

5 Compatability with other services

A recent development is the Huggingface datasets library (Lhoest et al., 2021), which has indexed 62K+ datasets in two years. This library does not share the text directly but through a Python API. We provide a convenient Python script that automatically downloads data from the datasets library and converts it to the tab-separated format of EEVEE.

One of the toolkits that operates on tab-separated formats is MaChAmp (van der Goot et al., 2021), which is focused on multi-task learning. MaChAmp supports all the tasks that are included in EEVEE. For convenience, we provide a conversion script that takes EEVEE files as input and outputs a MaChAmp configuration file and the corresponding training command.

6 System Usability Study

6.1 Procedure

To assess the usability of EEVEE, we conduct a case study with two annotators on two tasks, named entity annotation (span labeling), and German dialect identification (classification). Before annotating with EEVEE, annotators spent four months labeling named entities (NE) directly on tab-separated text files in a text editor using BIO encoding and dialect identification (DID) labels as utterance-level metadata. In this case study, we ask both annotators to conduct the same NE and DID annotation tasks on a set of new documents, similar to previous ones but using the newly introduced EEVEE.

During EEVEE training, we present a 12-minute tutorial video explaining the setup and annotation pages to the annotators and provide them with tab-separated unannotated files and the json configuration files. Two annotators separately annotate the same eight documents, four from Wikipedia (*wiki*) and four from Twitter (X, *tweet*), summing up to 14.2K tokens and 16 working hours per person.[3]

6.2 Results

The System Usability Scale (SUS) was introduced as a quick and reliable tool to measure the usability of user interfaces (Brooke, 1995). It consists of a 10-item questionnaire with 5 responses ranging from 'Strongly Agree' to 'Strongly Disagree'. SUS has become an industry standard and can be validly used with small sample sizes. Therefore, we evaluate the usability of EEVEE using SUS.

The responses given by both annotators (P1 and P2) are shown in Figure 6. The ratings of the annotators result in total SUS scores of 75.0 and 87.5, both above the average of 68.0 (Brooke, 2013). The standard method for interpreting these scores is to look at which percentile they fall compared to other systems. As we are not aware of SUS being used for annotation tools, we can only compare to more general figures, where our average of 81.25 ranks at the top 10% and indicates a good (close to excellent) usability (Bangor et al., 2009). We also qualitatively survey annotators' experience and opinions after two weeks of annotation. Both annotators appraise that the tool is easy to learn and use and found it pleasant to work almost exclusively with the keyboard in a lightweight interface. Both annotators responded that they would use EEVEE for

[3]Annotators are hired student assistants and paid according to national compensation tables.

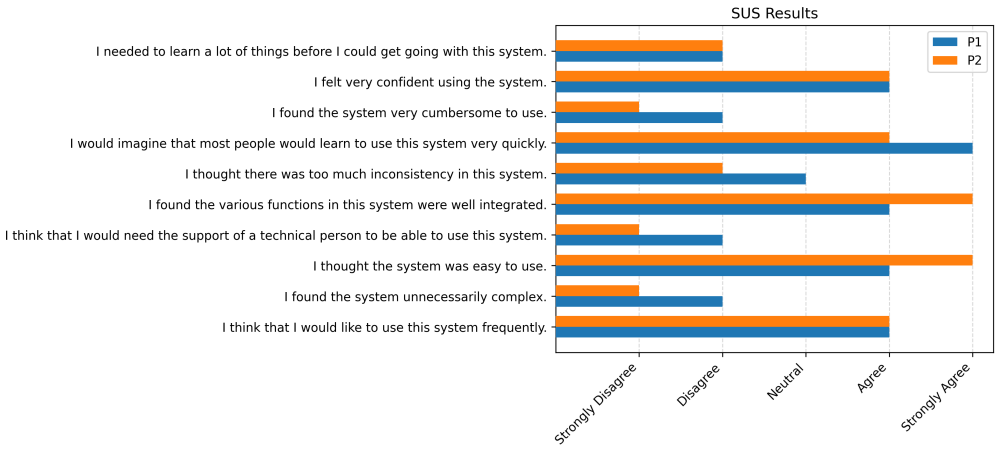

Figure 6: The results from the System Usability Scale Questionaire. The x-axis shows their agreement with a given statement, while the y-axis shows each item.

	Brat Stenetorp et al. (2012)	Potato Pei et al. (2022)	Doccano Nakayama et al. (2018)	Prodigy Montani and Honnibal (2018)	EEVEE
Open Source	✓	✓	✓	✗	✓
Character level	✓	✓	✓	✓	✗
Token level*	✗	✗	✗	✓	✓
Utterance level	✗	✓	✓	✓	✓
Data-format	standoff	json	json	json/csv	conll
Runs on	local	local	local	cloud	browser
Active learning	✗	✓	✗	✓	✗
User management	✗	✓	✓	✓	✗

Table 1: We only list the annotation export data files in this table, most tools (including EEVEE) also support importing .txt files. * Note that character level annotations are commonly used for token/span level tasks. But as noted in Section 4.3, this requires more efforts for annotation and conversion of data formats.

their next annotation jobs.

Since annotators typically spend many hours in an annotation environment, it is important that an annotation tool is built with user experience in mind. We encourage existing and future tools to consider usability studies such as SUS.

7 Comparison to other annotation toolkits

We compare EEVEE to other available toolkits in Table 1. While Eevee does not have the most functionality, it does clearly allow for a simple setup for token-level tasks. Also, EEVEE provides keyboard shortcuts for annotation speed.

Other techniques for improving annotation speed need more tuning and setup and could lead to biases. For example, active learning could lead to model bias (Berzak et al., 2016) and coloring relevant words for a task (Pei et al., 2022) could lead to biases towards these indicators. We leave the user management up to the organizer of the annotation efforts and prioritize the simplicity in tool setup. Furthermore, since EEVEE does not need installation, it does not store or send any data to the network, which is beneficial for data privacy.

8 Conclusion

We introduce EEVEE, an annotation toolkit focused on easy setup and usability. It runs directly in the browser and allows for annotation of multiple tasks. In addition, it provides convenience scripts for usage with other libraries. EEVEE's main distinguishing features, in contrast to other toolkits, are the simplicity of its setup and use, as well as annotation directly on the token level (tab-separated files). To evaluate the tool, we conducted a case study using the System Usability Scale, resulting in high usability scores. We also qualitatively surveyed the annotators' experience and noted that they would prefer to use the tool again for annotation.

Acknowledgements

We would like to thank Huangyan Shan and Marie Kolm for their invaluable feedback on EEVEE and Mike Zhang for giving feedback on earlier drafts of this paper. Huangyan Shan, Marie Kolm, Siyao Peng and Barbara Plank are supported by ERC Consolidator Grant DIALECT 101043235.

Limitations

We acknowledge that EEVEE assumes gold token detection (and annotates on the token level for SEQ and SPAN). For languages/datasets where tokenization is challenging, this would require a first pass of tokenization annotation before importing the data into EEVEE. Furthermore, the input is constrained to text in Unicode font, which is unavailable for some languages.

References

Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123.

Yevgeni Berzak, Yan Huang, Andrei Barbu, Anna Korhonen, and Boris Katz. 2016. Anchoring and agreement in syntactic annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2215–2224, Austin, Texas. Association for Computational Linguistics.

John Brooke. 1995. SUS: A quick and dirty usability scale. *Usability Evaluation in Industry*, 189.

John Brooke. 2013. SUS: a retrospective. *Journal of Usability Studies*, 8:29–40.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City. Association for Computational Linguistics.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ines Montani and Matthew Honnibal. 2018. Prodigy: A new annotation tool for radically efficient machine teaching.

Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. 2018. doccano: Text annotation tool for human. Software available from https://github.com/doccano/doccano.

Jiaxin Pei, Aparna Ananthasubramaniam, Xingyao Wang, Naitian Zhou, Apostolos Dedeloudis, Jackson Sargent, and David Jurgens. 2022. POTATO: The portable text annotation tool. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 327–337, Abu Dhabi, UAE. Association for Computational Linguistics.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. 2021. Massive choice, ample tasks (MaChAmp): A toolkit for multi-task learning in NLP. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 176–197, Online. Association for Computational Linguistics.