USING WEB-BASED NOTEBOOKS FOR BLENDED-LEARNING IN COMPUTER SCIENCE

Jens Albrecht
Technische Hochschule Nürnberg (GERMANY)

Abstract

The presentation discusses experiences with the use of web-based scientific notebook applications like Apache Zeppelin or Jupyter in the context of higher-level education. Notebooks created with these tools can contain a mixture of textual explanations, visualizations, and executable program code. Thus, besides being a platform for developing, documenting and sharing information on scientific problems, notebooks are also a great tool for education in computer and natural sciences as well as engineering. The notebook concept has been tested in lectures on databases and big data technologies. The prototypical teaching modules we developed show the great potential and some current limitations the notebooks.

Keywords: Blended Learning, Notebook, Apache Zeppelin, Jupyter.

1 INTRODUCTION

Today, e-learning plays a vital role in many businesses and educational institutions. A study conducted by Class Central in December 2016 states that over 700 universities are offering about 7,000 Massive Open Online Courses (MOOCs) with almost 60 Mio. registered students [1]. These course are a great possibility to get acquainted with new topics and are often used by individuals for continuing professional education. Many courses are excellently prepared by top-rated universities and show the great potential of interactive online trainings for higher education. Today, MOOCs even incorporate elements to of social collaboration among students [2]. However, MOOCs will not replace traditional teaching, because interaction with other students and instructors is still very important.

One approach to combine face-to-face teaching with e-learning and focus on their relative strengths is blended learning [3]. Even though almost all academic institutions have initiatives to support blended learning, the actual adoption on the course level is still rather low, because the use of new tools and the redesign of new course material is challenging and time consuming [4]. To provide an e-learning module, a lecturer has to do both: learn the functionalities of the e-learning platform and create resp. adapt the content to new concepts. Thus, the usability of the tool from the lecturer's perspective and its fitness for the specific topic of the lecture is critical for a wider application in university courses.

A common e-learning platform in many universities is Moodle [5]. Moodle is a mature learning management system and offers a large number of modules for different kinds of e-learning. However, several studies show that the usability of Moodle has a number of weaknesses (see [6] for an overview). While the system has been improved over the years, to set-up a good module and maintain it is still an intricate task requiring many mouse-clicks of the instructor. This maybe one reason, why e-learning modules still are missing in many lectures. In addition, e-learning tools for programming or data analysis require the possibility to write and execute program code in the required programming language. This is not supported by Moodle and similar platforms.

2 WEB-BASED SCIENTIFIC NOTEBOOKS

Independent from e-learning and instead motivated by an increased need to interactively work with data, a new kind of data analytics tools has emerged in the last years: web-based scientific notebooks [7]. The tasks of data scientists and analysts involve a mixture of scientific programming, data exploration, experimentation as well as the preparation of visualizations and the sharing of results. In the end it's all about developing a story to explain not only the final results but also the principles behind. This is very similar to what a lecturer has to do in class. Thus, web-based notebook tools like Jupyter and Apache Zeppelin which were designed for data science also have a great potential to support teaching ([8], [9]). In fact, many inspiring notebooks, primarily on Machine Learning and Python, have already been published and are available for use in academic courses ([10], [11]).
2.1 Notebook Interface

A notebook is a document consisting of a set of notes, similar to sections or paragraphs. Each note has a certain type: It can contain either executable code in one of many programming languages or some kind of documentation resp. visualization (see Fig. 1 for an example). The explanatory text is written in Markdown\(^1\), a language that can be used to create nice-looking html-output via simple plain-text commands. For example a line starting with “#” is converted to an H1-heading in html. Thus, writing formatted text in Markdown is similarly easy and fast as writing text in Word.

2.2 Popular Notebook Applications: Jupyter and Apache Zeppelin

The notebook principle was introduced by Mathematica and Maple in the 1980's and has evolved slowly over the last three decades. Today, many scientific computing platforms support some kind of a notebook interface. About 5 years ago the development of two new open source projects started: Jupyter\(^2\) and Zeppelin\(^3\). Both are web-based, i.e. the notebooks are created and running within a web-browser. Similar applications are now being developed by different vendors, e.g. Beaker\(^4\) or the Cloudera Data Science Workbench for Hadoop, but they are not yet as mature as Zeppelin and Jupyter [12].

---

\(^{1}\) [http://daringfireball.net/projects/markdown](http://daringfireball.net/projects/markdown)

\(^{2}\) See jupyter.org

\(^{3}\) See zeppelin.apache.org and zeppelin-project.org

\(^{4}\) See beakernotebook.com
2.2.1 Jupyter

Jupyter is an open source notebook application which was born out of IPython, a popular interactive shell for Python, in 2014 [13]. It supports about 50 different language backends, called kernels, including for example Python, R, C#, bash and even MATLAB\(^5\). Its current version is 4.1.

Although officially created in 2014, Jupyter shares a longer history with IPython. The standard configuration consists of a single-user web server and does not allow sharing of documents. An additional server, JupyterHub\(^6\), needs to be set up for multi-user environments ([14], [15]). The system seems to be more mature as Zeppelin, and additional modules for e-learning functionalities like grading are available [9].

2.2.2 Apache Zeppelin

Zeppelin development started about five years ago, and in 2016 Zeppelin became an Apache top level project. It is primarily used to interact with Apache Spark, a big data computation framework. Interpreters (similar to Jupyter Kernels) are responsible for the execution of notes. Today, there are over 20 interpreters available, primary for Big Data frameworks and query languages like Spark (Scala, Python), Flink, Hive, Cassandra, etc. But there is also a generic SQL-interpreter based on JDBC which can be used to execute statements on any database. In contrast to Jupyter, a notebook in zeppelin can contain notes of different interpreters, e.g. it is possible to mix for example Python, SQL and shell commands. The latest version is 0.8.

Zeppelin has built-in multi-user functionality. Notebooks can be private or public, personal or shared, and access rights control the privileges to read, modify and execute notebooks. However, this functionality is still immature as most of these functions have just recently been implemented. In addition, Zeppelin has a very open architecture that allows developers to add new interpreters and also extend the user interface relatively easy.

2.3 Notebook Applications for E-Learning

Notebooks can be a great tool for e-learning, but e-learning is not their primary purpose. Thus, it makes sense to have a look at their advantages and disadvantages with respect to teaching and self-learning.

2.3.1 Advantages of Notebooks for E-Learning

Notebooks are designed for documentation and reproducibility. That's why they make it very easy to create nicely-looking, well-structured interactive documents. Their main advantages are the following:

- **Ease of use**: Writing a teaching module to explain some kind of programming, a mathematical or a technical concept with interactive examples and exercises is as simple as preparing a word document for the same purpose.

- **Simplicity**: There are no complex elements in the user interface, just very few buttons. This helps both, the instructor and the students, to concentrate on the content. Especially programming exercises often require students to work with full-fledged integrated development environments (IDEs) which are completely oversized for initial learners.

- **Media-continuity**: Explanations, live code examples and programming exercises are all in the same tool.

- **Web-based**: The required infrastructure can be set-up on a central server. This eliminates the need to setup individual lab environments for the students.

- **Publication**: Notebooks can be published as static documents on the Internet, allowing students to search and review content at any time.

- **Sharing**: Notebooks can be shared, such that teams of students can work simultaneously on a document similar to in Google Docs or EtherPad.

\(^5\) For an overview check https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

\(^6\) https://github.com/jupyterhub/jupyterhub
2.3.2 Limitations of Notebooks for E-Learning

Notebook applications are not comparable to learning management systems (LMS) when it comes to features dealing with course organization and grading. In fact, neither Jupyter nor Zeppelin support these typical LMS functions:

- Management of courses and course related roles
- Course calendars and time-based visibility of notebooks
- Messaging and notifications
- Assessments including pre/post testing
- Tracking and reporting
- Multiple-choice tests and related methods

However, especially Jupyter seems to have a strong community among academic teachers ([8], [15]), and modules to allow grading of notebooks have recently been presented [9].

In addition, both notebook applications allow the creation of interactive dashboards by programming buttons, text input areas and so on. Jupyter provides the ipywidgets package (see [16] for a demo), while Zeppelin has its AngularJS-based display system [17]. However, adding buttons for interaction requires detailed programming knowledge. Moreover, the respective program code for the user controls currently cannot be completely hidden from the users, and that is giving room for cheating.

3 NOTEBOOK USE-CASES FOR E-LEARNING

Bloom's (revised) taxonomy [18] provides a well-established way to describe learning objectives (Fig. 2). Learning objectives in programming and data analytics generally target at least at level 3 (apply), but often up to levels 5 and 6 (evaluate and create). During lectures, students learn to understand the concepts and reach – in the best case – just level 2. All higher levels need extensive exercises where students can to write, analyze and evaluate programs themselves. Thus, notebooks should on the one hand help to reach higher levels more efficiently but increasing factors like motivation, engagement and fun. On the other hand, notebooks should also support the teacher and their preparation should not be an additional burden.

In the following, we will discuss three different scenarios for the use of notebooks: for self-learning, during supervised lab-exercises and during lectures.

3.1 Notebooks for Self-Learning

The combination of explanation, executable examples and interactive exercises make notebooks ideal for self-learning in the areas like programming, scientific simulation, data analysis or machine learning. Fig. 1 illustrates this mixture in a simple example. A complete set of excellent notebooks for a lecture on scientific Python was prepared and shared by Rober Johansson [11].

Notebooks are also well-suited to prepare flipped-classroom lessons [19]. Students can work through the simpler concepts on their own (Bloom level 1 and 2), the lectures can be used to discuss advanced concepts and solution approaches for more complex problems.

Figure 2. Bloom’s taxonomy. Figure 3. Setting permissions in Zeppelin.
3.2 Complementary use of Notebooks in Lectures

Programming lectures can be boring for students, because they have to learn the syntactical constructs of the language in theory before they are able to solve problems. Notebooks are a good option to complement frontal lectures on the basics of a language. Short theoretical presentations of 10-15 minutes can be mixed with similarly short exercises to support understanding and memorization of the concepts. However, there are two requirements to make this work:

1. Students must have access to a computer during lecture, which is often not the case. Mobile devices are not well suited, because writing a program – even a short one – requires a keyboard.
2. The notebook application must have the possibility to selectively give access to the exercises. If all exercises are accessible upfront, then students tend to work on exercises too early.

In Zeppelin, permissions can be set per note (Fig. 3). This feature can be used to control access to exercises during a lecture.

3.3 Notebooks in Supervised Lab-Exercises

Often lectures are accompanied by exercises in computer labs, where the instructor helps the students to solve complex problems. In this setting, notebooks do not introduce some new element to the lab, but still have certain advantages which can increase the learning efficiency.

The option to make the exercise tasks part of the same document as the solution, can simplify the understanding of the task for the students. Case studies or stories can be successively developed. Additionally, the simplicity of the notebook user interface can help students focus on the solution instead of the tool.

However, the simplicity of the tool can also be a disadvantage. For more complex problems in programming, language specific integrated development environments (IDEs) might be the better choice. Thus, notebooks seem to be well-suited to get started in programming, but after a few labs students might switch to specific tools, if they offer functions unavailable in the notebook application. Notebooks are generally better suited, whenever there is some focus on explanation and simple experiments.

4 EXPERIENCES AND STUDENT'S FEEDBACK

We experimentally used Apache Zeppelin for modules in an undergraduate class on databases (to learn SQL) and a graduate class on big data (to learn Hadoop and Spark). The feedback of the students suggests that the notebook lessons and exercises have deepened the understanding of the subject and increased motivation and engagement.

4.1 Notebooks in Database Lectures on SQL

In the database class we used complementary notebooks to mix short oral presentations with short exercises on the presented concepts. The exercises were very simple and mainly illustrated the operation of SQL statements. Their purpose was help students to remember and understand how they work as well as to keep the attention high.

Table 1 shows the feedback was gathered among the students after two lectures. The switch from presentation to hands-on exercises and back was perceived very positive. Additional free-text comments contained the following statements:

- It was very well perceived that after an initial presentation by the lecturer and some discussion it was immediately possible to reproduce the concepts on the system and experiment.
- The lecture was much more enjoyable, perceived engagement and attention was much higher.
- A minority of students would prefer to separate hands-on exercises from the lessons, because there is more thinking time for solutions.

From the lecturer's point of view it can be said, that hands-on exercises in a larger class are helpful, but they can be time-consuming, especially because each exercise needs some ramp-up time. In addition, it is not easily possible to support individual students if they are stuck on a problem.
Table 1. Student's feedback on the use of notebooks during a database lecture.

<table>
<thead>
<tr>
<th>No.</th>
<th>Question</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How do you rate the effect of the hands-on exercise on your general understanding of the presented topics?</td>
<td>Very Positive: 62%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Positive: 38%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral or negative: none</td>
</tr>
<tr>
<td>2</td>
<td>How do you rate the effect on your motivation and joy in learning?</td>
<td>Very Positive: 38%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Positive: 54%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral: 8%</td>
</tr>
<tr>
<td>3</td>
<td>How did you perceive the level of the exercises?</td>
<td>Little too hard: 8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Appropriate: 77%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Little too easy: 15%</td>
</tr>
<tr>
<td>4</td>
<td>How did you perceive the ration between lecture and hands-on exercises?</td>
<td>Not enough hands-on: 0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Well-balanced: 92%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Little too much exercise: 8%</td>
</tr>
</tbody>
</table>

4.2 Notebooks in Supervised Lab-Exercise on Big Data

In the master class we used notebooks in a supervised lab. The target was to understand the fundamental concepts of the distributed computing platforms Hadoop and Spark. Thus, the learning objectives were not to completely master a programming language, but to get some experience for deeper understanding of a complex topic in a short amount of time. In such a setting there is often the risk that the complexity of the tools or the assignment jeopardize the learning objective.

Notebooks provide the opportunity to create guided exercises that help to focus on the part of the solution that is really important. Textual explanations can be used to highlight and repeat aspects which were already discussed in the lecture. Prepared examples can serve as a template for the tasks.

5 SUMMARY AND OUTLOOK

Web-based scientific notebooks provide an interesting new platform for e-learning. The simplicity of the user interface makes it very easy for lecturers to prepare e-learning modules and for students to understand fundamental concepts by interactive examples and exercises. Notebooks can be best used if the tasks to be solved require much explanation, but little programming (few lines of code). Their ideal use case are relatively compact or guided hands-on exercises. Thus, they bridge the gap between theoretical understanding and practical solution of complex problems.

Both, Zeppelin and Jupyter support many different scientific programming languages – ranging from Python and R via SQL to Spark. Thus, they are especially well-suited for programming, data analytics and machine learning. However, these tools are fairly new, and features that are required to manage many users and assignments still need to be added or enhanced.

REFERENCES


