Collaborative Development of Modular Open Source Educational Resources for Natural Language Processing

Matthias Aßenmacher[1,2], Andreas Stephan[3,4], Leonie Weissweiler[2,5], Erion Çano[3,6], Ingo Ziegler[5,7], Marwin Härttrich[5], Bernd Bischl[1,2], Benjamin Roth[3], Christian Heumann[1], Hinrich Schütze[2,5]

[1]Department of Statistics, LMU Munich, [2]Munich Center for Machine Learning (MCML),
[3]Faculty of Computer Science, University of Vienna,
[4]UniVie Doctoral School Computer Science, Vienna, Austria,
[5]Center for Information and Language Processing (CIS), LMU Munich,
[6]Department of Computer Science, Paderborn University,
[7]Department of Computer Science, University of Copenhagen

Correspondence: matthias@stat.uni-muenchen.de

Abstract

In this work, we present a collaboratively and continuously developed open-source educational resource (OSER) for teaching natural language processing at two different universities. We shed light on the principles we followed for the initial design of the course and the rationale for ongoing developments, followed by a reflection on the inter-university collaboration for designing and maintaining teaching material. When reflecting on the latter, we explicitly emphasize the considerations that need to be made when facing heterogeneous groups and when having to accommodate multiple examination regulations within one single course framework. Relying on the fundamental principles of OSER developments as defined by Bothmann et al. (2023) proved to be an important guideline during this process. The final part pertains to open-sourcing our teaching material, coping with the increasing speed of developments in the field, and integrating the course digitally, also addressing conflicting priorities and challenges we are currently facing.

1 Introduction

The rapid acceleration of developments in natural language processing (NLP) research, starting with the introduction of the Transformer (Vaswani et al., 2017) in 2017, also poses a challenge to designing appropriate curricula for formal education in this area. Numerous longstanding paradigms have been replaced by new technologies enabled by a new class of (autoregressive) large language models (LLM; OpenAI, 2022, 2023; Anil et al., 2023; Touvron et al., 2023; AI@Meta, 2024) alongside massively increased computational capacities. The curriculum of deep learning (DL) courses for NLP before 2017 mostly consisted of teaching different types of word embedding models (e.g. Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017) as building blocks within specialized neural network architectures. This often pertained to employing and tuning recurrent neural networks (RNN) for solving various kinds of token- or sequence-level tasks. With the advent of transformer-based transfer learning models (Radford et al., 2018; Devlin et al., 2019; Raffel et al., 2020) developments sped up, the field has become a lot more diverse,[1] and hence course curricula require significant updates/enhancements more and more frequently. We believe that collaboration across and within universities (across different faculties and departments) can be one way to combat the resulting challenges. Furthermore, bringing together researchers with multifaceted backgrounds and different levels of seniority for co-creating lecture material can help create a more inclusive course suitable for a broad audience of undergraduate and graduate-level students from various fields.

2 Related work

Open Educational Resources. The number of Massive Open Online Courses has been rapidly increasing over the past decade, be it in machine learning (ML; Ng, 2021; Google, 2023) in general or in NLP specifically. Probably one of the most notable *applied* NLP courses is courtesy of Hugging Face (Hugging Face, 2022). It provides a hands-on introduction to the state-of-the-art (SOTA) software package for NLP, but in doing so it does not discuss the theoretical foundations in much detail. Other popular and very well-taught courses,

[1]This refers to both the kind of problem statements tackled with NLP technology and the academic audience of students and researchers interested in taking NLP-related courses.

like e.g. the Stanford CS224N lecture (Stanford NLP Group, 2024) or the `deeplearning.ai` NLP course (DeepLearning.AI, 2023) provide great theoretical (and applied) introductions to NLP, but are not truly *open source*: Neither do any of these courses provide open and modifiable sources of their lectures, nor do they explicitly specify the license(s) under which their material is released. The latter even requires registering at a platform and only provides the material as videos, not even releasing versions of their lecture material as PDFs. So while these courses can be considered open, they are unfortunately not fully *open-source* (Bothmann et al., 2023).

Open *Source* Educational Resources. According to Bothmann et al. (2023), open source educational resources (OSER) are characterized by a set of core principles (which also served as guidelines for the development of our course) motivated by best practices from open source software development. This is very well reflected in the following principles:

- Develop course material collaboratively.
- Make your sources open and modifiable and use open licenses.
- Release well-defined versions and maintain change logs.

Other principles Bothmann et al. (2023) define in their work are more focussed on pedagogical aspects and on the goal of enabling as many people as possible to learn from the developed material in their own way and at their own speed:

- Modularization: Structure the material in small chunks and disentangle theory and implementation
- Define prerequisites and learning goals
- Foster self-regulated learning and enable feedback from everyone

A notable NLP OSER is the course created for the software library *spaCy* (Montani, 2019), where developers comply with most of the OSER principles. A drawback of this course, however, is its high entanglements with one specific software library (spaCy), severely limiting the modular reuse of the resources. Further, "Deep Learning for Coders with Fastai and PyTorch" (Howard and Gugger, 2020) presents a fully open-source and modifiable online course (fast.ai, 2020) with extensive coverage of various DL applications, it focuses on NLP in just one chapter. Consequently, it falls short of delivering the comprehensive depth expected from a university-level NLP lecture series.

Course creation: Machine Learning vs. NLP. Further challenges going along with working on creating/teaching courses for NLP compared to ML pertain to the rapid speed of developments. In typical ML courses, there is a more or less agreed-upon set of topics being taught in most of the introductory/basic courses, including supervised learning methods (classification/regression), unsupervised learning, tree-based methods as well as approaches for hyperparameter tuning or resampling strategies. Building upon this foundation, various special topics such as, e.g., boosting, gaussian processes, or even neural networks can be flexibly added/exchanged, depending on the focus of the respective target audience or tailored to a certain program of studies. A prime example is the "Introduction to Machine Learning (I2ML)" course by Bischl et al. (2022), a collection of three ML courses taught at LMU Munich. The creators rely on the stable content of their undergraduate-level course, build up two M.Sc. level courses on top of this foundation, and open-source everything on one central platform.[2] This course perfectly shows the stability of the fundamentals for teaching ML while simultaneously stressing the modular extensions one can build upon these fundamentals. In NLP, however, the fundamentals are to some extent subject to change, since new training techniques and new model capabilities are constantly emerging, given the fluid fast-moving nature of the field. In the subsequent chapters, we will thus describe the rationale behind our design choices, try to make the underlying thought processes transparent, and illustrate the resulting OSER we created.

3 Course Design Principles

The design of our OSER relies on a set of principles laid out in greater detail in the following subchapters: First, we want the course to be created as a modular system allowing every partnering institution to adjust the teaching material to their specific needs and to change it between different iterations of the course. This encompasses a set of *core modules* (e.g. the Transformer) as well as a multitude

[2] `https://slds-lmu.github.io/i2ml/`

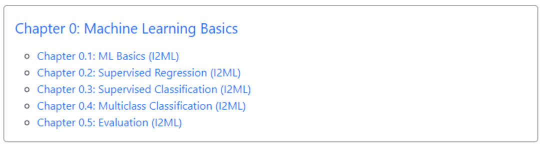
of more elaborated, rather optional, and audience-specific modules (e.g. Multilinguality). Second, we intend to provide the students with a set of challenging (programming) assignments while trying to balance the trade-off of the following question: What are longstanding and established concepts that need to be taught to students, and which are just of short-term importance?

Additionally, we want the students to be well-prepared for our course and to have the right expectations: Figure 1 shows our "Module 0", directing the students to several core chapters of the I2ML course (Bischl et al., 2022) for getting familiar with the machine learning basics.

3.1 Modularity of Teaching Material

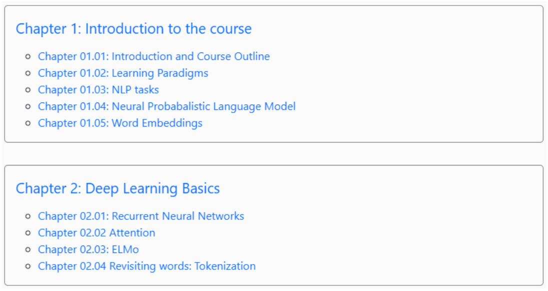
Bringing together different universities or study programs for a joint teaching project inherently requires building a modular system. This enables every party involved in this endeavor ("inside use"), as well as everyone else ("outside use") to pick out the parts that are relevant for this specific party in specific situations. For the inside use, we further define a set of *core modules* taught at every institution allowing for sharing the work when creating exams. This most likely results in (a) more comparable examinations ensuring (b) a higher quality of the exam questions while (c) gaining time efficiency during the creation of the exams. The second part of the *optional modules* pertains to topics that are either just targeted at a specific subgroup of the target audience or that are considered "hot topics" that need(ed) to be addressed at a certain point in time. This leads to a larger and more stable set of core modules, while the pool of optional ones is (a) smaller (but potentially growing over time) and (b) more fluid than the former (some modules might be deprecated over time). The second form of modularity pertains to disentangling theoretical concepts and implementation details (Bothmann et al., 2023). The slide sets we provide the audience with contain explanations and mathematical formulas, but rarely Python code. All programming-related tasks and explanations are outsourced to the programming assignments and the corresponding exercise sessions (cf. Sec. 3.3). This modular composition allows the audience to also use our slides alongside other, more software-centric tutorials (e.g. Hugging Face, 2022), to extract suitable parts for combining it with their own teaching material (without having to disentangle it with our code chunks or similar), and helps us in maintaining the material as we do not run into problems with dependencies or debugging ("Do not use literate programming systems everywhere", Bothmann et al., 2023).

3.2 Lecture Content


Chapter 0: Machine Learning Basics

- Chapter 0.1: ML Basics (I2ML)
- Chapter 0.2: Supervised Regression (I2ML)
- Chapter 0.3: Supervised Classification (I2ML)
- Chapter 0.4: Multiclass Classification (I2ML)
- Chapter 0.5: Evaluation (I2ML)

Figure 1: Referencing the prerequisites for the course.

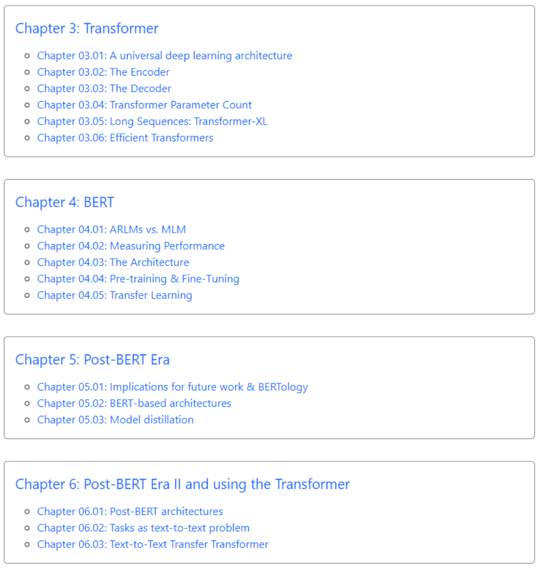

Chapter 1: Introduction to the course

- Chapter 01.01: Introduction and Course Outline
- Chapter 01.02: Learning Paradigms
- Chapter 01.03: NLP tasks
- Chapter 01.04: Neural Probabalistic Language Model
- Chapter 01.05: Word Embeddings

Chapter 2: Deep Learning Basics

- Chapter 02.01: Recurrent Neural Networks
- Chapter 02.02 Attention
- Chapter 02.03: ELMo
- Chapter 02.04 Revisiting words: Tokenization

Figure 2: First lecture block – Providing the heterogenous target group with a unified foundation.


Chapter 3: Transformer

- Chapter 03.01: A universal deep learning architecture
- Chapter 03.02: The Encoder
- Chapter 03.03: The Decoder
- Chapter 03.04: Transformer Parameter Count
- Chapter 03.05: Long Sequences: Transformer-XL
- Chapter 03.06: Efficient Transformers

Chapter 4: BERT

- Chapter 04.01: ARLMs vs. MLM
- Chapter 04.02: Measuring Performance
- Chapter 04.03: The Architecture
- Chapter 04.04: Pre-training & Fine-Tuning
- Chapter 04.05: Transfer Learning

Chapter 5: Post-BERT Era

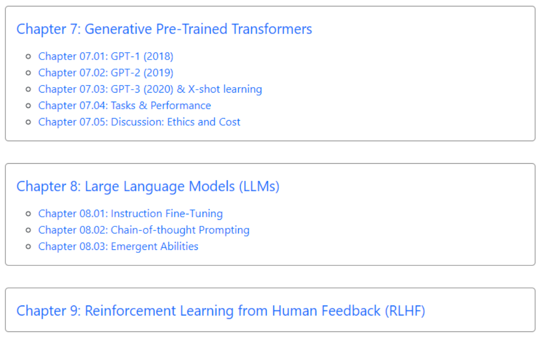
- Chapter 05.01: Implications for future work & BERTology
- Chapter 05.02: BERT-based architectures
- Chapter 05.03: Model distillation

Chapter 6: Post-BERT Era II and using the Transformer

- Chapter 06.01: Post-BERT architectures
- Chapter 06.02: Tasks as text-to-text problem
- Chapter 06.03: Text-to-Text Transfer Transformer

Figure 3: Second lecture block – Introducing important conceptualizations and architectural milestones.

The first block of the lecture material (equivalent to two 90-minute lectures) encompasses two modules for providing the quite heterogenous groups of students (cf. Sec. 4.1) with a unified knowledge base to start with (cf. Fig. 2): First, general NLP-specific topics are introduced before in the second lecture important conceptual topics regarding neural networks are dealt with. Building on this foundation, the next lecture block (cf. Fig. 3; equivalent to four 90-minute lectures) starts by covering the Transformer architecture in-depth since it is the focal methodological topic the students need to understand every tiny detail of. The next module is of similarly central importance, as it introduces BERT as one major cornerstone of the developments leading up to contemporary LLMs. It further deals with important concepts of transfer learning from a birds-eye perspective, the components of pre-training LLMs (objectives, hyperparameters, data), the implications of architectural choices (encoder-only, decoder-only, encoder-decoder), and the (efficient) fine-tuning of such models. The third (and final) central building block of the current version of the lecture is centered around decoder-only LLM architectures (cf. Fig. 4; equivalent to three 90-minute lectures). After having learned how to comprehend *all potential tasks* as a text-to-text problem in the previous block, the students will be introduced to alternative concepts of learning (zero-/few-short learning) before more elaborated alignment techniques (instruction fine-tuning, reinforcement learning from human feedback) are covered.



Chapter 7: Generative Pre-Trained Transformers
- Chapter 07.01: GPT-1 (2018)
- Chapter 07.02: GPT-2 (2019)
- Chapter 07.03: GPT-3 (2020) & X-shot learning
- Chapter 07.04: Tasks & Performance
- Chapter 07.05: Discussion: Ethics and Cost

Chapter 8: Large Language Models (LLMs)
- Chapter 08.01: Instruction Fine-Tuning
- Chapter 08.02: Chain-of-thought Prompting
- Chapter 08.03: Emergent Abilities

Chapter 9: Reinforcement Learning from Human Feedback (RLHF)

Figure 4: Third lecture block – Discussing the capabilities and the inner workings of contemporary LLMs.

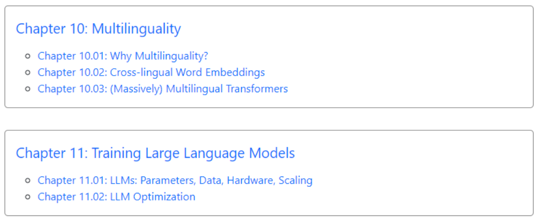
These three central building blocks can be flexibly extended using optional modules based on (a) the needs of the target audience, (b) the fit with the surrounding curriculum of studies, and (c) what is of particular interest based on how current research is developing. The subsequent list of lecture blocks has been employed over the past semesters:

0. *Machine Learning Basics:* Before Module 1 to bring everyone up to speed (if required, cf. Fig. 1).

4. *Multilinguality:* Multilingual alignment techniques for embeddings/pre-trained models, mostly taught at LMU for the computational linguistics (CL) students (cf. Fig. 5).

5. *LLMs in Practice:* Considerations regarding hardware, parameter counts, and scaling (including a guest lecture from industry, cf. Fig. 5).

We will continuously monitor future developments in the field and adjust the course accordingly. This allows us to react flexibly to newly emerging or newly established methods/topics which can be added as further modules whenever we see fit.



Chapter 10: Multilinguality
- Chapter 10.01: Why Multilinguality?
- Chapter 10.02: Cross-lingual Word Embeddings
- Chapter 10.03: (Massively) Multilingual Transformers

Chapter 11: Training Large Language Models
- Chapter 11.01: LLMs: Parameters, Data, Hardware, Scaling
- Chapter 11.02: LLM Optimization

Figure 5: Optional lecture blocks – Multilinguality and further insights into contemporary LLMs.

3.3 Programming Questions

The current design of our programming assignments tries to carefully balance basic understanding, i.e. bringing every student from this heterogeneous group up to speed, against novelty, i.e. conveying (sufficiently) interesting new content that is of interest to the majority of the students. At the same time, the degree of difficulty of the assignments is a further crucial consideration since they partly influence the grading of the students (cf. Sec. 4.2). Trying to get this interplay right, we designed the following five two-week assignments for the iteration in the winter term of 2023.

Assignment 1: Building, Tuning and Evaluating an RNN Model. The goal of the first assignment is to familiarize all the students with the general setup of the training loop for a DL model. We expect the students to be familiar with embeddings and the basic concepts of DL, and with the basic functionalities of Python on a technical side. The intended outcome is that the students become familiar with working with using PyTorch (Paszke et al., 2019) and learn how to set up everything from scratch, so they know what is happening behind the scenes once they start using high-level frameworks like Hugging Face (Wolf et al., 2020) later.

Contentwise, the students are tasked with loading, splitting, and formatting datasets from hosted places, such as Hugging Face, to gain hands-on experience with data preparation. They then learn to build parallelized dataloaders using the PyTorch API to efficiently handle datasets of varying sizes. Following that, students are asked to design a custom RNN model layer-by-layer and to write the training and evaluation processes by hand. This exposes them to the inner workings of PyTorch modules and helps them understand how to connect their own modules to the core "Autograd" functionality. Enhancing their understanding of model evaluation is another goal of this assignment. Consequently, they need to implement a classification metric tracking system and plot and document their results. The assignment provides the class with three hyperparameter configurations intended to produce an underfitted, overfitted, and well-fitted model, lastly prompting them to pinpoint and discuss notable training moments on their plots.

Assignment 2: Building the "Vanilla" Transformer from Scratch. The next important milestone in our curriculum is implementing the vanilla Transformer *from scratch*. We deem this task to be of extremely high didactic importance, as later on students will most likely use the Hugging Face libraries to train or to interact with transformer-based models. Obtaining a proper understanding of the model's inner workings is crucial, as the Transformer itself might be replaced in the future, but in general large, complex models with many hyperparameters are still likely to persist.

Since various well-known teaching-oriented implementations of the vanilla Transformer exist, notably "The Annotated Transformer" (Rush, 2018; Huang et al., 2022) and a chapter of "Deep Learning Notebooks" (Lippe, 2022), we refrain from asking the class to simply implement a working Transformer model. Instead, we provide specific instructions to break down the task into smaller sub-tasks, with clear expectations for intermediate results. This has two key benefits: First, it enables students to identify and debug issues early on, by checking their sub-results against expected outputs and tensor shapes. Second, it encourages students to engage deeply with the paper and code, making it difficult to simply copy from existing online resources. Even if students tried to copy, they would need to make significant changes to fit our instructions, ensuring a contribution to the implementation. To strike a balance between realism and learning value, we fit our assignment requirements into input and output specifications for each module, along with `assert` statements to verify intermediate results in `init` and `forward` methods of each module.

Assignment 3: The Hugging Face Ecosystem. As one central objective of the assignments is to convey hands-on practical knowledge to the students, the third assignment is centered around the Hugging Face ecosystem. The Hugging Face ecosystem has become a de facto standard in the NLP community, providing a unified interface for a wide range of SOTA models and datasets. By working with it, students can gain experience with a powerful tool that simplifies the process of building, training, and distributing NLP models, while facilitating reproducibility and collaboration.

Students were tasked with fine-tuning the encoder-only BERT (Devlin et al., 2019) for classification and the encoder-decoder T5 (Raffel et al., 2020) for text summarization. Through this exercise, students gain hands-on experience with tokenizers, loading, and preparing models from the Hugging Face hub for different tasks. A key focus is on the `Trainer` and its `TrainingArguments` class, where students are supposed to experiment with various techniques to reduce GPU memory usage, including batch size, gradient accumulation steps, gradient checkpointing, and 16-bit floating point data types. Taking it a step further, students are also introduced to the concept of reversing the abstractions provided by Hugging Face, for instance, by inheriting from the `Trainer` class and customizing the loss calculation logic according to our requirements. In the second part, we focus on approaching problems as text-to-text tasks, including the pre-/post-processing steps for summarization. Again, while models might change, implementing the whole pipeline (data-to-model; task description; GPU utilization) is a vital skill for the future.

Assignment 4: Interpretability and Decoding. As the parameter count of NLP models continues to grow, it has become increasingly important to understand the inner workings of these models. Currently, there are many ways to analyze models. The fourth assignment examines the topics of employing a simple interpretability method and investigating decoding strategies, both of which are essential for understanding the models' behavior/biases and improving the trustworthiness of models.

To gain insight into the attention mechanism, students explore the attention patterns of various heads in pre-trained models. By visualizing and comparing the patterns of BERT and GPT-2 (Radford et al., 2019) for the same sentence, students observe how BERT's bidirectional attention differs from GPT-2's left-to-right attention. They then quantify their observations by calculating and visualizing the entropy per head and layer, revealing how individual heads distribute their attention. Next, students implement and calculate importance scores (Michel et al., 2019) per head, visualize their results, and finally use them for pruning, i.e. removing heads below a certain importance score threshold. In the decoding part of the assignment, students load GPT-2 and compare its outputs for a single input prompt using different decoding strategies. They begin by implementing beam search (Vijayakumar et al., 2016) and contrasting it with standard greedy decoding, and then progress to more advanced strategies like top-k (Fan et al., 2018) or top-p (Holtzman et al., 2019) sampling and temperature scaling.

Assignment 5: LLMs and Prompting. Language-to-language is a promising paradigm, likely to persist, as it resonates with human interaction. Therefore, it is central for students to learn how to work with this paradigm. The fifth assignment covers this and showcases key prompting strategies to accomplish various tasks and structured, parseable output formats. Students begin by loading LlaMA-2 (Touvron et al., 2023) and familiarizing themselves with the concept and formatting the system prompt. By modifying the system prompt while keeping the downstream instruction equal, students observe how the model's behavior changes in response to different meta-instructions. Subsequently, students explore zero-shot inference (Radford et al., 2019) and investigate the model performance in a multi-class classification task without structured output. They learn to address the difficulties of parsing errors caused by varying model outputs and format errors, before repeating the same task while employing a batched, JSON input- and output structure. This makes it possible to implement formatting checks and parsing rules to reject certain outputs before encountering unknown outputs. Lastly, the assignment incorporates the few-shot learning paradigm (Brown et al., 2020) by solving a relation extraction problem while enforcing a custom, parseable output style while providing in-context examples in the desired format.

3.4 Grading

Two common approaches to grading coding assignments are automated evaluation through test suites and manual checks. While the benefits of automated test suits are fully automated autonomous testing in little time and certainly unbiased grading, the disadvantages are the requirement of precise task descriptions and little to no variance in allowed outcomes, partially solved or nonexecutable submissions may not be testable or it might require additional effort to define those tests. Manual evaluation, on the other hand, results in time-intensive corrections per submission and can introduce biases during grading. The benefits and disadvantages need to be considered beforehand and should also be considered during the task creation. The optimal approach toward grading depends on the expected number of submissions, the available (human) resources for (a) assignment preparation and (b) correction over the course of the semester, and the probable reuse in future iterations of the course.

Due to our design choice of integrating open-ended interpretations of results and observations, we have opted for a hybrid approach: We leverage the benefits of automated tests for the subset of well-defined tasks with clear expected outcomes while resorting to manual evaluation and grading of the more open-ended parts. This allows us to manually handle cases where the automated tests failed within the second pass. To allow the seamless combination of both grading approaches, textual mistake descriptions alongside their resulting point deductions are collected in one text file per submission. The automated tests log mistakes to this file as predefined textual statements, while mistakes encountered during manual inspection are added in the corresponding format. That way both the final grade of the assignment and the encountered mistakes can be reported back to the student, offering insightful feedback via comments. This allows the student to comprehend the grade and reflect on his/her solution, and misapprehension.

4 Collaboration Across Universities

4.1 Target groups and their prerequisites

The initial target group of the course was third-semester master's students in CL from LMU Munich for whom the compulsory module has been the first touching point with DL. Opening the course to master's students of statistics and data science (Stats+DS) by collaboratively improving and teaching it in 2020 brought in a group with a different background for whom this is contrarily the first touching point with linguistics. Students eligible to take the course at the University of Vienna (UNIVIE) have a multi-faceted background as well: While the majority of the students are enrolled in computer science (CS), there is also a share of students from business analytics in the target group of this course. This leaves us with students that can be (coarsely) categorized into three groups:[3]

- Strong CL background, but not much experience with ML, DL, and programming
- High level of technical and theoretical expertise in ML, DL, and programming, but (presumably) no knowledge about linguistics.
- Some affinity to digital tools and programming, but neither an in-depth formal education in ML/DL nor linguistics.

4.2 Examination requirements

While the modularization of the course makes it relatively straightforward to collaborate in creating the material, differences in examination regulations and grading requirements between universities are a stumbling block. Rules at LMU require us to assess a student's performance via one final examination at the end of the semester, whereas at UNIVIE it is strictly necessary to do 50% of the overall performance assessment during the semester. We manage these discrepancies by introducing three types of (self-)assessments, two of which happen continuously over the semester while the last one pertains to a written test at the end of the semester:

(A) Moodle Quizzes: Multiple-Choice/Cloze-style questions (*on a weekly basis*).

(B) Assignments: Advanced programming tasks (*on a bi-weekly basis*, cf. Sec. 3.3).

(C) Written exam (*90min, end of the semester*).

Despite the strict requirements regarding performance assessment at LMU, it is possible to use assessment types **(A)** and **(B)** for awarding bonus points to the students who complete them successfully. The students were able to achieve a maximum of 9 bonus points (10% of the total points of the 90-minute exam) weighted by the share of the quizzes/assignments they were able to solve correctly. Employment of the bonus points was restricted by one condition: The bonus only counted if a student had passed the exam already without the bonus. Table 1 shows the proportions of the students at LMU who were able to achieve a bonus when entering the written examination[4], highlighting the effectiveness of this type of incentive for working on the intra-semester assessments.[5]

year	2020	2021	2022	2023
# students	52	48	64	38
w/ bonus	57.7%	68.8%	98.4%	81.6%
bonus $> 50\%$	57.7%	54.2%	54.7%	60.5%

Table 1: Relative frequencies of students with bonus points among all students who took the exam at LMU.

There is an important breakpoint to be addressed when looking at the numbers in Table 1: From 2020 – 2022 there were ten assignments (to be completed on a weekly basis) with relatively easy tasks to be completed by the students, i.e. filling in some blanks in otherwise complete Jupyter notebooks. Starting in 2023, the assignments became significantly harder: The number was reduced to five, students were given two weeks to work on each of them and the task was to write the complete code by themselves. While this led to a substantial decrease in the share of students with bonus points among those who took the exam compared to 2022[6], the share of students who achieved over 50% of the bonus remained relatively constant over the whole observation period. For 2023, however, we observe a slight increase in the latter figure (plus 6 percentage points compared to 2022), hinting at the effectiveness of the more challenging assignments. We do not show similar numbers for students from

[3]One can argue that Stats+DS and CS students represent two distinct groups, but we believe that they are sufficiently similar concerning their prerequisites for this course.

[4]Unfortunately it is hard to calculate the share of students dropping out before the exam, since typically many more students enroll to the moodle course just to check out the material compared to the actual number of participating students.

[5]Note, that we only include students who actually took the exam here. Students with a bonus who did not register for the exam (or did not show up to it) are not counted in.

[6]Since the assignments were not substantially changed between 2020 and 2022 there was some "leakage", i.e. more senior students (presumably) passing on the solutions to their successors and thus leading to more students completing (some of) the tasks. This suspicion is supported by the rising numbers (until 2022) in the second row of Table 1.

UNIVIE, as they are obliged to successfully submit the assignments to pass the course. Hence a direct comparison does not make much sense here.

5 Open-Sourcing the Material

A core building block of this course is its public website alongside the complete source code for both website and slides. This differentiates the course from other open teaching resources (cf. Sec. 2) as it enhances its reusability. People are not only able to use the material as-is, but also to modify, extend, and enhance it. An important side effect of this policy is the potential feedback loop that we might hopefully enter at some point in time: Instead of only developing and improving the material ourselves, other parties re-using the material could reach out and become collaborators by contributing via issues or pull requests. The technical setup is kept pretty simple: We use two separate repositories on GitHub for the source code of the material and the source code of the website. We believe this separation helps interested third parties in finding what they are looking for and it also eases the whole development process. Using GitHub as a platform is motivated by its focal nature to the CS and NLP community, hence lowering the barrier for collaboration between the co-developers as well as for interested third parties.

The following list contains links to (i) the material, (ii) the website, and (iii) its source code for the interested reader and for potential collaborators:

(i) `https://github.com/slds-lmu/lecture_dl4nlp`

(ii) `https://slds-lmu.github.io/dl4nlp/`

(iii) `https://github.com/slds-lmu/dl4nlp`

6 Future Challenges and Next Steps

Open-source *everything*? An important case of conflicting priorities pertains to the goal of open-sourcing everything and the interest of providing the students with fair and challenging quizzes/assignments. While open-sourcing the solutions to the coding assignments perfectly aligns with the goal of open-sourcing, it contradicts the secondary goal to some extent as it could discourage students from working on the assignments and incentivize checking out the readily available solutions. Further, it would hinder re-using the same assignments for rewarding bonus points (LMU) or grading the performance over the semester (UNIVIE). Hence for the future, we are currently discussing the following scenarios:

- Keep assignments & solutions closed-source.
- Open-source only assignments w/o solutions.
- Substantially change assignments every semester. while open-sourcing everything.

Unavoidable Lecturer Turnover. Further challenges that will arise in the near future[7] originate from the way academia works. After finishing a PhD, people tend to leave the institution where they conducted their PhD studies either for a postdoc at another institution or for industry. While in the latter case, long-term cooperation on developing OSER together is most certainly not possible due to simply different priorities in an industrial job, also the former case does not automatically warrant further cooperation, although it might be more likely. Thus, we believe it is vital for the persistence of such a project (i) to have a clear ownership structure and consistent credit assignment policies, (ii) to set up lecture chapters as self-contained and independent of the lecturer as possible, and (iii) to create seamless documentation of reasoning behind the most important design choices, the workflows, and the responsibilities of the individual roles.

Related points are addressed by Bothmann et al. (2023), yet from a slightly different angle: They also stress the ownership issue as a crucial point concerning quality assurance and maintaining consistency in the narrative, notation, and correctness of the material. We think that the peculiarities of the academic job market are an important addition to these considerations.

Speed of Developments. Balancing the recency against stability is a major challenge for such courses (cf. Sec. 2). In general, university lectures should cover what can be considered established methodology or consensus in the research community. Examples of such topics can be easily found in the field of "classical" ML, considering concepts like logistic regression, support vector machines, and random forests, just to name a few. For DL and NLP this clear-cut definition proves to be much harder: The (relatively young) concept of embeddings as well as the Transformer (and its parts) can by now be considered fundamental/established, but already as soon as it comes to the encoder-based models surrounding BERT things begin to get complicated. While BERT itself might be a relatively unanimous choice, nearly all its successors can be

[7]Until now the core team has stayed mostly constant.

regarded as debatable. On the one hand, some of these papers represent (from today's perspective) important milestones or introduce smart ideas that might be worth teaching. On the other hand, these ideas might soon be considered outdated and other models might have taken their place in one year's time. So the question that has to be asked every semester is whether something can be considered "established" enough to enter the course or whether it is still too experimental or uncertain.

7 Conclusion

Throughout this paper, we shared some key takeaways and considerations when it comes to collaboratively developing OSER for NLP curricula. We highlighted crucial challenges that arise both due to the collaborative development and the different target groups and due to the peculiarities of the current fluid state of NLP research itself. Simultaneously we showcased the solutions we found, relying strongly on the OSER principles. We further view this paper as a means of advocating for more open source and more collaboration, also when it comes to teaching. While it may be common practice to collaborate in research itself or when it comes to using shared computational resources. Sharing and co-developing teaching materials, on the other hand, is far from being a commonly accepted best practice. We again argue, that one conclusion from the current speed of development is to join forces also for teaching. Finally, we share our material for re-use and inspiration and hope to attract other academics as future collaborators.

Limitations

While we do not claim that our course is all-encompassing or better than any other course, we still hope there is some value in (i) the course itself and (ii) the explanation of our thought processes. We think humbleness and the willingness to learn and improve one's teaching material continuously are key to the successful development of OSER. Everything we create and share happens to the best of our knowledge and we are always happy to be pointed at mistakes or inaccuracies so we can eradicate them.

Acknowledgements

MA is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of BERD@NFDI - grant number 460037581. This research was partially supported by DFG (grant SCHU 2246/14-1).

References

AI@Meta. 2024. Llama 3 model card.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Bernd Bischl, Ludwig Bothmann, Fabian Scheipl, Tobias Pielok, Lisa Wimmer, Yawei Li, Chris Kolb, Daniel Schalk, Heidi Seibold, Christoph Molnar, and Jakob Richter. 2022. Introduction to Machine Learning (I2ML). https://slds-lmu.github.io/i2ml/. [Online; accessed 2024-05-17].

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ludwig Bothmann, Sven Strickroth, Giuseppe Casalicchio, David Rügamer, Marius Lindauer, Fabian Scheipl, and Bernd Bischl. 2023. Developing open source educational resources for machine learning and data science. In *The Third Teaching Machine Learning and Artificial Intelligence Workshop*, pages 1–6. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

DeepLearning.AI. 2023. A complete guide to natural language processing. https://www.deeplearning.ai/resources/natural-language-processing/. [Online; accessed 2024-05-17].

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.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

fast.ai. 2020. Practical deep learning. https://course.fast.ai/.

Google. 2023. Introduction to machine learning. https://developers.google.com/machine-learning/crash-course/ml-intro. [Online; accessed 2024-05-17].

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Jeremy Howard and Sylvain Gugger. 2020. *Deep Learning for Coders with fastai and PyTorch*. O'Reilly Media.

Austin Huang, Suraj Subramanian, Jonathan Sum, Khalid Almubarak, and Stella Biderman. 2022. Tutorial 6: Transformers and multi-head attention. [Online; accessed 2024-05-17].

Hugging Face. 2022. The hugging face course, 2022. https://huggingface.co/course. [Online; accessed 2024-05-17].

Phillip Lippe. 2022. Tutorial 6: Transformers and multi-head attention. [Online; accessed 2024-05-17].

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ines Montani. 2019. Advanced NLP with spaCy: A free online course. https://github.com/explosion/spacy-course. [Online; accessed 2024-05-17].

Andrew Ng. 2021. Machine Learning. https://www.coursera.org/learn/machine-learning. [Online; accessed 2024-05-17].

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue.

OpenAI. 2023. Gpt-4 technical report. *arXiv*, pages 2303–08774.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Alexander Rush. 2018. The annotated transformer. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 52–60, Melbourne, Australia. Association for Computational Linguistics.

Stanford NLP Group. 2024. Cs224n: Natural language processing with deep learning (spring 2024). https://web.stanford.edu/class/cs224n/. [Online; accessed 2024-05-17].

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

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.