Online University - ICT for Education

"I am against education that is only available to the top 1% of all students. I am against tens of thousands of dollars of tuition expenses. I am against the imbalance that the present system brings to the world. I want to empower the 99%. I want to democratize education. Education should be free. Accessible for all, everywhere, and any time."

Sebastian Thrun – Artificial Intelligence Fall 2011 teacher, Google Fellow, ex-Stanford teacher

Background

A new revolution has started in the fall of 2011 on the Internet: 3 online interactive university courses were finished by thousands of students worldwide in Databases, Artificial Intelligence and Machine Learning.

After teaching the AI Class, Sebastian Thrun, a computer scientist known for working on self-driving cars at Google, said "I can't teach at Stanford again.", and gave up his tenure at Stanford to work full time on his new online university Udacity.

I finished all the 3 classes last fall. When I enrolled in the classes, I asked myself what could be so disrupting... I thought Khan Academy (a website by a teacher who recorded himself explaining Maths) or MIT OpenCourseWare (a YouTube channel, where different MIT courses are uploaded) to be the best resources on the Internet where to study a different topic. But none of these websites come near the interactivity of the online courses. Interactive quizzes, a big community waiting to help, homework deadlines, local study groups or the exam day, all of these were part of the atmosphere of online classes.

Idea

So, let's join the party. I want to build my own online university, to teach the whole world introduction in programming in Python, in a more fun way.

The first thing to do is to record the videos. I think that a webcam and a simple screen recording application are enough. The course will have 8-9 chapters, each with 15-16 videos of 1-2 minutes. Between certain videos, there will be a quiz, like the teacher asking the whole class a question, and the student will answer it.

After finishing the lesson, students will complete a smart review quiz, that contains questions from the entire lesson. The smart review quiz can be taken at any time, with a pause of an hour between two attempts.

In each lesson a student learns different "key concepts", and for each key concept a set of smart review questions will be created. For example, in the variables and constants lesson, a key concept is that constants are only assigned once. Different questions can be constructed, like lines from a program where a constant is assigned twice, or asking questions like "Should be defined as a variable or constant ?".

The exam will be personalized for each student, using more questions from "key concepts" where the student didn't perform well in the quizzes. This can ensure that the student understands the whole syllabus.

But the quizzes and exams shouldn't represent the whole part of the grade. There will be interesting and challenging programming challenges, one for each chapter. One example can be a semi-completed app, that comes with the Facebook API integrated. Then, the students would have to complete useful tasks, like counting the number of pictures of each friend, or printing the most addicted friend to Facebook. Other useful projects include coding spellchecker, a crawler or using the Twitter API. The apps then will be graded, by testing them on different pre-run cases, and verifying if they show the correct answer.

But a website isn't complete without a community. I think that a little gamification, like badges and a leaderboard could be enough to make students help each other. Another thing I can think about are series of videos "Ask the teacher", where the most active students in the forum could ask questions, and the teacher answer them in videos.

Another aspect of the community is the contribution for subtitles. Using open crowdsourcing technologies like Universal Subtitles can ensure that the videos can be available in a lot of languages.

The most important thing I think the other courses missed is the social network impact. People love to share what they do on Twitter or Facebook. Why not post on their wall that they helped someone with their homework. Or why not make a special page with the students who had perfect scores in a week. On the other part, you can send the lazy students a tweet to remember them to send the homework. I think that small details, like sending an email with encouragements before an exam, can make a big difference.

After the course ends, each student gets his enrollment proof, together with a full syllabus, where all the "key concepts" are graded. The student then will know where to concentrate to develop his skills better.

The teacher could also use the information gathered from the students to see what wasn't understood and record again a video, or change the questions of the quizzes.

Technologies involved

I think that the website will be built using only open source technologies. On the server side, I think that any popular framework can be used. I would tend to use Python Django, because it could be easier for those who finish the course and want to contribute.

On the client side, I think approaching a Responsive Design should be the focus of the user interface. Students want to see the lessons on desktops and laptops, but also on tablets and mobile phones.

Scalability and availability should also be kept in mind when choosing where to host the content. For videos, YouTube is the best choice. For the website, we can use cloud technologies like Heroku or Amazon Web Services. People don't want the website to go down in the last hours of the exam. But these services aren't cheap.

Money could be a problem when beginning the project. The first thing to do is looking for sponsor companies. If the course is say about web design, a website like Freelancer.com would pay for sending offers to the top students to register on their website. Another option could be partnering with a university. The university can send enrollment offers to the brightest student. This partnership will also be beneficial, because the university teachers could create their own courses for the website.

The future

After the introduction to programming, more advanced courses will begin, like algorithms, software and web design or mobile development. But computer science isn't the only domain, there can be courses on biology, chemistry, geography, economy, startups, all you can imagine.

"Typically I teach around 100 students per year in my introductory database course. This past fall my enrollment was a whopping 60,000. Admittedly, only 25,000 of them chose to submit assignments, and a mere 6500 achieved a strong final score. But even with 6500 students, I more than quadrupled the total number of students I've taught in my entire 18-year academic career."

Jennifer Widom - Online Introduction to Databases Fall 2011 teacher and Stanford teacher