Cognitive self-tutorial service

Patrick Nicolas V 0.14 June 11, 2018

	When	What
0.10	04.19.18	Initial draft. Introduction of the personalized self-tutoring and course workflow
0.11	04.27.18	Added the concepts of expertise maps, leaderboard. Added list of possible technologies
0.12	06.02.18	Refined problem statement, added strengths/weaknesses targeted students, TDD-based training
0.13	06.07.18	Added market positioning, analysis and elements of cognitive load
0.14	06.11.18	Added technologies section, abstract, details on interactive learning

Table of contents

Abstract

This document lays out a plan and service to address the struggle of aging engineers to stay current with the latest innovation in engineering.
We look at
1. The challenges of learning when getting older: abstraction and fading working memory as gathered through research papers
2. The definition and impact of cognitive load
3. Combination of interactivity and visualization as a solution to overcome learning limitations related to age
4. A typical user scenario
5. Potential techniques to support interactive and visual training environment.

Project & Plan

Requirements

To be viable the project should meet if not all, at least most of the following criteria:
- **Problem statement**: Service or product needs to solve a legitimate and clearly defined and understood
- **MVP** (Minimum Viable Product): The initial incarnation of the service or product should require a minimum effort [Incremental value]
- **Personal Development**: The project would include enough new technology for anyone joining the project to learn from [Personal dev.]
- **Exit strategy**: The project should have a target and deliverable beside the product and services: Would it be open sourced? Will the technology license? Do we plan a patent? Is there a possibility to start a company around the service?

Note: AI-based implementation may be way more appealing for investors

Feedback 0.11 & 0.12

The feedback on version 0.11 (05.30.2018):
- *Expertise maps*: It requires more work as even junior developer may list many skills set which may not reflect the actual experience. One option is to generate the expertise map after the interviews.
- *Personalized self-tutoring*: It may require a ‘adaptive assessment’ and result of psychology studies.
- *Features space*: It may need to rely on ‘Cognitive Dissonance’ to create a feature set that is appropriate for the targeted group of users.

What’s next

- Version 0.15 (target June 16)
- First estimate for POCs
- Update the list of references
- Get feedback from 1 or 2 potential users

Analysis

Problem statement

There is a significant human and financial cost associated with lack of appropriate training for the aging population of engineers.
Engineers must fight the perception of irrelevancy in a world that favor youth and wit. They also struggle to find adequate training methods.
Employers miss on the opportunity to leverage the experience of seasoned developers. They must deal with the perception and financial liability of age discrimination.

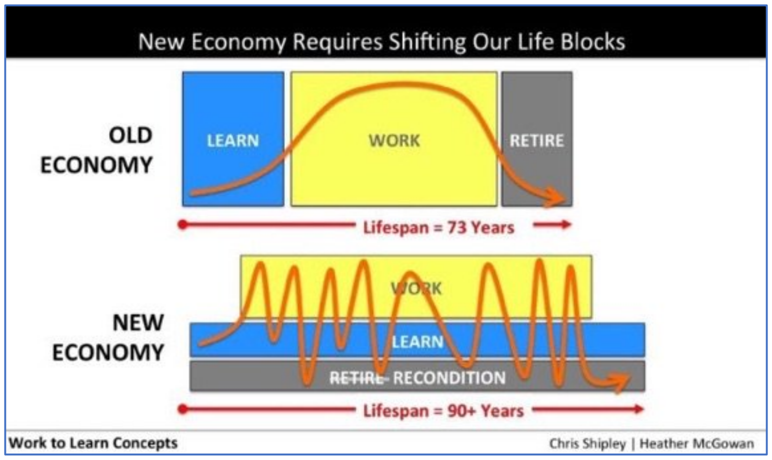
Current solutions

Any training program should leverage the strengths and ability specific to older engineers while minimizing the reliance on skills that degrade with age.
The problem statement can be summarized as:

There is no known tutorial service that caters to aging software engineers

Research

A new key concept that has been emerging for the last 5 years: *Computational Thinking* [*Review of Computational Thinking Assessment in Higher Education*, 1]. This concept is not completely new and offspring of the idea of continuous learning. In the new economy, learning new concept is a lifetime endeavor.



In the interaction between the student and agent/teacher, the questions are categorized as

- *Definitional questions*: Questions on the core concept (i.e. How do I compute the maximum likelihood in Naïve Bayes)
- *Tracing questions*: Understanding of execution and prediction of the output (i.e. Is an iterator be more appropriate to collect output?)
- *Code completion questions*: Understanding of expected output (positive, and errors) and discrepancy between actual and expected results (i.e. Why the search produced a different result?).

There are many studies that highlight the strengths and short comings age bring to engineers [*Normal Cognitive Aging:* 2], [*Reducing Cognitive Load in Learning Computer Programming*: 3], [*Educational attainment and cognitive decline in old age*: 4].

Moreover, it has been found that *concrete thinkers* are more proficient in visual and OOP while *logical thinkers* are doing better in procedural programming and pure problem solving (puzzles). These findings re-inforce the notion that aging engineers are more receptive to learn through visual and object representation.
[*A Theory of the Relationships between Cognitive Requirements of Computer Programming Languages and Programmers' Cognitive Characteristics*: 5]

Finally, here is a summary of the psychometric research over the last 20 years that applies to learners of any age.

- Excess information reduce ability to learn
- Low self-esteem and stress full environment hinder learning
- Learning around ½ time through errors is a good balance
- Brain needs novelty within a course (change of pace)
- No research has been able to isolate a single, more effective learning style
- Brain needs to practice (interact) with a topic or lose the newly acquired information
- Sharing with colleagues increase learning rate (sense of accountability, reduce fear of mistake)
- It has been proven that learning change the brain structure. The more you learn in a positive and fostering environment the faster you learn.

[*Biggest breakthroughs in the science of learning*: 6]

Here is a summary regarding the positive and negative impact of aging in cognitive learning.
Engineers preserve or even improve upon the following set of skills as their age

1. Good understanding of "big picture" (i.e. every stage of product development from requirements to delivery)
2. Ability to handle increasing complexity
3. Display strong semantic memory (practical knowledge, meaning, concrete example)
4. Good retention of information
5. Ability to conduct technical discussion and brainstorming sessions
6. Strong visuospatial abilities (Object perception, pattern detection)

The following abilities decline over time

- Focus on narrow problem and programming tasks that require *working memory*
- Processing speed
- Episodic memory (personal experience, events)
- Rate of acquisition of new information, subjects
- On the spot retrieval of information
- Visual construction skills

Cognitive load

Let's consider the different type of learning, ranked in decreasing order of abstraction and duration:

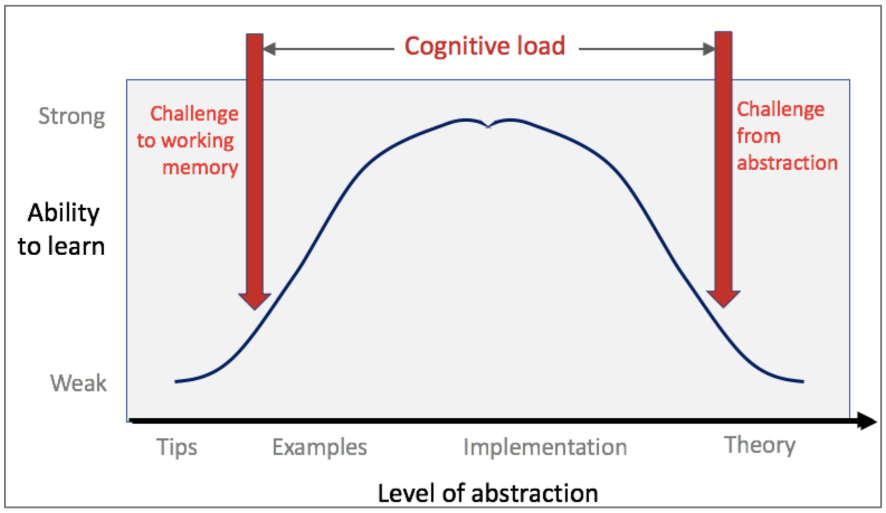
- Theory
- Implementation
- Practical, concrete example

- Tips

As discussed in the previous paragraph, aging engineers struggle with two types of mental activity:

- Learning new abstraction: The brain creates problem solving paths over the years. It impedes the ability to “absorb” new concepts (pathways in the brain)
- Retrieving and acquiring detailed information about a topic (small chunk of knowledge, DOs/DON’Ts, tips, …) that taxes the working memory.

Simplistically, we can define *cognitive load* for the aging population as *difficulty to learn new abstraction and detailed information.*



Solution

Interactive and visual learning

How can you reduce the cognitive load?

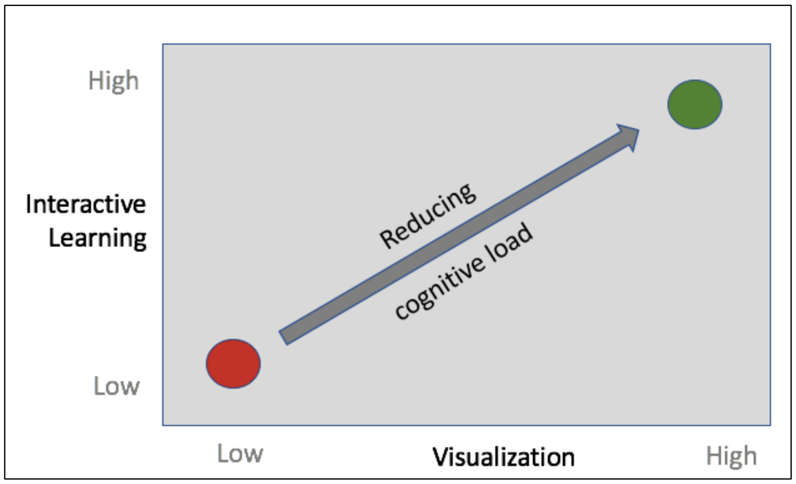
- The difficulty of *learning abstractions* can be alleviated by using graphical representations, or metaphors. The visualization of an abstract concept facilitates the memory retention
- *Frequent, interactive exercises* reduce the time to learn about detail information about a topic.

The following selection criteria are used to characterize the solution

- *Scope of learning*: What type of learning is provided, from theory, foundations, concrete examples, problem solving to tips for interview or troubleshooting?
- *Interactivity*: How frequent and relevant is the feedback provided to the student?
- *Cognitive load*: What is the load on the working memory and need for processing speed from the student?

An interactive learning process reduces the cognitive load and reliance on short term working memory. Visualization facilitates the understanding of new abstract concepts.

The service will gradually add capabilities to reduce cognitive load.



Scope of solution

Most of studies on the cognitive ability of aging people discuss ideas on how to remedy to their learning difficulties. The proposed solutions require a deep knowledge in cognitive learning not expected available to the contributor to this project.
This project introduces techniques that leverage the known, existing strengths of aging engineers not address their weaknesses.

Market positioning

I identified 5 different types of services, products or material which goal is to enhance the skills/knowledge of software engineers.

1. Preparation to interview (i.e. LeetCode):
2. Coding training (i.e. Hack reactor)
3. Formal topic centric training (i.e. Coursera)
4. Cookbooks (i.e. Apache Kafka cookbook)
5. College curriculum (i.e. Undergrad, Bachelor degrees, …)

User scenario

A typical course would contain:

1. Introduction to the subject: (topic, concept, algorithm): This is more as a reference to be used during the practical exercise than a formal training.
2. Presentation of the use case or requirement): We could/should allow the student to select a theme or related set of use cases that can be used across all the exercises (i.e. health care, advertising, IT, ….)
3. The service generates (or retrieves) the test code (what to expect) and/or a high-level template (interface) for the implementation of the use case (Test Driven Development): *The student should be able to select the language.*
4. The service generates a graphical representation of the test code or template.
5. The service generates and select several incomplete implementations with increased degree of complexity. The tutorial includes
    - Visualization of the source code

    - Simulation of the execution path.

   Each implementation will refer directly to a section in the introduction of the topic.

6. The service generated implementation with errors to be discovered by the student. *The objective is to add some interactivity with the subject.*
7. Finally, the service generates a summary of the topic, and evaluate the most appropriate next step subject that include some remedial tutorial.

Notes:

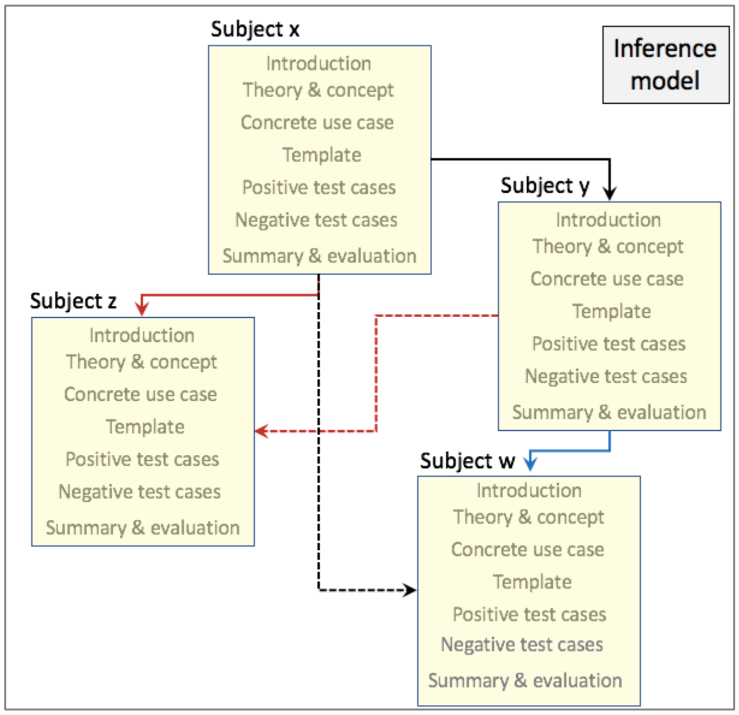
- The type of implementation (positive case) and error (negative case) is customized to the level of the student. Some incomplete tasks will be revisited later. The idea is to preserve the momentum in the learning process by setting aside a difficult task and came back to it later.
- The progress (reward) of the student toward the goal should be visualized at each step (Gamification)

Service definition

Curriculum

The curriculum or the course is represented as a graph or a sequence of learning exercise or topics. The course flow is tailored to each student depending on his/her

- Existing knowledge of the subject. At some point, a semi-automated initial assessment of the student ability need to be defined in order the customize the curriculum for a given student from the start.
- The pace and ability to complete the previous exercises.

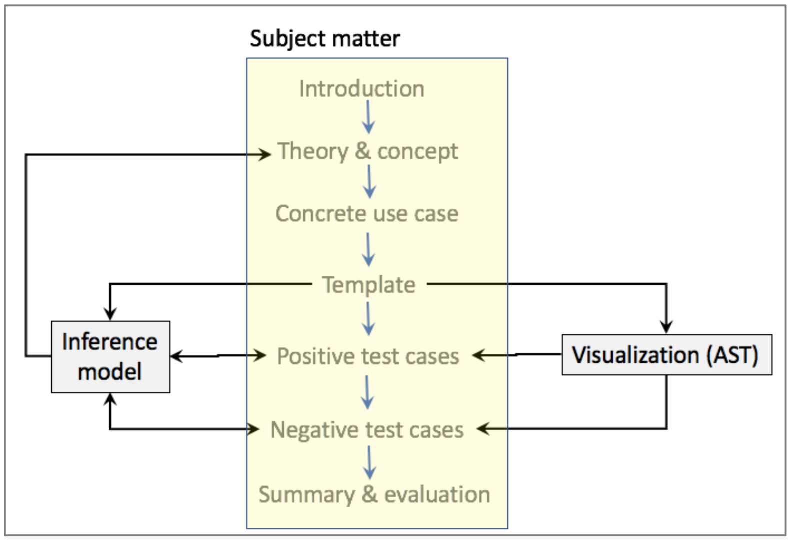


Tutorial

Some of the tasks should be manually implemented or semi-automated before leveraging AI. There is no need to start significant development before getting some level of validation. The most critical or/and uncertain tasks may require a proof of concept.

Here is a partial list of key components of the service

- Definition of an easy to understand "concrete" user scenario for a set of exercises
- Mechanism to either associate or generate test code in student favorite language.
- Mechanism to generate an AST from test code. The AST is represented graphically
- Generation of AST for different solutions to the problem with increasing complexity. *Each solution is illustrated as a visualization of an AST*
- Generation of the implementation code from each AST selected by the student
- Definition of the features set of a topic/coding exercises. The features set is used to 1- customize the different solution for a given problem/use case, 2- evaluate the progress of the student and select the next, most appropriate subject and level of difficulty for the exercise.
- Define the feature set for quantifying the ability, expertise of the student
- Create a model to suggest an implementation according to the level/ability of the student
- Design an active learning or RL to transition the student from one topic to another topic, depending on the topic attribute and the student expertise.
- User interface (Mobile? Web…)



Notes

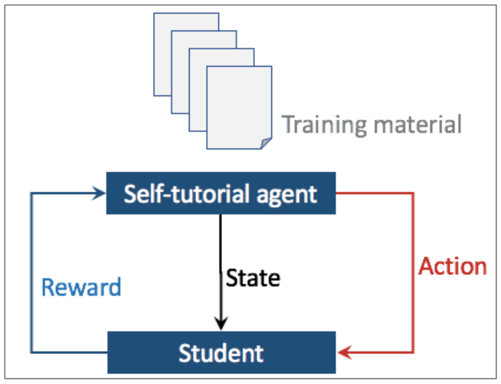
- The tasks should be manually implemented or semi-automated before leveraging AI
- A proof of concept for the most critical tasks should be done prior of a formal implementation
- Models requiring training are created using simulation
- An important element is to leverage the existing student skills (i.e. Machine learning for Java developers)

Technology

Various technologies are introduced to implement interactivity between the student and the training material. [*A tutorial on machine learning in educational science* 7], [*The benefits and limitations of machine learning in education,* 8]

Deep reinforcement learning (rewards)

This family of machine learning techniques relies on a state (knowledge of the student), action (next topic or training exercise) and reward (progress).



There have been quite a few studies in the field of applying RL to education, including learning how to play a game. [*Playing Atari with Deep Reinforcement Learning* 9], [*Better Education Through Improved Reinforcement Learning,* 10].

Active learning (discovery)

This technique selects the most appropriate exercise according to the characteristics of the student. The algorithm attempts to increase the knowledge about the ability of the student by proposing exercises, examples and topics which highlight the most unknown skills set.
This is about exploring the state of the knowledge of the student in each field. [*A tutorial on active learning* 11]

Conversational AI (student bot)

This sophisticated approach relies on a sequence questions/answers to detect short comings of the student. The key ingredient is the task-oriented dialog agents which are derived from the work done for customer support agent. [*Conversational AI: What to expect for the next 5 years,* 12]
The most commonly used AI techniques are the Recurrent Neural Network (RNN), Gated Graph Neural Networks (GGNN), Long-Short Term Memory (LSTM) and bi-directional attention networks.
[Automated Curriculum Learning for Neural Networks 13], [*ScreenerNet: Learning Self-Paced Curriculum for Deep Neural Networks,* 14]
There are many more different AI-based techniques that can be applied to smooth learning curve [*Learning to teach*, 15]

Proof-of-concepts (POC)

The purpose of POCs is to evaluate the technical feasibility and the user response to key components of the service. The POCs cover:

- Mockup for visualization/Metaphor for representing code and concepts
- Generation of graphical representation of concepts
- Example and validation of the criteria of difficulty of exercises
- Validation of the 3 types of questions (Definitional, Tracing, Completion)

Development requirements

What are the minimum set up for this project?

- Minimum set of development procedures
- GitHub account

- AWS deployment environment
- Communication channel

Summary

Does this product meet the 4 requirements defined in the introduction?

Problem statement	Existing learning tools and process are not adequate for experienced and older engineers.
MVP	TBD
Personal dev.	Reinforcement learning, adaptive learning, modeling and graph theory will be among the technology that will be evaluated as part of the project. It should be enough for anyone to learn algorithms and techniques that can be valuable in their day-to-day job.
Exit strategy	1. MVP service available to anyone for free 2. ??

Appendix

References

[1] *Review of Computational Thinking Assessment in Higher Education* D. Sondakl
https://arxiv.org/pdf/1703.07659.pdf

[2] *Normal Cognitive Aging* C. Herada, M Natelson, K. Triebel
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4015335/

[3] *Reducing Cognitive Load in Learning Computer Programming* M. Yousoof, M. Sapiyan, K. Kamaluddin
https://mafiadoc.com/reducing-cognitive-load-in-learning-computer-programming_59e28bfe1723dd3a8d0a6e6f.html

[4] *Educational attainment and cognitive decline in old age* R. Wilson, L. Hebert, P. Scherr, L. Barnes, C. Mendes de Leon, D. Evans
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2677531/

[5] *A Theory of the Relationships between Cognitive Requirements of Computer Programming Languages and Programmers' Cognitive Characteristics* G. White
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.4257&rep=rep1&type=pdf

[6] *Biggest Breakthroughs in the science of learning* A. Moritz-Saladino
https://www.brainscape.com/blog/2012/10/breakthroughs-science-of-learning-2/

[7] *A tutorial on machine learning in educational science* – L. Kidzinski, M. Giannakos, D. Sampson, P. Dillenbourg

[8] *The benefits and limitations of machine learning in education –* W. McGuinness
http://www.gettingsmart.com/2018/02/the-benefits-and-the-limitations-of-machine-learning-in-education/

[9] *Playing Atari with Deep Reinforcement Learning* – V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller
https://arxiv.org/abs/1312.5602

[10] *Better Education Through Improved Reinforcement Learning* – T.S. Mandel
https://digital.lib.washington.edu/researchworks/bitstream/handle/1773/40543/Mandel_washington_0250E_17666.pdf

[11] *A tutorial on active learning* S. Dasguta, J. Langford
http://hunch.net/%7Eactive_learning/active_learning_icml09.pdf

[12] *Conversational AI: What to expect for the next 5 years* – Vu Ha
https://chatbotsmagazine.com/conversational-ai-what-to-expect-in-the-next-five-years-26518c8cd385

[13] *Automated Curriculum Learning for Neural Networks* - A. Graces, M. Bellemare, J. Menich, R. Munos, K. Kavukcuoglu
https://arxiv.org/pdf/1704.03003.pdf

[14] *ScreenerNet: Learning Self-Paced Curriculum for Deep Neural Networks* – T.H. Kim, J. Choi
https://arxiv.org/pdf/1801.00904.pdf

[15] *Learning to teach* – Y. Fan, F. Tian, X.Y. Li, T.Y Liu
https://openreview.net/pdf?id=HJewuJWCZ