# MATILDA: Multi-AnnoTator multi-language Interactive Light-weight Dialogue Annotator

**Davide Cucurnia**[♢] **Nikolai Rozanov**[♠] **Irene Sucameli**[♢]
**Augusto Ciuffoletti**[♢] **Maria Simi**[♢]

[♢]Department of Computer Science, University of Pisa, Pisa, Italy
[♠]Wluper Ltd., London, United Kingdom
[♢]{d.cucurnia2@studenti.,irene@phd.,
augusto.ciuffoletti@, maria.simi@}unipi.it
[♠]{nikolai}@wluper.com

## Abstract

Dialogue Systems are becoming ubiquitous in various forms and shapes, from virtual assistants (like Siri, Alexa and various chat-bots) to customer support systems embedded within websites. Recent publications and advancements with natural language modelling have opened up NLP (and its more advanced applications like conversational AI) to a wider audience. Unfortunately, the lack of labelled data within this field remains a significant barrier and so we have developed MATILDA (the first multi-annotator, multi-language dialogue annotation tool) as an initial contribution to help the community overcome this barrier. MATILDA is a tool for creating high-quality corpora via a user-friendly interface so as to facilitate the annotation of dialogues, resolve inter-annotator disagreement and manage multiple users at scale. We have evaluated the tool on ease of use, annotation speed and inter-annotation resolution for both experts and novices and can confidently conclude that MATILDA offers a novel, streamlined, end-to-end solution to dialogue annotation and is intuitive enough to use, even for a non-technical audience. The tool is completely open-sourced at https://github.com/wluper/matilda and is easily adaptable to any language. We are also providing a complementary tutorial video[1].

## 1 Introduction

As a community, we have observed great advances in the last decade that include word-embeddings (Mikolov et al., 2013), seq-to-seq models for a variety of tasks (Sutskever et al., 2014) and pre-trained, transformer-based language models (Devlin et al., 2019). Relying on these seminal works, a plethora of downstream tasks (e.g. NMT, Q&A, dialogues, summarisation, etc.) have seen notable improvements and some have even been "solved". Many of the advancements made in computational modelling and power owe a lot of their success to the careful curation and annotation of huge datasets, which are thus equally pivotal to recent advancements and progress in general. In particular, datasets such as (Budzianowski et al., 2018) and (Byrne et al., 2019) have allowed data-hungry neural-models to advance the field of task-oriented dialogues.

In the field of annotation tools and data generation, recent advances such as (Collins et al., 2019) show similar promise by open-sourcing technology and developing it with modern usability-related principles in mind. Following in the spirit of such similar research, we present MATILDA (a full dialogue annotation tool specifically focused on the inclusivity for all languages and facilitating multiple annotators). We evaluate it on a variety of usability aspects, both with experienced and untrained users, and conclude that both our dialogue annotation and creation tools are easy-to-use. Furthermore, MATILDA offers more features than any comparable tool in the research community; comfortably supporting multiple annotators as well as multiple languages during the annotation process. Therefore, we have open-sourced it and provided precompiled docker images for easy setup.

MATILDA's main contributions are: 1) a native annotation tool that is quick-to-adapt[2] for multi-language support; 2) a user-friendly interface to simply and intuitively manage multiple users as well as easily distribute datasets to crowd-workers for annotation; 3) task-oriented multi-speaker annotation capabilities (in the style of MultiWoz and Taskmaster); 4) inter-annotator resolution; and 5) integrated recommendations to assist annotators.

[1] https://vimeo.com/500125248

[2] As an example the full adaptation of the annotation tool from English to German took roughly 30 minutes.



*Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 32–39
April 19 - 23, 2021. 

## 2 Related Work

Table 1 compares MATILDA with other recent annotation tools.

TWIST (Pluss, 2012) is a dialogue annotation tool which consists of two stages: turn segmentation and content feature annotation. Turn segmentation allows users to create new turn segments from raw text. After this, users can annotate sections of text in a segment by highlighting them and selecting from a predefined feature list. However, this tool doesn't allow users to specify custom annotations or labels and doesn't support classification or slot-value annotation. This is not compatible with modern dialogue datasets which require such annotations (Budzianowski et al., 2018). INCEpTION (Klie et al., 2018) is a semantic annotation platform for interactive tasks that require semantic resources like entity linking. It provides machine learning models to suggest annotations and allows users to collect and model knowledge directly in the tool. GATE (Cunningham, 2002) is an open source tool that provides predefined solutions for many text processing tasks. It is powerful because it allows annotators to enhance the provided annotation tools with their own Java code, making it easily extensible and provides a great number of predefined features. However, GATE is a large and complicated tool with a significant setup cost - its instruction manual alone is over 600 pages long[3]. Despite their large feature sets, INCEpTION and GATE are not designed for annotating dialogue and cannot display data as turns, an important feature for dialogue datasets. BRAT (Stenetorp et al., 2012) and Doccano[4] are web-based annotation tools for tasks such as text classification and sequence labelling. They have intuitive and user-friendly interfaces which aim to make the creation of certain types of dataset such as classification or sequence labelling datasets as fast as possible. BRAT also supports annotation suggestions by integrating ML models. However, like INCEpTION[5] and GATE[6], they are not designed for annotating dialogues and do not support the generation of formatted conversational data from a raw text file such as might be outputted by a transcription service. LIDA (Collins et al., 2019) provides an easy-to-setup annotation tool for modern task-oriented dialogues and also supports the integration of recommendations. However, LIDA is not accessible for multiple users and is only intended for the English language. MATILDA addresses these shortcomings and adds features such as: annotation styles compatible with modern dialogue datasets, inter-annotation resolution, customisable recommendations and user administration. DialogueView's (Heeman et al., 2002) main use-cases are focused on segmenting recorded conversations, annotating audio files and discourse segmentation. Granular labelling of the dialogue, recommenders, inter-annotator agreement, and slot-value labelling are not possible.

[3]https://gate.ac.uk/sale/tao/tao.pdf

[4]https://github.com/chakki-works/doccano

[5]A plugin allows calculation of scores not resolution: https://dkpro.github.io/dkpro-statistics/dkpro-agreement-poster.pdf

[6]Again inter-annotator score calculation capabilities are available as separate plug-in https://gate.ac.uk/releases/gate-5.1-beta1-build3397-ALL/doc/tao/splitch10.html - however support for resolutions is not apparent

## 3 System Overview

We introduce an annotator service that extends previous successful experiences, like LIDA, by introducing features that address large-scale, task-oriented dialogue annotation projects. In particular, we allow for distributed multi-annotators, multi-language support, interannotator resolution and custom recommenders to assist the annotation process. Furthermore, our modern and modularised implementation simplifies extension to additional languages, use-cases and annotation styles. A typical use-case follows this workflow:

**Creation of a Dataset** We envision two main ways to create a corpus: either interactively or by uploading existing data. We adopt data representations that allow backward compatibility with other tools based on text files with a simple syntax, and a JSON format that is easy to operate.

**User Administration** Once a corpus consisting of several collections is created, administrators can then proceed to assign those collections to one or more different annotators. The assigned partition will then be shown to the designated annotators in their "Collection" view, ready to be annotated. According to the typical use case, we need two roles for the users, which we call *Annotators* and *Administrators*. We want our system to include user management with a simple interface for creation, editing and removal.

**Annotation and Supervision** Each annotator has access only to the subsets of dialogues assigned to them to add/modify annotations and monitor

| Annotation Tool | Dialogue-specific Annotation | Multi-language Support | Crowd Multi-annotator Support | Recommenders | Inter-Annotator Disagreement Resolution | Language |
|---|---|---|---|---|---|---|
| MATILDA | YES | YES | YES | YES | YES | PYTHON |
| LIDA (Collins et al., 2019) | YES | NO | NO | YES | YES | PYTHON |
| INCEpTion (Klie et al., 2018) | NO | NO | NO | YES | YES/NO[3] | JAVA |
| GATE (Cunningham, 2002) | NO | NO | NO | NO | YES/NO [4] | JAVA |
| TWIST (Pluss, 2012) | YES | NO | NO | NO | NO | - |
| BRAT (Stenetorp et al., 2012) | NO | NO | NO | YES | NO | PYTHON |
| DOCCANO[3] | NO | NO | NO | NO | NO | PYTHON |
| DialogueView (Heeman et al., 2002) | YES | NO | NO | NO | NO | TcK/TK |

Table 1: Annotator Tool Comparison Table
Dialogue-specific Annotation: Support to annotate datasets such as MultiWoz or Taskmaster. Multi-language Support: The ability to localise the annotation tool for different languages. Crowd Multi-annotator Support: The possibility to manage users and easily deploy to many annotators in different locations. Recommenders: ML models to suggest annotations. Inter-Annotator Disagreement Resolution: whether the system has an interface to resolve disagreements between different annotators. Language: what programming language the system uses

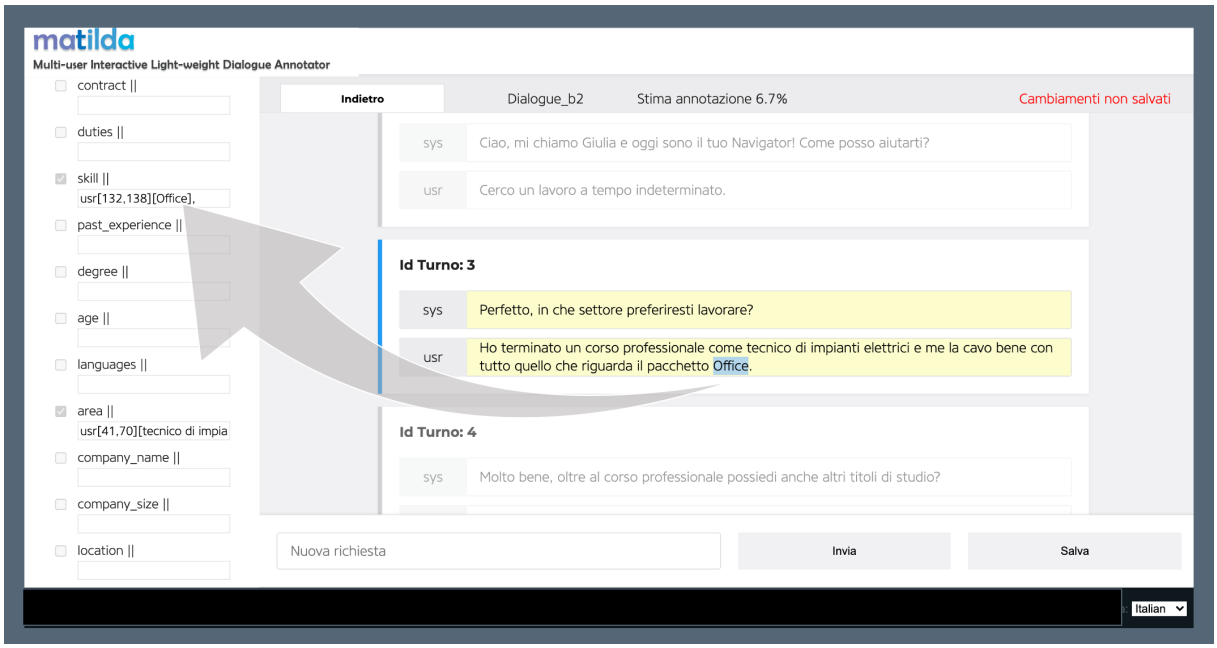


Figure 1: Dialogue annotation interface: filling slots by selection of text fragments.

work progress. Figure 1 shows a screenshot of the annotation interface and highlights the slot-filling functionality. Administrators inspect annotators' work and resolve conflicts in the *interannotation* interface. When annotators provide diverging annotations, a designated supervisor provides a *gold standard* either opting for one of them or introducing an alternative one. Besides, the system computes interannotator agreement metrics, such as Cohen's Kappa. Gold standard annotations provided by supervisors are recorded separately and do not overwrite the original ones.

The Interannotator is designed to confront two or more annotated dialogue collections and resolve annotation conflicts between them. MATILDA automatically retrieves all annotated versions of one corpus partition present in the database; administrators are also allowed to upload a file to add to the confrontation. This can be seen in Figure 2

### 3.1 System architecture

MATILDA is designed as a Web Service: a browser hosts the user interface while the server supports data and control. Our use case envisions all components running on user premises, but it is straightforward to distribute them on distinct hosts.

On the server side, MATILDA is a bundle of two components: a web server and a database server.

Each of them is encapsulated in a Docker, so that complex configurations are carried out by the designer and invisible to the non-technical end-user. In fact, MATILDA operation depends only on Docker support, which is available for major operating systems. In order to have MATILDA operational, the end-user installs the Docker support and launches a Docker script that downloads and deploys on the user's PC the server-side Dockers. MATILDA is then reachable from the browser

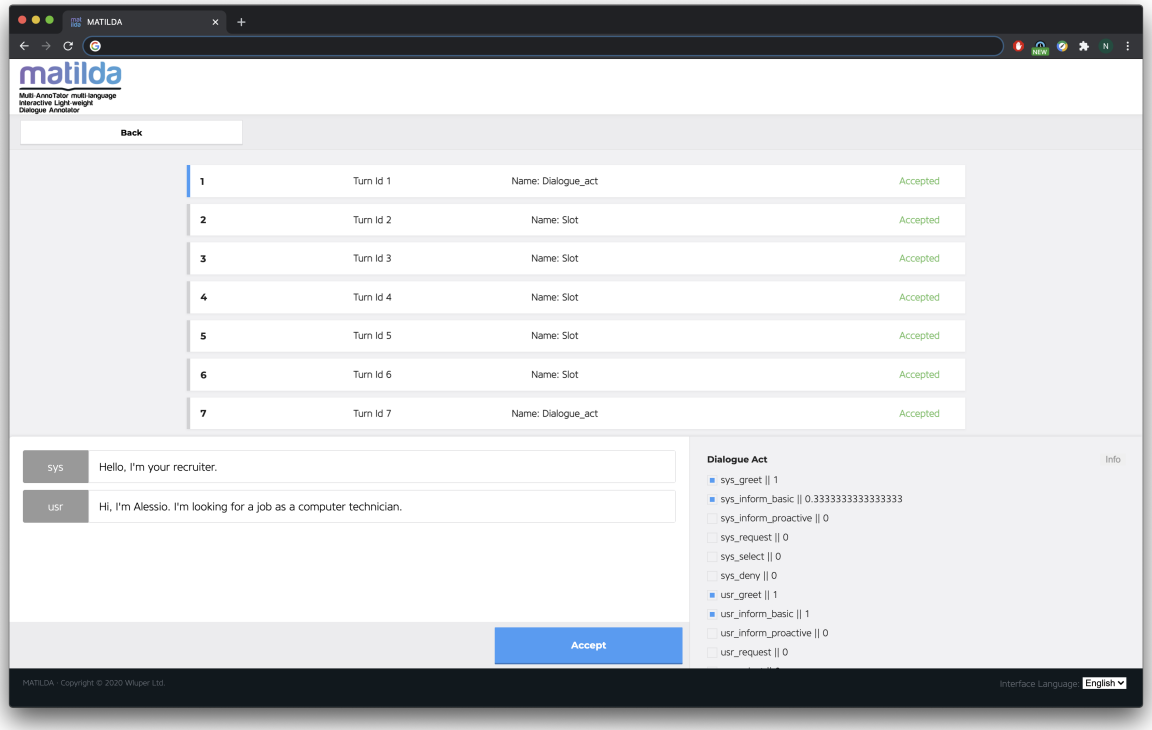


Figure 2: Inter-annotation Resolution Interface.

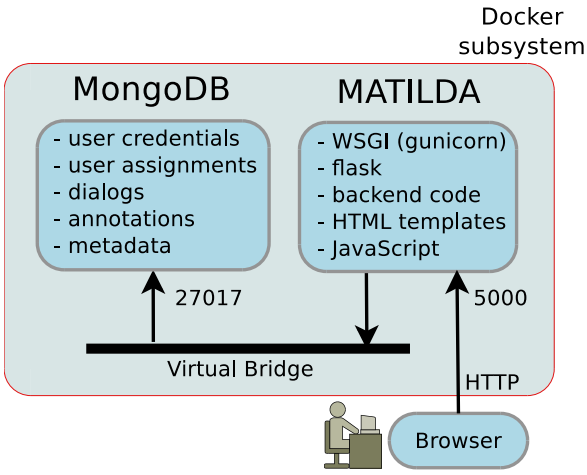


Figure 3: architecture

at the URL `http://localhost/index.html`. The tech-inclined user has the option to configure some features, like the use of an external database or the access through a network interface. The installation script and the operation manual are distributed on GitHub `https://github.com/wluper/matilda`, while the Dockers are available from `https://hub.docker.com`.

As seen in Figure 3, the MATILDA engine is written in Python using the *Flask* framework, while the client-side JavaScript uses the *Vue* framework. The MongoDB database provides NoSQL access to the dialogs, the annotations and their metadata. This technology meets the required flexibility, allowing heterogeneous types of documents and an agile structure. The native support of JSON documents matches with the format used for the internal representation of the dialogs. Finally, the availability of both an open-source server and a public service is useful when implementing either a service on-premises, according to the reference use-case, or, in a more advanced use-case, to implement a cloud database for sharing dialogs.

The most stringent requirement on host hardware is that the processor must belong to the 64-bit family; this is inherited from Docker. To analyse the footprint of MATILDA components, we installed it on a system based on the Intel Celeron J3355, a 2-core microprocessor dated 2016, created for entry level desktop systems and with a 2GB RAM. During a significant processing peak, induced with an upload, the footprint did not exceed a few percent of hardware capacity.

The developer can find the engine source code in the GitHub repository mentioned above; this allows them to customize or to add new features to MATILDA and to produce a new Docker. Locale-dependent information is recorded in an indepen-

dent JSON document, and so introducing a different localization of the interface is non-intrusive (?).

## 4 Evaluation

MATILDA was evaluated on two experiments: the first evaluated MATILDA's admin-related capabilities while and the second evaluated its annotation performance. Both experiments were conducted across three different languages (English, Italian and German) to assess MATILDA's cross-language adaptability.

### 4.1 Quantitative Evaluation

#### 4.1.1 Administration and Supervision

The administration experiment involved a total of six participants, each representing different supervisory roles: i) an expert supervisor (**ES**) who is familiar with MATILDA or has relevant background knowledge in NLP and dialogue annotation and ii) an untrained supervisor (**US**) who has never used MATILDA before and has little to no experience with dialogue annotation in general. The initial *admin task* consisted of adding two new users (A1 and A2) into MATILDA and assigning them as annotators, then creating a new dialogue collection and defining its features (e.g. collection's title, its description, etc.) and assigning the new collection to all the annotators. The second *inter-annotator task* consisted of resolving inter-annotator conflicts which may occur at the end of the annotation work, which involved the supervisor comparing conflicts on MATILDA for each annotator disagreement and selecting one, thus creating a final, gold dataset.

During the two phases of the experiment, we record the time needed for ES and US to complete the tasks. Table 2 describes and compares the time taken on the admin task for the two supervisors across the three languages considered. It also shows the time taken to resolve inter-annotator disagreements as well as the total number of disagreements resolved.

The quantitative evaluations show that both trained and untrained supervisors were able to successfully complete the predefined tasks, with the untrained supervisors performing only marginally worse, despite having never used an annotation tool before. The untrained supervisors were provided with a 15 minute guided training prior to the inter-annotation task as they were unfamiliar with the task (having no prior NLP knowledge or experience).

| *Time(min:sec) per admin task* | | | |
|---|---|---|---|
| | **English** | **Italian** | **German** |
| **ES** | 03:45 | 03:05 | 02:20 |
| **US** | 02:52 | 02:55 | 03:30 |
| *Time(min:sec) per inter-annotator task* | | | |
| **ES** | 22:05 | 09:31 | 17:30 |
| **US** | 26:30* | 25:02 | 15:13* |
| **Conflicts** | 38 | 40 | 25 |
| **Total Labels** | 130 | 130 | 130 |

Table 2: Comparison of the time taken by different supervisors to carry out admin and inter-annotators resolution tasks. *Needed additional training before being able to perform the task

The evaluation revealed a strong dependency on the execution of admin tasks with the supervisor's familiarity with MATILDA and annotation systems in general. However, the results also indicate that users who are unfamiliar with annotation tools are still able to easily use MATILDA and complete administration and inter-annotation tasks.

#### 4.1.2 Annotation

The second evaluation focuses on quantitatively analysing the tool's annotation interface. An expert annotator (**EA**) and an untrained annotator (**UA**) were both asked to annotate five dialogues and the time taken to complete the task was recorded (the results are shown in Table 3). Each dialogue, across all languages tested, had an average of eight turns (wherein a turn consisted of one user utterance and system response) and twenty-four possible class labels per turn (10 dialogue acts and 14 slots). This complexity is comparable with those of public dialogue datasets, like Multiwoz or Taskmaster-1 (Budzianowski et al., 2018; Byrne et al., 2019).

| *Time(min:sec) per annotation task* | | | |
|---|---|---|---|
| | **English** | **Italian** | **German** |
| **EA** | 34:27 | 16:35 | 27:55 |
| **UA** | 37:30 | 49:48 | 45:00 |

Table 3: Time taken to annotate a set of 5 dialogues by different native-speaker annotators

The results of this experiment show that even untrained annotators were able to use MATILDA to successfully complete the annotation task. In fact, a substantial increase in the users' annotation

speed can be observed within just a few annotations, demonstrating a fast learning curve for MATILDA.

For expert annotators, the average annotation time was 26:17 minutes for five dialogues (giving an average of approximately 5:16 minutes per dialogue). For untrained annotators, this increases to approximately 8:50 minutes per dialogue. Therefore, annotating a data-set of 10,000 dialogues (with two annotations per dialogue) can be calculated as requiring 1,756 hours or 100x 8-hour working days for two expert annotators to complete on MATILDA. However, this time can be massively reduced using untrained crowd-workers, wherein approximately 52 untrained workers could complete the annotation of such a dataset within a week. Thus highlighting the importance of such tools and software as MATILDA, that can manage, collate and resolve annotation conflicts across the crowd-workers.

### 4.2 Qualitative Evaluation & User Feedback

#### 4.2.1 Questionnaire

In addition to the quantitative evaluations, a qualitative analysis was conducted in the form of a questionnaire about MATILDA's usability, provided to each annotator and supervisor as an an anonymous feedback form. Each supervisor was asked to evaluate the following features with a *Low-Medium-High* score:

- Q1: ease of completing the admin task;
- Q2: ease of resolving inter-annotator conflicts;
- Q3: quality of the feedback provided by the tool.
- Q4: overall usability of MATILDA admin interface.

| *Supervisors evaluation* | | | |
|---|---|---|---|
| | **Low** | **Medium** | **High** |
| **Q1** | 0.0% | 16.7% | 83.3% |
| **Q2** | 16.7% | 50.0% | 33.3% |
| **Q3** | 33.3% | 50.0% | 16.7% |
| **Q4** | 0.0% | 33.3% | 66.7% |

Table 4: Evaluation of MATILDA usability

Similarly, we ask annotators to evaluate:

- Q1: ease of annotation;
- Q2: ease of understanding how to work on a dialogue collection and how to sent it to supervisors at the end of the annotation;
- Q3: quality of the feedback provided by the tool.
- Q4: overall usability of MATILDA annotator interface.

| *Annotators evaluation* | | | |
|---|---|---|---|
| **Q1** | 0.0% | 66.7% | 33.3% |
| **Q2** | 0.0% | 33.3% | 66.7% |
| **Q3** | 66.6% | 16.7% | 16.7% |
| **Q4** | 0.0% | 33.3% | 66.7% |

Table 5: Evaluation of MATILDA usability

Tables 4 and 5 show the percentages of responses to each question for supervisors and annotators respectively. Question 4 (Q4) about overall usability shows 66.7% Good usability, 33.3% Medium usability and nobody answered with Low usability (including the untrained annotators) which confirm the quantitative results regarding MATILDA's low-friction usability. Questions about the individual aspects of the tasks (Q1 and Q2) also confirm the overall usability of the tool, receiving mostly Good or Medium scores. The main point for improvement, according to the responses, was the level of feedback the tool provides to the user (i.e. prompts that show whether a user action was successful at a task, like the successful creation of a user, etc)

#### 4.2.2 Feedback

We have also provided the study participants the venue to express their feedback in an unstructured way, by prompting them, "Please provide feedback in a couple of sentences on the usability of the annotation and supervision aspects of the app and the improvements you would suggest".

The feedback can be summarised in three categories:

1. Feedback and Information Prompts by the tool
2. Improving slot-filling for the annotation tool
3. Improving the layout of the inter-annotator resolution

The first feedback was also apparent from the feedback forms provided in the previous section. We have accepted this feedback to improve our

tool and the to-be-published version is planned to include these improvements.

The second feedback point was very important and the future version of the tool will work on improving the slot-filling annotation format.

The final feedback was more of an aesthetic feedback about the location and visibility of certain aspects of the interannotator resolution screen.

## 5 Conclusion and future work

We have presented MATILDA the first, to the best of our knowledge, multi-annotator, multi-language dialogue annotation tool that allows the user to annotate, distribute annotation work among crowdworkers or colleagues and to resolve annotation conflicts. We evaluate the tool based on the ease and rapidity of use and show that even untrained novices can quickly learn to use it.

Thanks to the open-source nature of the original LIDA project, we hope the community will pick-up on this work both in terms of using it to create strongly needed corpora for different languages as well as extending it to allow even more use-cases and more advanced annotation styles.

To this end we have conducted qualitative feedback sessions with study participants and provided a potential avenue of concrete improvements. We hope that this work will be a meaningful stepping stone for our community to create more useful resources in many languages.

## 6 Acknowledgements

In this work we would like to acknowledge the great input from EACL Reviewers that helped us push the paper to a new level.

We particularly would like to thank the thoughtful input of our colleagues in the University of Pisa, especially Clara Casandra and Carla Congiu.

We would also like to thank members of the Wluper team that acted as Testers, Annotators and Paper Reviewers. In particular, special thanks go to Mohammed Terry-Jack, Lamia El Afani, Andrew Burnie, Ed Collins and Maurice von Sturm. Furthermore, additional thanks goes to the authors and developers of the previous version of this annotation tool - LIDA - Ed Collins and Bingbing Zhang.

Furthermore, the work of Nikolai Rozanov was done under the Research Lab of Wluper Ltd. (UK/ 10195181) and part of the contribution of this lab was supported by the *Innovate UK Smart Grants: October 2019*.

## References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Daniel Duckworth, Semih Yavuz, Ben Goodrich, Amit Dubey, Kyu-Young Kim, and Andy Cedilnik. 2019. Taskmaster-1: Toward a realistic and diverse dialog dataset.

Edward Collins, Nikolai Rozanov, and Bingbing Zhang. 2019. LIDA: lightweight interactive dialogue annotator. 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 2019, Hong Kong, China, November 3-7, 2019 - System Demonstrations*, pages 121–126. Association for Computational Linguistics.

Hamish Cunningham. 2002. Gate, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter A. Heeman, Fan Yang, and Susan E. Strayer. 2002. DialogueView - an annotation tool for dialogue. In *Proceedings of the Third SIGdial Workshop on Discourse and Dialogue*, pages 50–59, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The inception platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Brian Pluss. 2012. Twist dialogue annotation tool. http://mcs.open.ac.uk/nlg/non-cooperation/resources/user-guide.pdf.

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. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.