AdapterHub Playground: Simple and Flexible Few-Shot Learning with Adapters

Tilman Beck[1], **Bela Bohlender**[1], **Christina Viehmann**[2], **Vincent Hane**[1], **Yanik Adamson**[1], **Jaber Khuri**[1], **Jonas Brossmann**[1], **Jonas Pfeiffer**[1], **Iryna Gurevych**[1]
[1]Ubiquitous Knowledge Processing Lab (UKP Lab)
Department of Computer Science
Technical University of Darmstadt
[2]Institut für Publizistik
Johannes Gutenberg-University Mainz

Abstract

The open-access dissemination of pretrained language models through online repositories has led to a democratization of state-of-the-art natural language processing (NLP) research. This also allows people outside of NLP to use such models and adapt them to specific use-cases. However, a certain amount of technical proficiency is still required which is an entry barrier for users who want to apply these models to a certain task but lack the necessary knowledge or resources. In this work, we aim to overcome this gap by providing a tool which allows researchers to leverage pretrained models without writing a single line of code. Built upon the parameter-efficient adapter modules for transfer learning, our AdapterHub Playground provides an intuitive interface, allowing the usage of adapters for prediction, training and analysis of textual data for a variety of NLP tasks. We present the tool's architecture and demonstrate its advantages with prototypical use-cases, where we show that predictive performance can easily be increased in a few-shot learning scenario. Finally, we evaluate its usability in a user study. We provide the code and a live interface[1].

1 Introduction

The success of transformer-based pretrained language models (Devlin et al., 2019; Liu et al., 2019) was quickly followed by their dissemination, gaining popularity through open-access Python libraries like Huggingface (Wolf et al., 2020), AdapterHub (Pfeiffer et al., 2020a) or SBERT (Reimers and Gurevych, 2019). Researchers and practitioners with a background in computer science are able to download models and fine tune them to their needs. They can then upload their fine-tuned model and contribute to an open-access community of state-of-the-art (SotA) language models for various tasks and in different languages.

This has significantly contributed to the democratization of access to the latest NLP research as the individual implementation process has been simplified through the provision of easy-to-use and actively managed code packages. However, one still needs a certain level of technical proficiency to access these repositories, train models, and predict on new data. This is a limiting factor for researchers in disciplines who could benefit from applying SotA NLP models in their field, but lack the technical ability. Furthermore, there is growing interest for text classification models in interdisciplinary research (van Atteveldt et al., 2021; Boumans and Trilling, 2016), although often the methods are not SotA in NLP.

In this work, we hope to bridge this gap by providing an application which makes the power of pretrained language models available without writing a single line of code. Inspired by the recent progress on parameter-efficient transfer learning (Rebuffi et al., 2017; Houlsby et al., 2019), our application is based on adapters which introduce small and learnable task-specific layers into a pretrained language model. During training, only the newly introduced weights are updated, while the pre-trained parameters are frozen. Adapters have been successfully applied in machine translation Bapna and Firat (2019); Philip et al. (2020), cross-lingual transfer (Pfeiffer et al., 2020b, 2021b; Üstün et al., 2020; Vidoni et al., 2020), community QA (Rücklé et al., 2020), task composition for transfer learning (Stickland and Murray, 2019; Pfeiffer et al., 2021a; Lauscher et al., 2020; Wang et al., 2021) and text generation (Ribeiro et al., 2021). Adapters are additionally computationally more efficient (Rücklé et al., 2021a) and more robust to train (He et al., 2021; Han et al., 2021). In our work, we build our application on top of the AdapterHub (Pfeiffer et al., 2020a) library which stores task-specific adapters with a large variety of architectures and offers upload functionalities

[1] https://adapter-hub.github.io/playground

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
System Demonstrations, pages 61 - 75

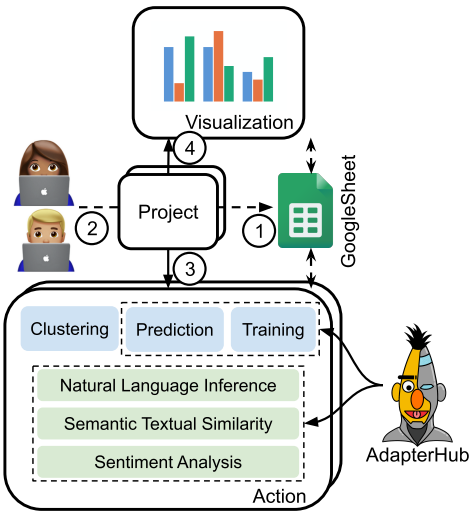

Figure 1: Diagram of the AdapterHub Playground workflow. ① Users upload their text data to GoogleSheets and ② link it to a new project. ③ In each project, users can create multiple actions by selecting a specific action type `Training`, `Prediction`, `Clustering`. For the `Training` and `Prediction` action types, the user needs to define the desired downstream task (e.g. `Sentiment Analysis`). Information about available pretrained adapters for the specified task are dynamically retrieved from AdapterHub. ④ After generating predictions, the user can visualize the results within the project.

for community-developed adapter weights. We leverage this library to allow no-code access to pretrained adapters for a many text classification tasks using dynamic code generation. Finally, our application enables the analysis of multi-dimensional annotations to further investigate model performance.

Our application supports both NLP and interdisciplinary researchers who want to evaluate the transferability of existing pretrained adapters to their specific domains and use cases. We intend this application for both zero-shot as well as few-shot scenarios where a user annotates a small number of data points and monitors model improvements. This is especially interesting for *intermediate* task training (Phang et al., 2018) where models trained on a compatible task are utilized and fine-tuned on the target task.

Some efforts are being made to abstract away engineering requirements to use SotA NLP (Akbik et al., 2019), but their usage still requires certain technical skills. Existing no-code (or AutoML) applications like Akkio[2], Lobe[3], or Teachable Machines[4] allow users to upload data, annotate it using self-defined labels, and train a model for prediction. Most approaches focus on vision tasks and follow commercial goals. To the best of our knowledge, we are the first to provide a non-commercial, no-code application for text classification. Our application is transparent (i.e. details about usable pretrained adapters are traceable), and extendable via the community-supported AdapterHub library. Finally, we enable execution on third party computational servers for users without access to the required GPU hardware for efficient training and prediction (Rücklé et al., 2021b), while also providing the necessary scripts to setup a self-hosted computing instance, mitigating technical dependencies.

Our contributions are: **1)** The AdapterHub Playground application which enables no-code inference and training by utilizing pretrained adapters; **2)** Prototypical showcase scenarios from social sciences using our application for few-shot learning; **3)** An elaborate user study that analyzes the usability of our proposed application.

2 AdapterHub Playground

The AdapterHub Playground is a lightweight web application offering no-code usage of pretrained adapters from the AdapterHub library. A user interface accompanied by dynamic code generation allows the utilization of adapters for inference and training of text classification tasks on novel data. Below, we describe the application workflow[5], provide details on the specific functionalities and highlight the technical architecture.

2.1 Workflow

The workflow of the AdapterHub Playground is depicted in Figure 1. First, a user creates a GoogleSheet[6] and uploads the input data for the desired classification task. If applicable, additional metadata, for example, annotations or timestamps, can be added. Next, a new project can be created and linked to the data via the GoogleSheet sharing functionality. Within a project, the user can define an *action*, resembling a computational unit (e.g. training an adapter). Upon submission of a new action, the input text data is downloaded and the specified computation is performed. The user is

[2] `https://www.akkio.com/`
[3] `https://lobe.ai/`
[4] `https://tinyurl.com/teachablemachines`
[5] We provide information about user requirements in the Appendix A.
[6] `https://docs.google.com/spreadsheets/`

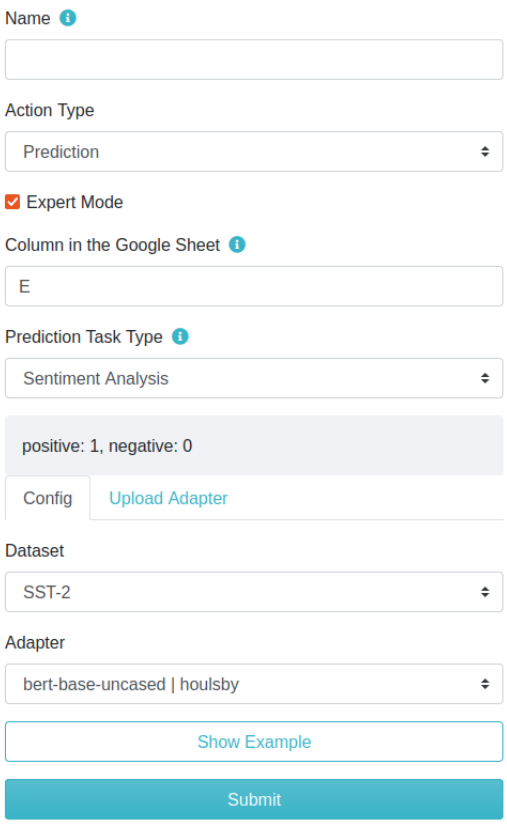

Figure 2: Screenshot of the action creation dialogue. A user has to provide a name for the action, the action type (here `Prediction`), the column in GoogleSheet where results are written to and the downstream task (here `Sentiment Analysis`). In the expert mode, the user has additional options for the pretrained adapter, i.e. the dataset which was used for pretraining and the specific architecture. The available options are dynamically retrieved from AdapterHub. Alternatively a self-provided adapter can be uploaded.

informed visually about the status of the execution in the application. After finishing the computation, the results are written directly into the GoogleSheet by the system and evaluation details are provided in the action interface. By default the system supports accuracy and macro-F1 evaluation metrics. To aid users in estimating model performance we additionally provide the results for random and majority prediction.

A user can create multiple projects, and within each project, multiple actions can be triggered using the same input data.[7] Finally, within a project a user can explore the predictions on the data using different visualization methods.

2.2 Actions

Our application focuses on three main *actions*, namely `Prediction`, `Training` and `Clustering`. For each action, the respective code is dynamically generated by merging static code snippets with parameters defined by the user (e.g. the specific adapter architecture). In the following we describe the procedure of each action.

Prediction. Pretrained task-specific adapters can be utilized for predictions on proprietary data. The user creates a new action in the project detail page and selects as action type `Prediction`, defines the column of the GoogleSheet in which the predictions should be written, and selects the respective downstream task which is dynamically retrieved from the AdapterHub.[8] Execution triggers the backend program to load the specified adapter and data, and produce task-specific labels for the data. A screenshot of the action creation dialogue is provided in Figure 2.

Training. To allow for continual training of adapters on labeled data, the user creates a new action of type `Training`. When executed, the backend process loads the specified adapter, downloads both data and target labels, and starts the training procedure. Once training is completed, the user can download the fine-tuned adapters as a zipped file. This makes fine-tuned adapter weights available for another `Prediction` action.

The choice of hyperparameters can have substantial influence on task performance but evaluating these effects is out of scope for this work. Defaults are set based on the literature (Pfeiffer et al., 2021a), however, if necessary, the user can modify training hyperparameters through various dropdown fields. This allows to compare multiple adapters trained with different hyperparameters.

Clustering. Discovering recurrent patterns in text data is a common procedure in various research disciplines. To allow for deeper text analysis, we additionally provide the `Clustering` action which enables users to apply clustering algorithms on the data based on their textual similarity. We provide K-Means and hierarchical clustering (Pedregosa et al., 2011) as algorithm choices and support Tf-Idf and SBERT embeddings (Reimers and Gurevych, 2019) as text representations.

[7] This allows direct comparison among multiple adapters.

[8] We currently focus on (pairwise) text classification tasks.

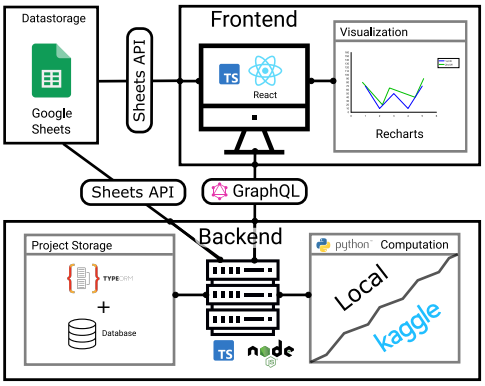

Figure 3: The AdapterHub Playground architecture

2.3 Architecture

The architecture of the AdapterHub Playground (Figure 3) is designed to be easy to setup and requires a minimal set of dependencies. The tool is based on three main components; `Frontend`, `Backend` and `Data Storage`. A user interacts with the frontend and triggers various actions as described in §2.2. The backend receives instructions via the frontend and manages the execution on the computational resource. Our application additionally hosts a local database for user and project management.

We chose GoogleSheet as text data storage component due to its similarity to established easy-to-use spreadsheet applications. It supports a variety of import and export mechanisms which simplify the data management process, especially for non-technical users. GoogleSheet also reduces storage requirements on the local computational resource, keeping the application lightweight and manageable. Finally, the Sheets API[9] provides a programmatic interface for communication. Although this requires users to use a Google account, we argue the advantages compensate for this restriction.

Below, we describe the technical details of the implementation for the specific components.

Frontend. The frontend provides the visual interface for management (i.e. creation, editing, deletion) of projects and their respective actions. After login with an authentication token[10], a webpage lists all user projects. By selecting a project, a corresponding details page allows actions to be managed (see §2.2) and visualizations to be created using project-specific data storage.

The frontend is implemented using the React[11] framework and is written in TypeScript[12]. The frontend design is based on Bootstrap[13]. Communication with the backend is realized via the GraphQL query language. The data is retrieved using the Sheets API and can be visualized via Recharts[14] which offers seamless integration of the D3[15] visualization library within the React framework.

Backend. The backend organizes the storage of application-relevant objects (i.e. users, projects, tasks) and manages both dynamic code generation and execution. User credentials, projects, and tasks are stored in a SQL database. When an action is executed in the frontend, the backend server loads the task-specific code template and dynamically integrates parameter information provided for the individual task. Depending on the choice of the computational node, the generated Python script is scheduled for execution either locally or on a Kaggle compute node via the KaggleAPI[16].

The backend is implemented using Node.js[17] and TypeScript. For application-relevant data, any TypeORM[18]-supported database (e.g. MySQL, PostgreSQL, etc.) can be used. Communication with data storage is realized via Sheets API.

3 Few-shot scenario

Several prominent tasks in NLP such as sentiment analysis (Socher et al., 2013; Rosenthal et al., 2017), stance detection (Mohammad et al., 2016; Schiller et al., 2021) or identifying semantically similar texts (Cer et al., 2017; Agirre et al., 2012) are of great interest in social science research (Boumans and Trilling, 2016; Beck et al., 2021; van Atteveldt and Peng, 2021). We therefore replicated two scenarios, namely sentiment analysis and semantic textual similarity.

We envision a situation where a user has collected textual data (e.g. sentence-level) for a given task and wishes to perform analysis using a text classification pipeline. A labeled test set to evaluate the performance of the classifier, and further training data is available.

[9] https://tinyurl.com/SheetsAPI

[10] Depending on the chosen backend solution, this can be a JSON file provided by the system administrator of the backend server or the authentication token provided by Kaggle.

[11] https://reactjs.org/

[12] https://www.typescriptlang.org/

[13] https://getbootstrap.com/

[14] https://recharts.org

[15] https://d3js.org/

[16] https://www.kaggle.com/docs/api

[17] https://nodejs.org

[18] https://typeorm.io/

3.1 Experiments

Data. For demonstration purposes, we recreated the above-mentioned scenario using existing datasets for both tasks. For sentiment analysis, we use the dataset by Barbieri et al. (2020). In particular, we retrieve text for the Twitter Sentiment Analysis dataset which was originally used for the Semeval2017 Subtask A (Rosenthal et al., 2017). At time of writing, the AdapterHub provides mostly pretrained adapters for binary sentiment classification (*positive*, *negative*). Thus, we discarded all items labeled as *neutral* from the dataset and are left with 24,942 Tweets for training and 6,347 Tweets for testing.

For semantic textual similarity, we use the dataset by Lei et al. (2016) which is a set of pairwise community questions from the AskUbuntu[19] forum annotated for duplicates. Specifically, we use the question titles of the human-annotated development (4k) for training and the test instances (4k) for testing.

Setup. For binary sentiment classification, we use the AdapterHub to obtain three different adapters which were previously trained (Pfeiffer et al., 2021a) on English datasets from the movie review domain. The `IMDB` adapter was fine-tuned on the dataset by Maas et al. (2011), the `RT` adapter was trained on the Rotten Tomatoes Movie Reviews dataset by Pang and Lee (2005), and the `SST-2` adapter was trained using a binarized dataset provided by Socher et al. (2013).

For semantic textual similarity, we obtained the `MRPC` adapter trained on the paraphrase dataset by Dolan and Brockett (2005) and the `QQP` adapter trained on the Quora Duplicate Question dataset.[20]

The experiments were conducted using the AdapterHub Playground without writing any code. We experiment with different training dataset sizes, repeated three times with different subsets of the training data randomly selected for each run.[21]. We evaluated statistically significant differences ($p < 0.05$) between zero-shot and few-shot results of each adapter using a paired Bootstrap test (Efron and Tibshirani, 1994).

3.2 Results

The results for both tasks are shown in Table 1.

Sentiment Analysis. The overall best performance is achieved by the `SST-2` adapter, simultaneously the most robust performance in terms of the standard deviation across different runs and varying amounts of training data. This is most likely due to the substantially larger size of the initial training data (`SST-2`: 67k, `RT`: 8k, `IMDB`: 25k) for the adapter. Although, on average, for all adapters zero-shot performance could be outperformed using a minimum of 10 instances, the differences between individual runs vary largely and statistically significant improvements are only achieved using a larger number of training instances (e.g., at least $N \geq 100$ for `SST-2`). We find using a small number of annotated examples ($N \leq 50$) leads to worse performance compared to zero-shot performance (N=0) and to less robust results across runs with randomly sampled training data. Providing 1,000 training samples leads to significant improvements for adapters `IMDB` and `SST-2` but only providing the full dataset results in statistically significant improvements for all adapters.

Semantic Textual Similarity. The performance gap between both adapters is large, with a difference of 42.10 in the zero-shot setting, favoring `QQP`. The results for the `MRPC` adapter show no clear tendency to improve as the training data size grows, with performance peaking at 50 training instances. Most surprisingly, using 1,000 or all available training samples (4k) leads to a severe performance decrease. For the `QQP` adapter, performance variations are minimal and none of the few-shot experiment settings leads to a significant improvement over zero-shot performance.

Summary. Poth et al. (2021) investigated the effects of intermediate task fine-tuning in adapter settings. They showed that domain similarity, task type match and dataset size are good indicators for the identification of beneficial intermediate fine-tuning tasks. Our experiments confirm this finding although we cannot observe consistent improvement with larger training data size. Thus, more research on robust few-shot learning is necessary.

In contrast to relying on off-the-shelf tools for automated content analysis, our application enables direct evaluation of both zero-shot and few-shot performance of existing pretrained adapters. This is especially helpful for assessment of the applicability of such models for interdisciplinary research (Grimmer and Stewart, 2013) but can also be used to test robustness with varying hyperparameter configurations.

[19] `https://askubuntu.com/`

[20] `https://tinyurl.com/quora-qp`

[21] See Appendix B for experimental details.

Adapter	0	5		10		20		50		100		1,000		N
IMDB	71.99	65.40	±2.08	**72.25**	±14.78	67.51	±11.25	71.37	±4.93	81.87	±2.49	84.10	±5.34	88.36
RT	76.24	72.33	±0.97	**76.76**	±10.08	67.38	±10.09	67.44	±5.57	76.88	±4.22	82.64	±6.01	90.50
SST-2	84.61	84.53	±0.15	**86.23**	±2.91	84.22	±0.53	82.33	±2.41	83.54	±0.61	88.19	±2.08	92.04
MRPC	31.18	**31.64**	±5.37	29.22	±0.08	28.46	±1.30	38.57	±3.66	36.66	±8.54	28.31	±3.18	26.23
QQP	73.28	73.10	±0.16	73.19	±0.06	72.79	±0.44	71.08	±1.17	69.60	±0.48	73.01	±0.30	**73.68**

Table 1: Few-shot performance of various pretrained adapters from AdapterHub using increasing size of training data. Underlined scores are significantly ($p < .05$) better than their zero-shot counterpart. Bold scores resemble experiments with minimum training data required for outperforming zero-shot performance of respective adapter. All numbers are accuracy scores. N is for using all available training data.

4 Usability Study

AdapterHub Playground is designed to be simple to use, requiring minimal training effort and technical knowledge. While we followed these principles throughout the conception and implementation of the application, we also evaluated the usability with users from our target group. Therefore, we followed the approach by Hultman et al. (2018) and let study participants conduct a series of tasks which were designed to reflect a use-case scenario as described in §3. Afterwards, we used a questionnaire to capture their experiences.

Participants. We recruited study participants (N=11) from the communication science field, the majority of whom were (post)graduate-level researchers at a university (two Professors, two PostDocs, six PhDs, one B.Sc.). Our data suggests that the participants have limited or no understanding of the technical computer science concepts but can envision themselves using the AdapterHub Playground (for details see Appendix D). Thus, our participants belong to one of the target groups we aim to aid with this application.

Procedure. The participants were provided a textual description of several tasks to be completed.[22] Users were asked to complete a `Training` and `Prediction` action in a sentiment analysis scenario. We provided both labeled test data and unlabeled training data again using the dataset by (Barbieri et al., 2020).[23] After completing the tasks, we asked the participants to complete a questionnaire targeting their experience with the tool.

Results The participants were asked to assess the difficulties they faced on a five-point Likert scale, specifically, their experience with the overall task, the navigation of the application, and the difficulty of the task description (see Figure 4). The majority of participants found the task and the navigation of the application to be simple.

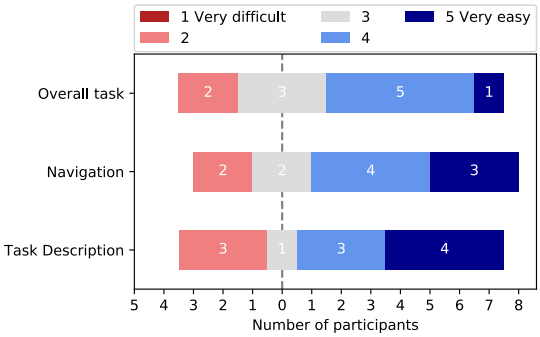

Figure 4: Participants' estimation of difficulty.

Three participants found the task description difficult to understand. We note here that the task description did not explain each individual navigation step in the application. This was designed on purpose - both to reduce the reading volume of the task description and to evaluate the accessibility of each feature of the application.

We further asked the participants about the difficulty of each individual task they had to solve, i.e. prediction, annotation, and training, on a five-point Likert scale ranging *very difficult* (1) to *very easy* (5). Participants had the least trouble with the prediction action (91% voted either category 5 or 4; none voted category 1 or 2). Despite the training action being technically similar to the prediction action, participants perceived it as more difficult with only 64% selecting easier categories (4 and 5) and 27% of the participants being undecided (category 3). This is most likely due to some participants having issues finding the downloadable zip file which required opening the action detail page after training (we received this information as feedback in a free-answer form).

[22] We provide the full task description in the Appendix D.

[23] Our focus is to evaluate the usability of the AdapterHub Playground application. Therefore, we did not require the participants to import the data on their own but rather provided them links to Google Docs containing the imported data.

5 Conclusion and Future Work

Open-access dissemination of SotA NLP models has accelerated their use and popularity, yet the required technical proficiency to apply them remains a limiting factor for their democratisation. To mitigate this, we introduced the AdapterHub Playground application which provides an easy-to-use web interface for no-code access to light-weight adapters for text classification tasks. Our system is built on-top of the open-source AdapterHub library and uses a dynamic code generation approach. We demonstrated the features of the application using two exemplary use-case scenarios and evaluated its usability in a user study with researchers from communication sciences. In addition to providing execution on third-party hardware, we also enabled a self-hosted computational instance.

As future work, we plan to extend the application with dynamic user control over all hyperparameter specifications in the expert mode. To support users in efficient sampling of profitable training instances, we plan to investigate the integration of active learning methods (Yuan et al., 2020). A running instance of our tool can be found under `https://adapter-hub.github.io/playground`. and the open-source code[24] under `https://github.com/Adapter-Hub/playground`.

Broader Impact Statement

Intended Use Our proposed application can be used in several ways and by different audiences. First of all, it allows evaluating the performance of already existing fine-tuned adapters for various prominent text classification tasks on hold-out data, possibly from another domain. Further, one can provide annotated data for any of the supported tasks and continue training the corresponding adapter. Training procedures can be repeated using different hyperparameters to investigate the effect of those on the prediction performance. This makes our application interesting for both our target group, i.e. researchers outside of NLP using text classification methods, as well as NLP researchers interested in comparing various adapter models without setting up the required codebase to do so.

Possible Risks Primarily, the goal of our application is to lower the technical entry barrier for users interested in using state-of-the-art text classification models. These users usually also lack the expertise to evaluate all aspects of the language understanding capabilities of such a model, as compared to researchers from within the NLP domain. Rightfully, one can argue that publishing such an application increases the opportunities to develop more *bad* black box models, caused by limited evaluation and missing expertise. This can lead to severe misjudgements if conclusions are drawn based on predictions of such a model.

While we cannot eliminate this risk, we would like to raise some points which, in our opinion, put it into perspective with regard to the benefits of having such an application.

From a broader perspective, the AdapterHub Playground contributes to the democratization of access to the latest NLP research by simplifying the process of applying language model adapters for training and prediction. This is especially helpful for interdisciplinary research where the applied text classification tools often rely on outdated methods (Stoll et al., 2020) or off-the-shelf tools (Sen et al., 2020). As a consequence, details about the model architecture, training procedure or out-of-domain performance are mostly omitted. While this does not imply low performance on hold-out data per se, it limits the possibilities for model evaluation and demands a certain level of trust from the end user. In many cases, adapting the model to the target domain is not possible or requires some technical proficiency. In addition, these models are often trained once-and-for-all while our framework allows for an interactive approach to evaluate model performance and offers the rich variety of pretrained adapters being available from the community-driven AdapterHub.

Further, we argue that advancements in NLP research should be made available to the researchers most profiting from them as soon as possible - not only for the sake of accelerating research outside of NLP but also to enable a feedback loop informing NLP researchers about the shortcomings of such models. While the generalization capabilities of state-of-the-art language models are subject to increased scrutiny within NLP (Sanchez et al., 2018; Gururangan et al., 2020; Tu et al., 2020), the datasets and tasks to test them often originate from within the same community, thereby introducing a selection bias (Ramponi and Plank, 2020). By enabling interdisciplinary researchers to eval-

[24] Licensed under Apache License 2.0.

uate NLP models without the technical barriers involved, we are able to gain more insights about the robustness and out-of-domain performance of these models. Our application is a first step into this direction.

Acknowledgements

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group KRITIS No. GRK 2222/2 and by the LOEWE initiative (Hesse, Germany) within the emergenCITY center. We want to thank the participants who volunteered to participate in our user study as well as Luke Bates and Ilia Kuznetsov for their valuable feedback.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada. Association for Computational Linguistics.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. 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 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Tilman Beck, Ji-Ung Lee, Christina Viehmann, Marcus Maurer, Oliver Quiring, and Iryna Gurevych. 2021. Investigating label suggestions for opinion mining in German covid-19 social media. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–13, Online. Association for Computational Linguistics.

Jelle W Boumans and Damian Trilling. 2016. Taking stock of the toolkit: An overview of relevant automated content analysis approaches and techniques for digital journalism scholars. *Digital journalism*, 4(1):8–23.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

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.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Justin Grimmer and Brandon M Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political analysis*, 21(3):267–297.

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.

Wenjuan Han, Bo Pang, and Ying Nian Wu. 2021. Robust transfer learning with pretrained language models through adapters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 854–861, Online. Association for Computational Linguistics.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jiawei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th*

International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 2208–2222, Online. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Gretchen Hultman, Reed McEwan, Serguei Pakhomov, Elizabeth Lindemann, Steven Skube, and Genevieve B Melton. 2018. Usability Evaluation of an Unstructured Clinical Document Query Tool for Researchers. *AMIA Summits on Translational Science Proceedings*, 2018:84.

Anne Lauscher, Olga Majewska, Leonardo F. R. Ribeiro, Iryna Gurevych, Nikolai Rozanov, and Goran Glavaš. 2020. Common sense or world knowledge? investigating adapter-based knowledge injection into pretrained transformers. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 43–49, Online. Association for Computational Linguistics.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez. 2016. Semi-supervised question retrieval with gated convolutions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1279–1289, San Diego, California. Association for Computational Linguistics.

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 preprint*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. A dataset for detecting stance in tweets. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3945–3952, Portorož, Slovenia. European Language Resources Association (ELRA).

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021a. 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.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021b. UNKs everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint*.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, Online. Association for Computational Linguistics.

Clifton Poth, Jonas Pfeiffer, Andreas Rücklé, and Iryna Gurevych. 2021. What to pre-train on? Efficient intermediate task selection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10585–10605, Online

and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Alan Ramponi and Barbara Plank. 2020. Neural unsupervised domain adaptation in NLP—A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6838–6855, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pretrained language models for AMR-to-Text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4269–4282, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021a. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021b. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Andreas Rücklé, Jonas Pfeiffer, and Iryna Gurevych. 2020. MultiCQA: Zero-shot transfer of self-supervised text matching models on a massive scale. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2471–2486, Online. Association for Computational Linguistics.

Ivan Sanchez, Jeff Mitchell, and Sebastian Riedel. 2018. Behavior analysis of NLI models: Uncovering the influence of three factors on robustness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1975–1985, New Orleans, Louisiana. Association for Computational Linguistics.

Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2021. Stance detection benchmark: How robust is your stance detection? *KI-Künstliche Intelligenz*, pages 1–13.

Indira Sen, Fabian Flöck, and Claudia Wagner. 2020. On the reliability and validity of detecting approval of political actors in tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1413–1426, Online. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Anke Stoll, Marc Ziegele, and Oliver Quiring. 2020. Detecting impoliteness and incivility in online discussions: Classification approaches for german user comments. *Computational Communication Research*, 2(1):109–134.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. 2020. An empirical study on robustness to spurious correlations using pre-trained language models. *Transactions of the Association for Computational Linguistics*, 8:621–633.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages

2302–2315, Online. Association for Computational Linguistics.

Wouter van Atteveldt and Tai-Quan Peng. 2021. *Computational Methods for Communication Science*. Routledge.

Wouter van Atteveldt, Mariken ACG van der Velden, and Mark Boukes. 2021. The Validity of Sentiment Analysis: Comparing Manual Annotation, Crowd-Coding, Dictionary Approaches, and Machine Learning Algorithms. *Communication Methods and Measures*, pages 1–20.

Marko Vidoni, Ivan Vulić, and Goran Glavaš. 2020. Orthogonal language and task adapters in zero-shot cross-lingual transfer. *arXiv preprint arXiv:2012.06460*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418, Online. Association for Computational Linguistics.

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.

Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020. Cold-start active learning through self-supervised language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7935–7948, Online. Association for Computational Linguistics.

A Frequently Asked Questions

What are the requirements to use the AdapterHub Playground? The most basic usage requirement is an up-to-date modern web browser[25]. To use the application without setting up your own computing instance, one needs to create a Kaggle account and download the API Token for login. We provide information on setting up a local compute instance at https://github.com/Adapter-Hub/playground. As we use GoogleSheet as data hosting platform, the user needs an active Google account.

If used for prediction, textual target data must be uploaded to a GoogleSheet and linked with a project within the application. For training, each text must be additionally labelled according to the target task's label matching schema. While we also provide information about each supported task on a separate page in the application, we expect the user to have a basic understanding of the procedure of the targeted task (e.g. *Sentiment Analysis* is about predicting the sentiment tone of a given text).

How is a user able to identify label mismatch? For each supported task, we provide the necessary label matching information within the dialogue to create a new Action (e.g. in Figure 2). If the user-provided labels in the Google data sheet do not match the selected task or adapter architecture, an error message will be provided giving information about the indices of the mismatched data points.

How could a new user determine the task for their data? In general, this application is intended for users who know what type of predictions (i.e. the task) they want to apply on their data. We provide support for a subset of (pairwise) text classification tasks from AdapterHub.ml with the goal to cover the most prominent ones used in interdisciplinary research. However, we also provide basic information about each supported task on a separate page in the application.

What if my task is not supported in the AdapterHub Playground? In general, we can only provide support for the classification tasks which are covered on Adapterhub. We have selected a subset of tasks which we deem to be of interest in interdisciplinary research (e.g. computational social science). Integration of new tasks is possible by extending the application which requires some technical background in coding and web development. If you are a researcher and lack the technical proficiency to do so, we encourage you to get into contact with us to find out if and how we can integrate your task.

Which pretrained adapter should be used? This is still an open research question and we refer to the literature for more details (Phang et al., 2018; Pruksachatkun et al., 2020; Poth et al., 2021). However, there are some heuristics which can be followed. Regarding adapters, it has been shown that domain similarity (e.g. training and test data are both from Twitter) and training dataset size (the more the better) can be indicators for good transfer performance (Poth et al., 2021).

How should hyperparameters be set? Hyperparameter optimization for machine learning is a research field of its own and there is no one-size-fits-all solution to this. Especially for users without experience in tuning ML models identifying reasonable hyperparameter values might seem rather arbitrary.

Currently, we support tuning the learning rate and the number of epochs. In general, if the learning rate is high the training may not converge or even diverge. The changes in the weights might become too big such that the optimizer will not find optimal values. A low learning rate is good, but the model will take more iterations to converge because steps towards the minimum of the loss function are tiny. In practice it is good strategy to test different (high and low) learning rates to identify their effect on the model performance.

One epoch describes a full cycle through the entire training dataset. A single epoch can sometimes be enough to improve performance significantly and training text classification adapters longer than for 10 epochs rarely provides substantial improvements. We recommend testing different numbers of epochs (between 2 and 5) to evaluate if longer training is beneficial for the task at hand.

B Training Details

We did not perform any hyperparameter optimization for our experiments and used the default settings in the AdapterHub Playground application. We adopted a learning rate of 1e-4 from related work (Pfeiffer et al., 2020a) and trained each adapter for three epochs. In Table 2 we provide the

[25] We tested the application using Desktop Firefox and Desktop Chrome.

Adapter	Task	Pretrained Language Model Identifier	Architecture
IMDB	Sentiment Analysis	distilbert-base-uncased	Pfeiffer
RT	Sentiment Analysis	distilbert-base-uncased	Pfeiffer
SST-2	Sentiment Analysis	bert-base-uncased	Houlsby
MRPC	Semantic Textual Similarity	bert-base-uncased	Houlsby
QQP	Semantic Textual Similarity	bert-base-uncased	Houlsby

Table 2: Adapter architecture details for each specific task.

respective adapter architectures which were used for each specific adapter.

C Extensibility

Extending the AdapterHub Playground with a new text classification task requires adaptations to both the frontend and backend.

The repository supports a deployment workflow which will update a configuration file with all relevant information from the AdapterHub. This enables that all tasks and their corresponding pretrained adapters (with a classification head) are potentially available within the AdapterHub Playground. The tasks for these adapters are filtered based on a predefined set of tasks which should be available to users of the application. Within the application, the filter list needs to be adapted such that the new task is not filtered during startup of the application. Additionally, the task name and its description need to be added to the frontend code as well as the label mapping information. In the backend we need to add the label mapping and the list of supported tasks such that the evaluation computation is correct.

We provide the technical details within the code repository at https://github.com/Adapter-Hub/playground.

D Usability Study

D.1 Participants

As can be seen in Figure 5, most participants have only a basic understanding of the technical concepts related to machine learning or natural language processing. However, it is likely they have experience with annotating data. We further asked them if they can envision using the AdapterHub Playground application in their research. Slightly more than half gave a positive answer (54%) and the rest were undecided; no participant claimed they would never use our application.

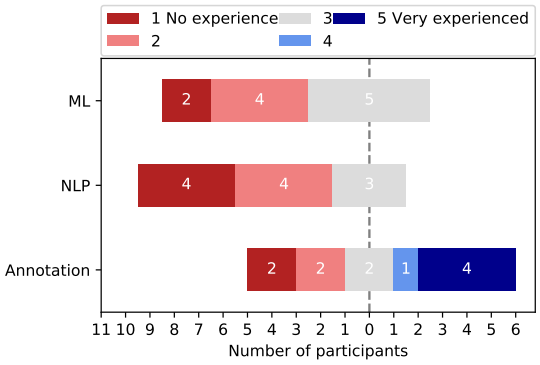

Figure 5: Participants' experience with underlying technical concepts.

Thus, we conclude that our participants belong to our target group.

D.2 Instructions

We provide the instructions for the usability study in Figure 6.

User study for the AdapterHub:Playground application

Study Details

This study is conducted for research purposes by the Ubiquitous Knowledge Processing (UKP) Lab of the Technical University of Darmstadt. Your participation is completely voluntary and you are allowed to cancel at any time. The goal of this study is to evaluate the usability of the Adapterhub:Playground application. The whole study will take approximately 25 minutes. During the study you will be asked to complete certain tasks in the application and fill out a questionnaire afterwards.

Contact Person:
Tilman Beck, M.Sc.
S2|02 B104
Hochschulstraße 10
64289 Darmstadt
beck@ukp.informatik.tu-darmstadt.de
+49 6151 16-25294

Preliminary

To take part in this study, you need to register on Kaggle.com. After registering, click on your profile picture in the top right corner and choose `Account` in the menu. Then, scroll down to the `Phone verification` field and provide your mobile number for verification (**Important**: without verification we can't use Kaggle as computational resource). Next, scroll to the `API` field, click `Create new API token` and download the `kaggle.json` file to your computer. You are now ready to take part in the study.

Data:

User ID: <user-ID>
D1: <url-to-data-D1>
D2: <url-to-data-D2>

Task Description

Prediction

You are provided a set of Social Media posts and are asked to analyze the sentiment in these texts. The AdapterHub:Playground offers you a tool which allows you to make use of machine learning models to analyze these Social Media posts. They are specialized on a variety of natural language processing tasks such as sentiment analysis.
Please evaluate the performance of those models on the first set of Social Media posts we provided (see D1 above). This Googledocs file does not only contain the Social Media posts, but also their respective sentiment label (`positive` or `negative`). Log into the AdapterHub:Playground tool, create a new project and insert the above mentioned link to the Googlesheet. Enter your new project by clicking on your chosen name in the list, then start a new task with type `Prediction` for `Sentiment Analysis`. In the expert mode you can specify additional details about the model selection like the dataset the model was trained on (e.g. `SST-2`) or the model architecture (e.g. `bert-base-uncased | pfeiffer`). Upon starting the task, the tool will write its predictions directly into the provided Googledocs in the column you provided. This may take some minutes. Once the task is finished, you can investigate the performance of the task using the measures `Accuracy` and `F1`.

Training

To further improve the performance of the models, the AdapterHub:Playground tool allows you to train existing models with annotated data. Therefore, you are provided a second set of Social Media posts without labels (see D2 above).
Please, visit this Googledocs and start with providing labels (type `positive` or `negative`) in the column `annotation` for the corresponding texts in the first column (`input1`). We recommend annotating at least 10-15 texts, but you are free to annotate more texts.
After you finish your annotation, return to the AdaterHub:Playground tool and create a new project. To do so, please insert the link to the Googledocs (D2 above) where you have made annotations on your own (please be aware, the link is different to the first Googledocs).
Now, create a new `Training task` for `Sentiment Analysis`. Choose the same model selection details as for the previous prediction task and start the training task by submitting. Once the training is finished, download the trained model (`trained_adapter.zip`) to your computer. Congratulations, you have just trained your own sentiment analysis model! Now, please evaluate if your own model achieves better performance than the off-the-shelf model from AdapterHub:Playground. Please, use it to create predictions on your initial set of Social Media posts. Therefore, repeat the process of the first part of this study (Prediction), i.e. create a new project, start a new task with type `Prediction` for `Sentiment Analysis`. However, this time make use of the `Upload Adapter` function in the expert mode and upload the previously downloaded file (`trained_adapter.zip`). After completion of the task, investigate the performance measures again.

Now, please answer the questions in this questionnaire:
<url-to-questionnaire>

Thank you very much for your participation!

Figure 6: Instructions for the participants of the user study.