

# Enhancement of an Educational Interactive Story Generator for Apprentices

Tim Nguyen

Supervisors: Catharine Oertel, Pierre Dillenbourg

École Polytechnique Fédérale de Lausanne, Switzerland, June 2019

## Abstract

In this project based on a past paper<sup>[1]</sup>, we are working on an educational platform where the user is immersed in a story and meet different fictional characters. Depending on the choices the player has decided to take will derive different scenarios. The aim is to improve this existing product in order to enhance the experience of young students when learning how to interact with each other or with other external persons in the context of their apprenticeship, and also to upgrade the project itself in order to facilitate the generation of such experiences.

Until now, the paths were arbitrarily chosen given some input data. One of the specific goals is to define different algorithms to automatically choose the outcome of a scene after making a decision. The first algorithm was designed by a member of the CHILI laboratory<sup>[2]</sup>, while the second was based on Bayesian Knowledge Tracing<sup>[3]</sup>. These algorithms were designed such that their objective is to select an appropriate scene that would train a skill that an apprentice needs to practice, depending on his/her previous choices. In addition, since the platform can adapt to any possible situations and contexts given the inputs, the other goal is to implement a node-based interface visualization to ease the generation of these experiences.

This paper relates a step-by-step process of all the different features that were added to such a project using Unity<sup>[3]</sup>. The project is available on GitHub<sup>[4]</sup> for further improvements.

# 1 Introduction

In a previous project, we have created an educative platform that allows apprentices to train their social interactions with non-player characters through an interactive story, which may help to improve their self-confidence and social experience over time. For young students, being successful in an apprenticeship and in the future by building a prosperous career might be facilitated by having a large network of peers. The essential goals were to build a first functional prototype of this platform to generate such stories. It appeared that its implementation can be greatly improved with many features.

According to those improvements, we have considered the main aspects this project has, which are the ability to adapt to any type of experience one is willing to build, so any apprentice coming from any environment can exercise her or his social skills using this platform, and the educational nature of this immersive adventure through a playful activity.

The first aspect finds its importance in our will to make this product available for every educational domain. Inspired by some speech-based interfaces for task-oriented dialogues, a visualization of the dialogues was implemented to facilitate the creation of these experiences.

Then, one of the ways to arouse the interest of the users and to encourage them to pursue their training is to make the experience more immersive for them. Hence, we have tested and implemented in this project two different algorithms, that are both based on the skills of the user, to choose the outcome of a situation depending on her or his performance.

## 2 Aims

### 2.1 General goals

More specifically, the main challenges will be:

- To improve the Story Generator so that any user who would like to create a customized experience can make it conveniently, using a **node-based visualization** of the dialogues,
- To implement **skill-based algorithms** to automatically select the best scenario for the users to adapt to their own personal skills and experience: the first algorithm was designed by a member of the CHILI laboratory<sup>[2]</sup>, while the second implements the Bayesian Knowledge Tracing<sup>[3]</sup>.

### 2.2 Description of the demo

As an example, this process of building such a project has been made with respect to a *hospital* context. The apprentices will be nurses who want to practice their social interaction with their patients and colleagues. All the resources shown in this paper were based on this idea, but the project can adapt to any possible context or environment.

### 3 The project

In this section, we will give an overview of the project before adding the enhancement features and describe its current state. A more detailed description can be found on the original paper<sup>[1]</sup>. We will also briefly approach the minor changes the project has received before implementing the main features.

#### 3.1 Overview of the original project

The project we have worked on was greatly inspired by story-based choice and consequence games that recently appeared in this industry. What makes this type of entertainment so popular is the ability to try different scenarios and see their outcomes, which is not possible in real life. The user is immersed in an adventure where she or he is master of what happens throughout the story. Combining with an educative purpose, the players will be able to practice their social interactions skill while enjoying testing different scenarios available in the experience.

A classic interface for such a game is a background representing the environment where the story takes place, a dialogue box containing the narrative texts, and a sprite of a character with whom the user interacts (Figure 1). Generally, the name of the character who is currently talking is displayed above the dialogue, and the story is told from the player's point of view (first person).

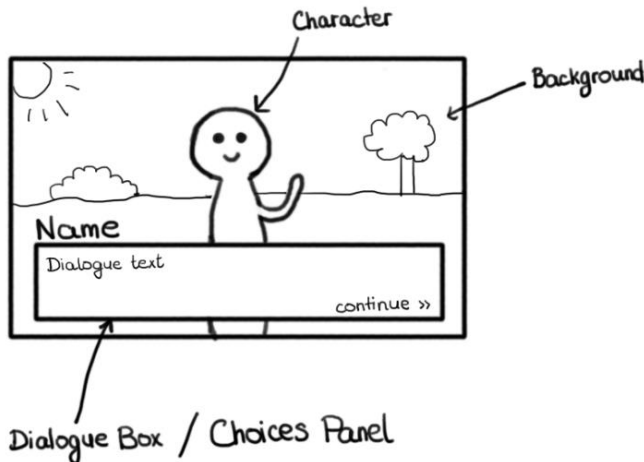


Figure 1: The prototype interface

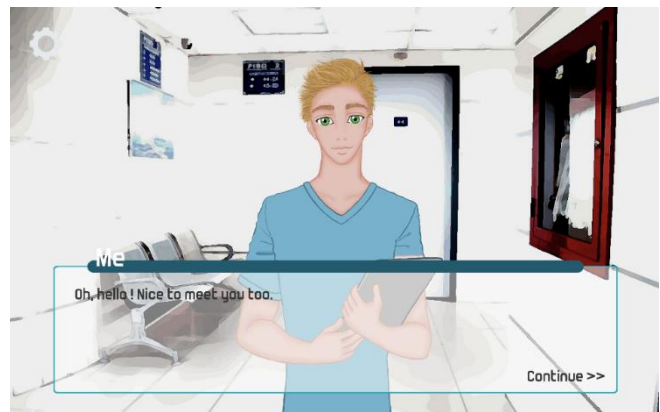


Figure 2: The resulting interface

The main goal of this project was to build a product that, given a set of resources, would generate any type of interactive experience to allow the apprentices to train their social skills. At the end of this previous project, we were able to have a functional experience to show as a demo<sup>[5]</sup>.

### 3.2 Current state of the project and first enhancements

The platform was created using a system of Managers, where each Manager is responsible of a specific task. We have chosen to use this structure because of its modularity. Each Manager has to perform of a particular task and communicates with the main Manager to access to a part of the resources of the game. Thus, to add a new functionality to the experience, we can just add a new Manager with its specific functions and modify the main Manager to have access to the desired resources.

Initially, the structure of the project is pictured in Figure 3. We have first modified this hierarchy to give the main role to a new Manager, the Story Manager. Basically, this Manager will be the only one who has access to the Resources Table (instead of the Dialogue Manager) and its main role is to handle the general flow of the game and to ensure that all the other Managers have access only to the resources they need. By doing so, the Dialogue Manager has now a clear role, which is to handle at which moment the dialogues are correctly displayed.

The new structure of the project for this report is described in Figure 4.

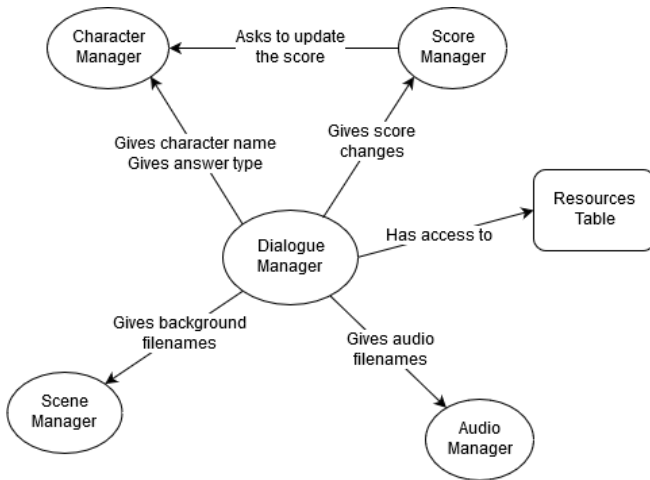


Figure 3: The original structure

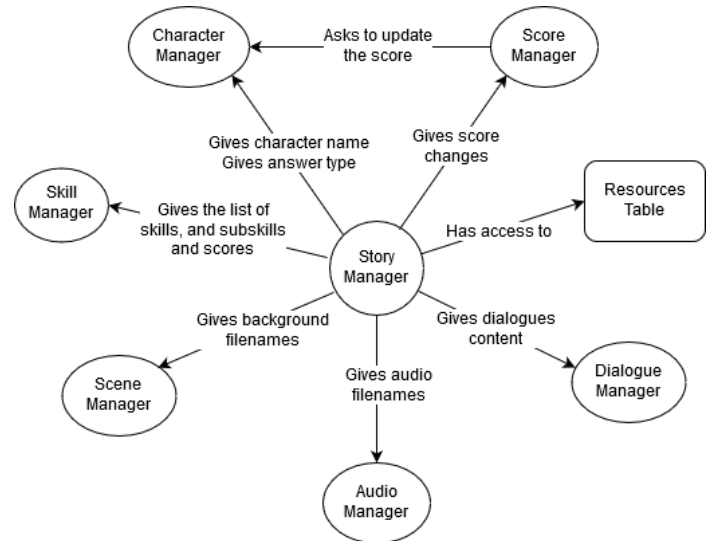


Figure 4: The enhanced structure

The Skill Manager is a new Manager we have implemented. It will contain the next part of this project, which focused on implementing an automatic skill-based scene selector, as one of the improvements we have applied on this platform. We will describe this Manager in more details later (see Section 5).

## 4 Node-Based Dialogues Generator

We will first describe the improvement made on the story generation side of this project, which is a visual interface that can generate the resources needed on our platform, the Story Generator.

### 4.1 The Issue

Initially, the Story Generator takes as input a CSV file, where all the information about the story is gathered in **one unique file**. Each row of the CSV file describes one dialogue extract of one character. Then, each column gives information about that dialogue extract, as for instance the name of the character who is saying this extract, the scene where the dialogue is taking place, the name of the background that should be displayed at this moment, and so on.

It appeared that using this kind of structure was more challenging and not as user-friendly as expected. Hence, the first goal of this project was to implement another interface for the users so that any person who wishes to generate a story using our product would be able to perform it without struggling with the handling structure of the data.

An intuitive idea was to design an interface to visualize the dialogues using a *Node-Based System*. An idea of this visualization is shown in Figure 6.

sceneID	character	dialogue	answer_good	answer_bad	...	next_good	next_bad	...	background_img
intro_0	Me	(Oh, okay?)	NA	NA	...	NA	NA	...	reception
intro_0	Doctor	I'm very pleased to have you here. How do you feel?	Well, I feel great, but...	I've never wanted to be here! Let me leave!	...	intro_0A	intro_0B	...	reception
intro_0A	Doctor	This is wonderful! I hope you're not too afraid of what will happen next.	NA	NA	...	intro_1	intro_1	...	reception
intro_0B	Doctor	Oh, don't leave yet, the fun hasn't began yet!	NA	NA	...	intro_1	intro_1	...	reception
intro_0C	Doctor	I know, I know, you will understand a bit more later.	NA	NA	...	intro_1	intro_1	...	reception
intro_1	Me	(What does he means? Now I'm afraid!!)	NA	NA	...	NA	NA	...	reception

Figure 5: The original data structure

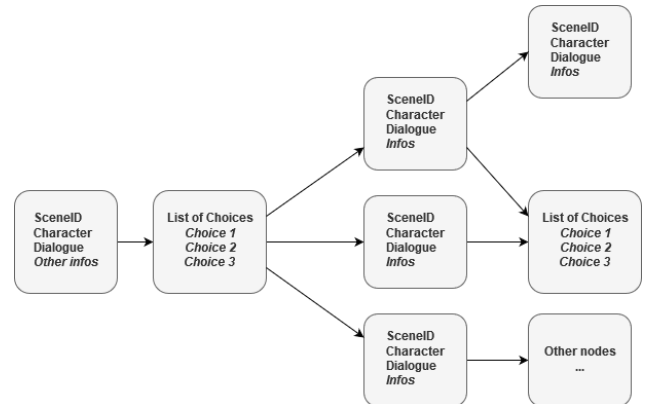


Figure 6: An idea of visualization

## 4.2 Implementation

### 4.2.1 Tools & Resources

To implement such an interface that would generate automatically the CSV file with the right structure, we have used a JavaScript framework called **Rete.js**<sup>[6]</sup>. It is a modular framework for visual programming that allows to create a *Node-Based* editor directly in the browser. One of its main advantages is that it is flexible, meaning that we can create and design any component with any features we want, and it is also universal: the framework is not bound to any domain, it allows us to implement an interface that can be open in any browser, and generate easily our dialogues.

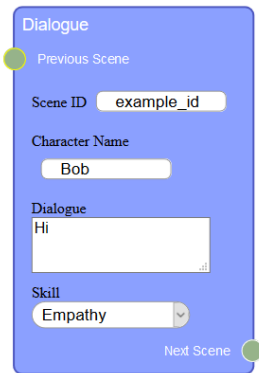
### 4.2.2 Components

We first started with a “basic” component so that all the others can derive from this model. A component can be added, dragged, selected, removed and connected to other components. The interface can be zoomed in and out to have an overview of the dialogues and also dragged as well. We have added a button to generate the dialogues from the editor into a CSV file, that has the valid shape to be given as input in our Story Generator.

To build our editor, we thought of two main components to implement:

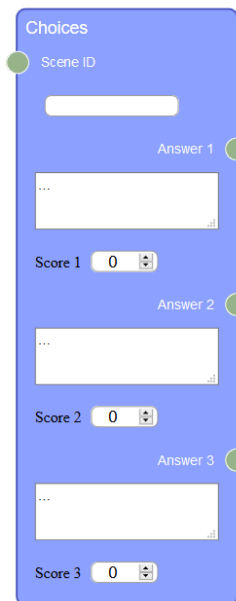
- **Dialogue Component:** This is the main component of the editor. It will contain all the necessary information that is needed in the CSV file. For the moment, it has a few fields that define a dialogue extract, like the sceneID, the character name, the dialogue content and a select box in order to choose the skill to which the current dialogue belongs to. Currently, we can choose between two skills: Task and Empathy.
- **Choices Component:** This is the component that will be used to define the possible choices in the experience and that will allow to separate the story into multiple paths.

Two additional components were added in order to facilitate the implementation of the dialogues generation, but also to improve the visualization: the Start and End components. They are used to define the very beginning of a story and its end. Since multiple scenarios can derive and not all paths have the same length, it was necessary to indicate when a path is ending by using these components.



The Dialogue component is a blue rounded rectangle. It has a green dot on the top-left corner and a green dot on the bottom-right corner. Inside, it contains the following fields: 'Previous Scene' (a small text label), 'Scene ID' (a text input field with 'example\_id' entered), 'Character Name' (a text input field with 'Bob' entered), 'Dialogue' (a text area with 'Hi' entered), and 'Skill' (a dropdown menu with 'Empathy' selected). At the bottom, there is a 'Next Scene' label.

Figure 7: The Dialogue component



The Choices component is a blue rounded rectangle. It has a green dot on the top-left corner and three green dots on the right side. Inside, it contains the following fields: 'Scene ID' (a text input field), 'Answer 1' (a text input field), 'Score 1' (a numeric input field with '0' entered), 'Answer 2' (a text input field), 'Score 2' (a numeric input field with '0' entered), 'Answer 3' (a text input field), and 'Score 3' (a numeric input field with '0' entered).

Figure 8: The Choices component



The Start and End components are two rounded rectangles. The Start component is blue and labeled 'Start' and 'First Scene'. The End component is yellow and labeled 'End' and 'Last Scene(s)'. Both have a green dot on their right side.

Figure 9: The Start and End component

### 4.2.3 Results

By combining all those components together, we can build a multi-paths story that will generate our data into the desired CSV file.

An example of the resulting visualization is shown on the Figure 10 below.

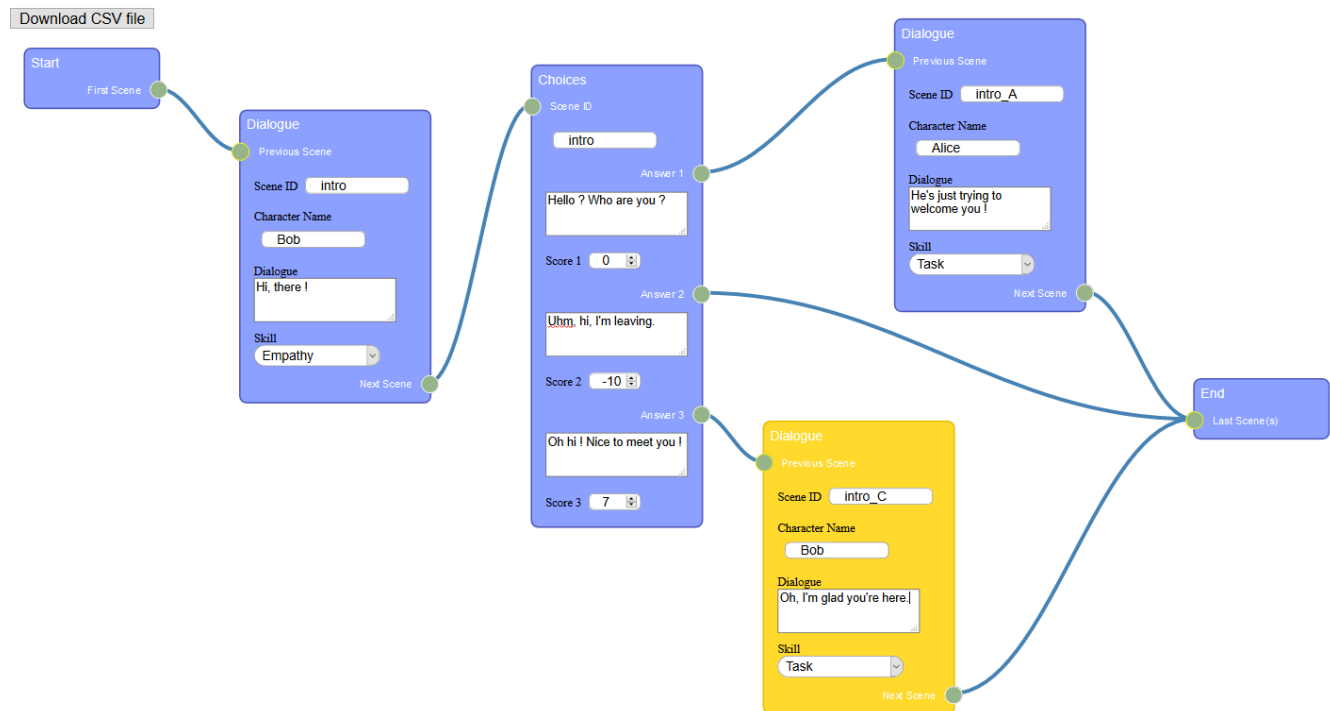


Figure 10: Example of a resulting visualization

### 4.2.4 Future works

The implementation can always be improved, either to add more features to a dialogue extract, or to facilitate even more the integration of the data. Ideas of future works can for example be to add an uploading system in the Dialogue Editor, to allow to upload directly the images, backgrounds and audio files, and then to generate a ZIP file that would contain not only the CSV data but also the folders of the media files.

# 5 Skill-Based Probabilistic Algorithms

## 5.1 The idea

Until now, the scenarios to which the story derive were defined in the CSV file itself: we had for each row of our data some columns that specify the IDs of the scene to jump for a given answer. For a more immersive and adaptive experience, we wanted to implement a skill-based algorithm that would consider and prioritize the scenes that evaluate a certain skill for a user.

Hence, we have studied two algorithms for this purpose: the *Probabilistic Cases Algorithm*<sup>[2]</sup> and the *Bayesian Knowledge Tracing Algorithm*<sup>[7]</sup>, whose parameters have been improved by a learning algorithm using a *Knowledge Heuristic and Empirical Probabilities*<sup>[8]</sup>.

## 5.2 Probabilistic Cases Algorithm

### 5.2.1 Idea

The concept of this first algorithm is to assign for each skill a probability case that will be updated throughout the experience. Each time the user is asked to answer to a question (meaning that she/he is evaluated), we choose a random skill based on these probabilistic cases and then choose a scene that evaluate this skill. If the evaluation was a success, we decrease the probability to pick that skill to the next “probability case”, otherwise, we increase it.

### 5.2.2 Implementation

Here is a pseudo code to give a general idea of the algorithm:

---

```
1    // Initialize values for the different probabilistic cases
2    float probaCase1 = 0.9f          // for example
3    float probaCase2 = 0.3f          // for example
4    float probaCase3 = 0.05f         // for example
5
6    // Assign the probabilities of all skills to probaCase1
7    skillsProbs = {"skill1": probaCase1, ... , "skilln": probaCase1}
8
9    // Each time there is a choice (= we are evaluating a skill)
10   // 1) Choose a skill "skill_i" depending on probability distribution
11   ...
12   // 2) Evaluate the skill I and update its probability
13   // If success, we "upgrade" its probability (for e.g. from probaCase1 to
14   // probaCase2)
15   ...
16   // Else, we "downgrade" its probability (for e.g. from probaCase2 to probaCase1)
17   ...
18   ...
```

---



## 5.3 Bayesian Knowledge Tracing

### 5.3.1 Algorithm

The *Bayesian Knowledge Tracing* is an algorithm used in many intelligent systems that models a person's mastery of a certain knowledge being evaluated. It uses a Hidden Markov Model<sup>[9]</sup> as a latent variable and takes into account the correctness of a student each time we are evaluating her or his skill.

This method considers 4 parameters:

- **$P(L_0)$  or Pre-learn:** The probability that the user already knows the skill before learning it
- **$P(T)$  or Learned/Transit:** The probability for the user to demonstrate knowledge of the skill after an opportunity to apply it
- **$P(G)$  or Guessed:** The probability that the user does not know the skill, but answered correctly by chance ("lucky guess")
- **$P(S)$  or Unusual Slip-Up:** The probability that the user makes a mistake when applying a known skill (something unusual happened, like mind-blank or small error)

The initial probability of a user  $u$  to learn a skill  $k$  was set as the following:

$$p(L_1)_u^k = p(L_0)^k$$

We then compute the conditional probabilities for a correct or incorrect application:

$$p(L_{t+1}|obs = correct)_u^k = \frac{p(L_t)_u^k \cdot (1 - p(S)^k)}{p(L_t)_u^k \cdot (1 - p(S)^k) + (1 - p(L_t)_u^k) \cdot p(G)^k}$$
$$p(L_{t+1}|obs = wrong)_u^k = \frac{p(L_t)_u^k \cdot p(S)^k}{p(L_t)_u^k \cdot p(S)^k + (1 - p(L_t)_u^k) \cdot (1 - p(G)^k)}$$

We can then compute the probability of the skill mastery using the previous equations:

$$p(L_{t+1})_u^k = p(L_{t+1}|obs)_u^k + (1 - p(L_{t+1}|obs)_u^k) \cdot p(T)^k$$

Once we obtained this probability, we adapted our implementation to work with this model. For each skill, we assign an "importance weight" corresponding to the probability of learning the skill. The more weight the skill has, the more likely we will pick it. Hence, since a probability of 0 means that we don't know the skill, and a probability of 1 means that we have completely mastered it, we have assigned a weight of  $(1 - p(mastering\ the\ skill))$  for a skill each time we evaluate it.

### 5.3.2 Learning the parameters

When implementing the BKT algorithm, we noticed that the results may change depending on the initial four parameters we have chosen. To improve our results, we have decided to enhance our algorithm with a learning algorithm to update the parameters depending on the user's choices. After some research, we found a paper<sup>[8]</sup> which describes a method to update these parameters using a *Knowledge Heuristic and Empirical Probabilities*.

Although the paper relates the process with more details, we will describe a brief overview of this algorithm.

### 5.3.2 Empirical Probabilities

*Empirical Probabilities (EP)* is a two-step process, where we have to keep track of the user's performance throughout the experience to update our parameters:

**Annotating Knowledge:** The first step requires to keep track of a user's performance for each skill with an estimate of when the student learned the skill. It assumes that there are only two knowledge states: known (1) and unknown (0).

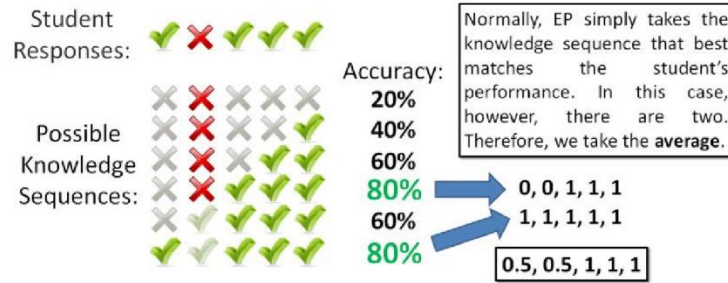


Figure 11: We test all possible knowledge sequences and take those that have the highest accuracy in terms of similarity with the student's answers

**Computing the Probabilities:** Using the knowledge estimates, we can then compute each of the four BKT parameters for each skill empirically from the data.

$$P(L_0) = \frac{\sum K_0}{|K_0|}$$

$$P(T) = \frac{\sum_{i \neq 0} (1 - K_{i-1}) K_i}{\sum_{i \neq 0} (1 - K_{i-1})}$$

$$P(G) = \frac{\sum_i C_i (1 - K_i)}{\sum_i (1 - K_i)}$$

$$P(S) = \frac{\sum_i (1 - C_i) K_i}{\sum_i K_i}$$

# Conclusion

The potential of such interactive stories can still be exploited for a better user experience and there are still many features that can be added to the project. However, we are quite satisfied to have been able to implement the previous enhancements. Having a user-friendly interface based on a node system visualization to generate the dialogues will greatly ease the creation of these experiences and support the accessible aspect of this project.

In addition, implementing skill-based path-choosing algorithms may help to add a more immersive dimension to improve the users' experience. Both algorithms we have studied work well but we may have a preference for the second one, since it takes into account the full history of the user's choices. The project remains still subject to more improvements but seems to be moving in a good direction.

# References

- [1] *“Educational Game Design for an Interactive Experience”*, January 2019
- [2] The *Probabilistic Cases Algorithm* cited in this paper was designed by Thibault Asselborn
- [3] Unity, a cross-platform game engine: <https://unity3d.com/>
- [4] Project repository: <https://github.com/Akynna/InteractiveGame>
- [5] Previous project demo: <https://akynna.github.io/>
- [6] Rete.js, a JavaScript framework for visual programming: <https://rete.js.org/>
- [7] *BKT Algorithm*: [https://en.wikipedia.org/wiki/Bayesian\\_Knowledge\\_Tracing](https://en.wikipedia.org/wiki/Bayesian_Knowledge_Tracing)
- [8] *“Learning Bayesian Knowledge Tracing Parameters with a Knowledge Heuristic and Empirical Probabilities”*: [https://link.springer.com/chapter/10.1007/978-3-319-07221-0\\_18](https://link.springer.com/chapter/10.1007/978-3-319-07221-0_18)
- [9] *Hidden Markov Model*: [https://en.wikipedia.org/wiki/Hidden\\_Markov\\_model](https://en.wikipedia.org/wiki/Hidden_Markov_model)