SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

DUMOULIN Zoé - 3702742

GIGACZ Kiara - 3803025

MU4IN205 - Projet ANDROIDE

Implémentation d'un modèle d'adaptation multi-aspects et sa connexion au modèle de l'apprenant BKT

Rapport final

Encadré par : Amel YESSAD

Année universitaire 2021-2022

Table of contents

Introduction

The objective of this project was to implement the Multi-Aspect Generic Adaptation Model (MAGAM) proposed by Monterrat *et al.* [1] and to connect it to the Bayesian Knowledge Tracing (BKT) model [2].

MAGAM is a generic model that makes it possible to adapt and personalise learning activities according to different aspects that characterise a learner (cognitive profile, motivational profile, time constraints for training, etc.).

BKT is a probabilistic model based on the Hidden Markov Model that calculates the belief of a system on a learner's level of mastery of specific concepts in a certain domain (mathematics, medicine, physics, biology, etc.).

MAGAM was first implemented in PHP in 2017 by Baptiste Monterrat. The main objective of this new implementation was to create a version of MAGAM that allowed users to use BKT to estimate students' mastery of different concepts, as the previous version did not offer this feature. It was also crucial for this new implementation to be more accessible to users with little technical background such as teachers, and at the same time offer flexibility for more experienced users to be able to easily add new calculation options.

We were supervised for this project by Amel Yessad, associate professor at Sorbonne University and part of the MOCAH (Models and Tools in Knowledge Engineering for Human Apprenticeship) team of the LIP6 laboratory.

The project consisted of three main phases: understanding the needs of our "client" (in this case Mrs Yessad) and defining the scope of the project; conceptualising the application, and implementing and documenting it. The fourth phase of the project - the testing and evaluation phase - is under way and that will continue to be carried out in the upcoming months.

The result of our work can be found on GitHub at the following link: https://github.com/dreamlumos/MAGAM.

Context

Different forms of adaptive systems for human learning have been proposed in the past, but most of them describe systems that adapt to one type of user characteristics (cognitive profile, motivational profile, player profile, etc.).

Monterrat *et al.* [1] suggest that it is possible to adapt to different aspects, and to this end proposed a model called MAGAM which achieves this by fusing the output of different processes.

In the domain of adaptive systems,

- the *source* refers to the learners' characteristics that the system adapts to;
- the *target* refers to the element that is being adapted to produce an output (content, presentation, instruction path, etc.); and
- the pathway refers to the manner to adapt the target to the source.

In our report, we will follow the terminology proposed by Monterrat *et al*., which is to combine the source and target under the term *aspect*. An example that they give which helps understand the close link between the source and the target is the following: "a system adapting didactic contents relies on cognitive profiles, while a system adapting the game mechanics of a learning game relies on player profiles".

Different types of aspects exist in the literature. In their paper, Monterrat *et al.* identified six types of aspects that they described in detail:

- the Didactic aspect, referring to the learner's mastery of knowledge,
- the Pedagogic aspect, referring to adaptation according to certain pedagogical approaches,
- the Affective and Motivational aspect, which includes adapting to the learner's interests and emotions,
- the Strategic aspect, referring to systems that interact with a learner to remind them to use certain self-regulating strategies,
- the Learning Styles aspect, referring to the learner's cognitive style, and finally
- the Gaming aspect, referring to the learner's gaming styles and player/personality types.

In the following sections we will briefly present the MAGAM and BKT models to provide further context to the project.

Presentation of MAGAM

As mentioned above, MAGAM is a model that was proposed in order to adapt to multiple aspects. In particular, its aim is to propose adapted learning activities to different students according to their specificities in different aspects.

To achieve this, MAGAM relies on two types of inputs: the M matrix and the Q matrix. Both of these matrices share a common axis, which contains a list of properties which we will define a bit further. The M matrix numerically represents the students' relationships to these properties, whereas the Q matrix represents the relationship between these properties and the available learning activities.

The recommended plan of activities is output in the form of a matrix as well. This is called the R matrix. It has the users on one axis, and the activities on the other. The numerical values inside the matrix indicate the relevance of different learning activities for each student.

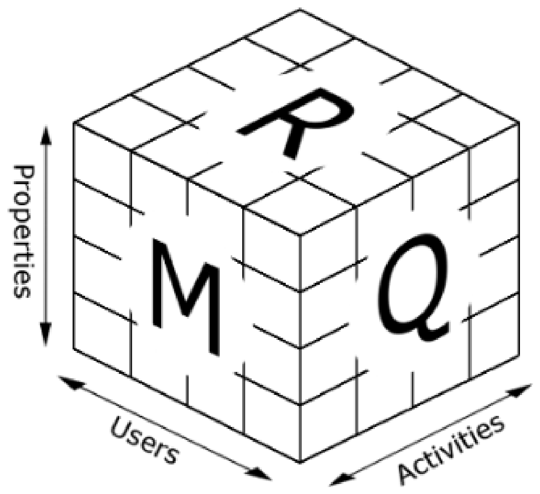


Figure 1: A cubic representation of the MAGAM matrices. [Extracted from source [1]]

For the sake of comprehension, we will henceforth refer to the M matrix as the *user matrix*, the Q matrix as the *activity matrix*, and the R matrix as the *recommendation matrix*.

To understand what the properties we mentioned refer to, we shall use an example from the original paper [1].

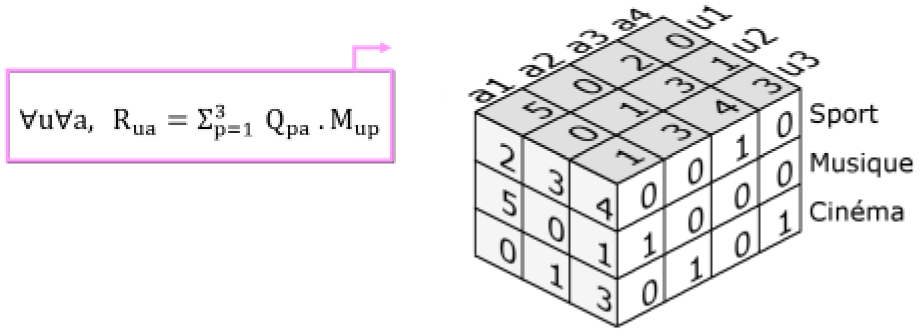

$$\forall u \forall a,\ R_{ua} = \Sigma_{p=1}^{3}\ Q_{pa}\,.\,M_{up}$$

Figure 2: An example of matrices for the motivational aspect [Source: Amel Yessad]

In the figure above, we are considering the motivational aspect. We see that the properties listed are different interests students might have. The values in the user matrix thus refer to a student's affinity for a particular subject. The values in the activity matrix refer to the presence of a particular subject in each activity, 1 if the subject is present, 0 otherwise.

This data is then used to calculate the recommendation matrix which can be seen on the top face of the cube. The calculation applied here between the two matrices is a simple matrix multiplication. This results in a favourable recommendation for activities in which students will find subjects that they have an interest in. For example, student *u1* enjoys music, and thus activity *a1* is highly recommended for this student as it contains references to music.

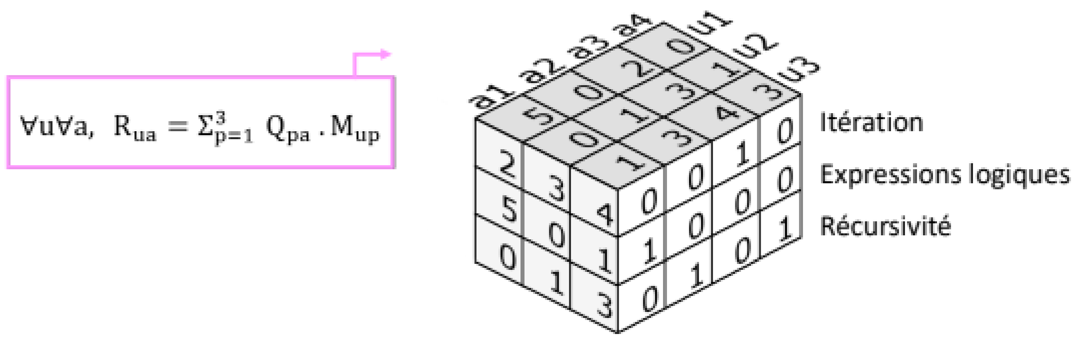

$$\forall u \forall a,\ R_{ua} = \Sigma_{p=1}^{3}\ Q_{pa}\,.\,M_{up}$$

Figure 3: An example of matrices for the didactic aspect [Source: Amel Yessad]

The figure above shows an example for the didactic aspect. In our implementation of MAGAM, users will have the option to use BKT, which we will present just below, to obtain the user matrix of the didactic aspect.

Presentation of BKT

The BKT model [2] relies on a hidden Markov model that is used to infer students' mastery of specific knowledge components. The observations in this model are the students' performances in answering questions pertaining to these knowledge components, and the hidden states are their mastery of these same components.

A correct or wrong answer will each indicate with a certain probability whether the student has mastered the related knowledge components or not. An incorrect answer on a knowledge component that a student has mastered is referred to as a *slip* (parameter S). Likewise, a correct answer on a component that a student has not mastered is referred to as a *guess* (parameter G). At every practice opportunity, a student who has not mastered a specific knowledge component also has some probability of attaining *mastery* (parameter T).

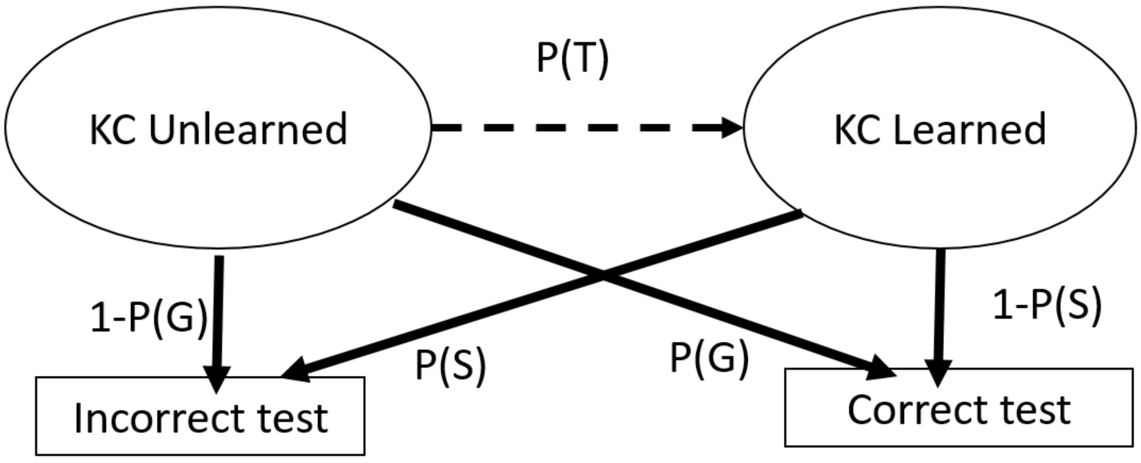


Figure 4: The Hidden Markov Model behind BKT [Source: Amel Yessad]

BKT is trained to infer these probabilities, after which it can be used to estimate a student's mastery of a concept based on their answers to a certain sequence of questions. Each knowledge component is modelled separately without considering any potential relationships between different components.

Other works on individualised BKT models [4] describe different approaches for defining and learning student-specific parameters, but the Python implementation of BKT (pyBKT) which we use follows the original BKT model.

Contribution

Phase 1: Understanding and Defining the Project

The first few weeks of our work on this project consisted of exchanges with Mrs Yessad where she presented the concepts of MAGAM and BKT to us. We also took some time to read through the paper on MAGAM, as well as to study some others that were related, for example on BKT.

To better understand BKT and pyBKT in particular, at the suggestion of Mrs Yessad, we did the pyBKT tutorial that is available online.

This is also the phase in which we tried to understand and define the scope of the project. We obtained screenshots of the first implementation and discussed with Mrs Yessad what needed to be improved in this version. Mrs Yessad also explained to us the types of users that the application would serve, which helped us better understand how to tailor the user interface.

We eventually decided to settle on Python as our programming language, as it seemed suitable to tap into the power of the numpy and pandas libraries for operations with large arrays. This choice was also motivated by the fact that we were learning to use PyQt in our Human-Computer Interaction course this semester, and that based on our research, this library seemed adapted to our needs for the graphical interface. Finally, using Python would facilitate the connection to pyBKT, which is a small but convenient advantage.

Phase 2: Conception

Once the outlines of the project were agreed on, we started to draw some rough prototypes of what the interface could look like. Below are a few examples of prototypes that we drew.

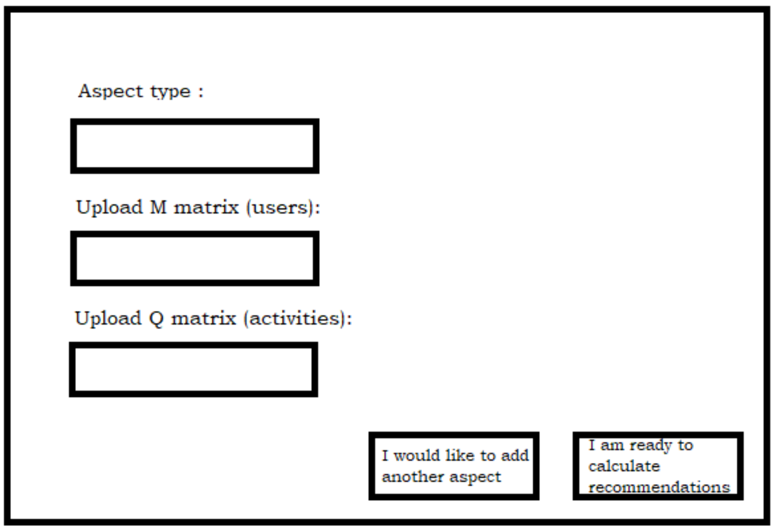
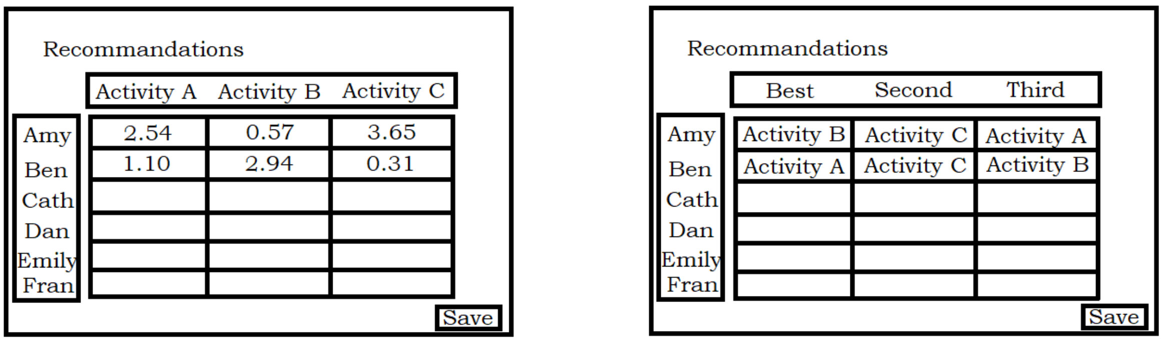

Aspect type :
Upload M matrix (users):
Upload Q matrix (activities):
I would like to add another aspect
I am ready to calculate recommendations

Figure 5: Initial prototype for the data input page



	Activity A	Activity B	Activity C
Amy	2.54	0.57	3.65
Ben	1.10	2.94	0.31
Cath			
Dan			
Emily			
Fran			

	Best	Second	Third
Amy	Activity B	Activity C	Activity A
Ben	Activity A	Activity C	Activity B
Cath			
Dan			
Emily			
Fran			

Figure 6: Initial prototype for the different views of the recommendation tables

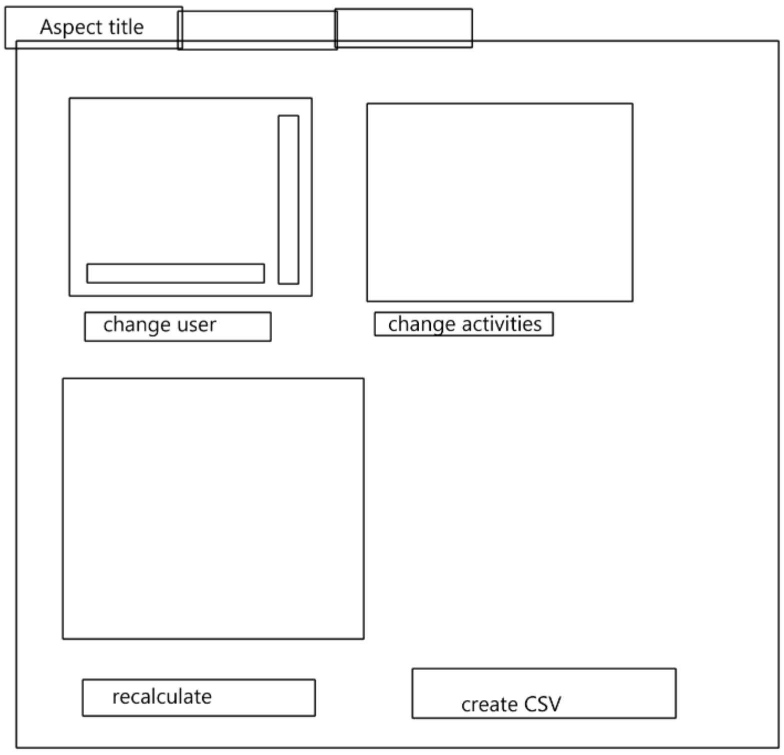


Figure 7: Prototype of the aspect tab

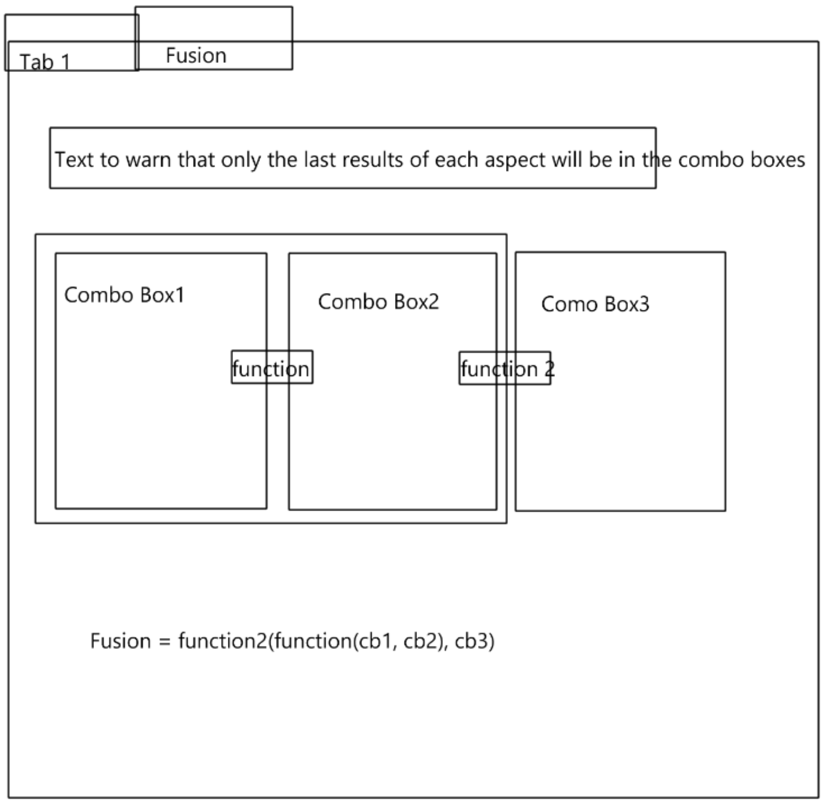

Tab 1
Fusion
Text to warn that only the last results of each aspect will be in the combo boxes
Combo Box1
Combo Box2
Como Box3
function
function 2
Fusion = function2(function(cb1, cb2), cb3)

Figure 8: Prototype of the fusion tab

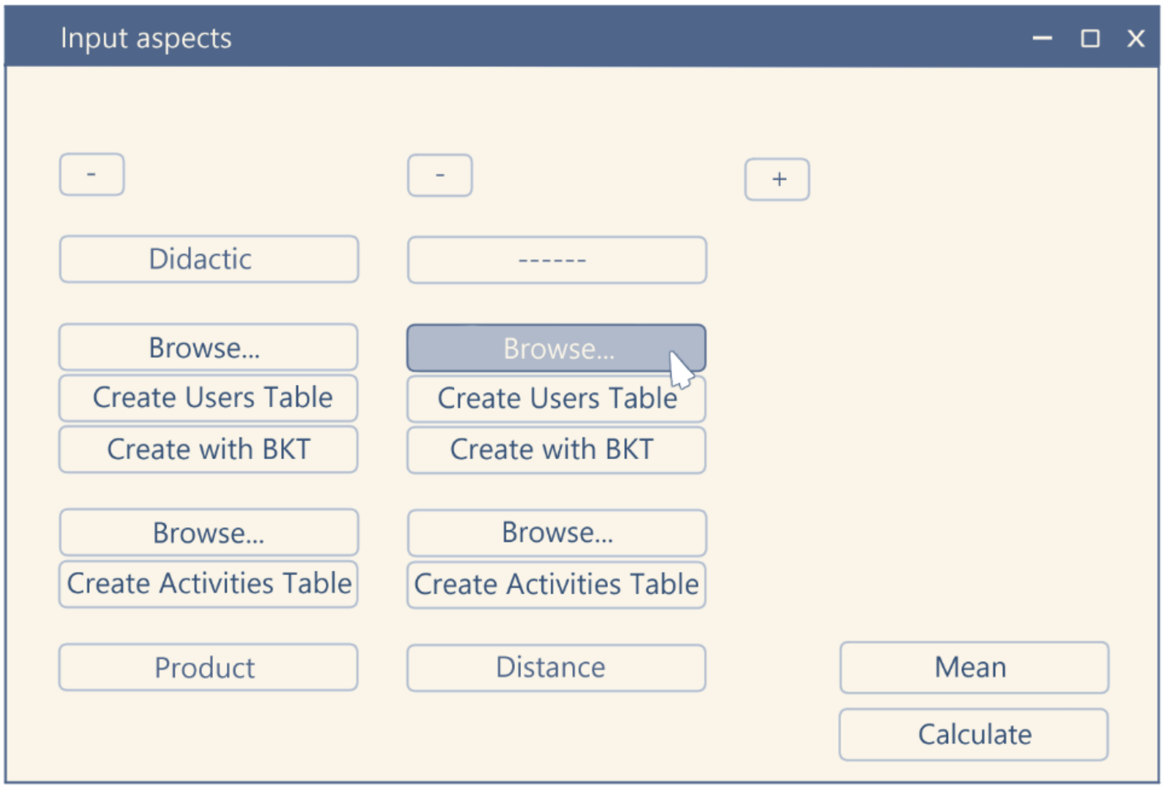


Figure 9: Prototype of the colours of the interface

For the architecture of the code, we decided to go with the Model-View-Controller (MVC) pattern, separating the graphical interface from the model.

Phase 3: Implementation

The implementation of MAGAM consisted of four main tasks: programming the model, programming the user interface, linking the model to pyBKT and finally designing the application with CSS.

Our main plan in terms of division of responsibility was that Kiara would work on the model, and Zoe on the interface. Knowing that the interface would be the main challenge of this project, it was also decided that Kiara would help with the interface once the model was finished.

For the code, we agreed on some basic rules from the beginning and followed through with them. One of these rules was that the code would be written in English. This was decided with and approved by Mrs Yessad, as it would allow the code to be read and used by other researchers or teachers that are not francophones.

We decided to opt for Object-Oriented Programming despite this not being compulsory in Python as this organisation made the most sense to us for this project. We applied ourselves to follow the principle of encapsulation, despite it not being enforced by Python. This, together with the decision to organise the code according to the MVC pattern, allowed us to follow the principle of separation of concerns.

Another rule that we have followed is that the code should be self-documenting as far as it is possible, which means that we preferred writing slightly more verbose code if it made the code easier to understand without the need for comments. It also means that variable and function names were chosen to be as representative as possible of what they were within their context. All this was done with care as to stay within reasonable limits, and not push this principle to its extremes.

3.1 The Model

Implementing the model was one of the more straightforward tasks we had in this project. The data in this application consists mainly of numerical matrices that are created by the user or loaded from an existing file for the duration of the session, and the main operations that are done on this data are mathematical matrix operations. As such, there was no need for an external database.

Two of the main classes defined in the model are the Aspect and Fusion classes. The Aspect class contains all the information related to one aspect that the user has inputted and calculated. The user and activity matrices are stored as *DataFrames* in order to easily conserve the associated column and row names (it is more complicated to do this with numpy's *ndarrays*). Besides this, the class mainly provides a list of getters and setters that allow easy manipulation of the data from outside the class while maintaining a level of coherence in the way information is stored. The class also provides a few functions that allow for internal and external checking and/or maintenance of data coherence. For example, the aspect_filled function ensures that the class contains the necessary data to perform calculations, and the reset_recommendations function ensures that if any input data is changed, the output is reset accordingly.

The Fusion class is very simple. Its aim is to maintain a link between the recommendation tables that were fused and the new table created from this fusion. It also takes care of the conversion of the resulting arrays into *DataFrames* after a calculation is applied.

Another important class in the model is BKTData. This class contains the necessary functions to create a user matrix for the didactic aspect using pyBKT. It takes data from a CSV containing students' performances on questions that test their knowledge of different concepts, and calculates an estimation of their mastery of those concepts. This was one of the more difficult classes to write, as it required a good understanding of how pyBKT works in order to extract the information we needed.

SystemState and Data are two classes that are specific to the application. Their aim is to store information concerning the current user session and provide a coherent and convenient way to access it. For the moment, SystemState only contains a Data object, but its objective in the long term is to also store other information regarding the state of the application, such as the mode (novice or expert) chosen by the user. Data contains a list of all the Aspect and Fusion objects that have been created, and provides an ID for them which the views can then use to ask for more information.

The matrix operations are defined in two classes, one being the BasicFunctions class, which consists of operations that can be applied on a user matrix and an activity matrix to obtain a recommendation matrix. The other class is called FusionFunctions and defines operations that are used to fuse two recommendation matrices into one. These classes were coded separately so that researchers or teachers with some programming experience can easily add new functions to the application without having to change or read any other piece of code but their own. The calculations are done using numpy's *ndarrays* as numpy is generally faster than pandas when performing arithmetic operations.

3.2 The View

The view holds three types of tabs: the overview menu, the aspects tabs, and fusion tab.

The overview menu looks like this.

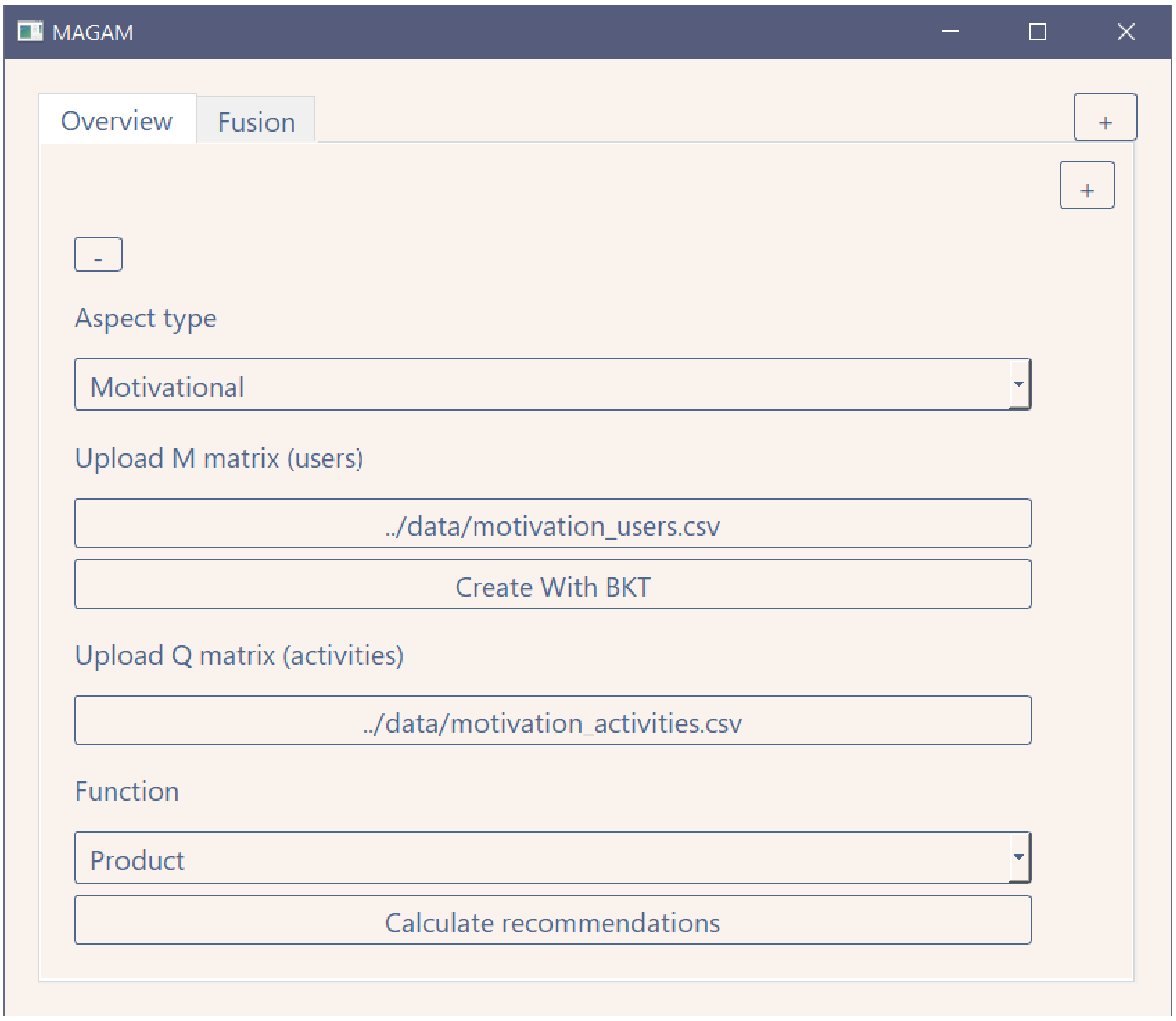

MAGAM
Overview
Fusion
+
+
-
Aspect type
Motivational
Upload M matrix (users)
../data/motivation_users.csv
Create With BKT
Upload Q matrix (activities)
../data/motivation_activities.csv
Function
Product
Calculate recommendations

Figure 10: Overview menu

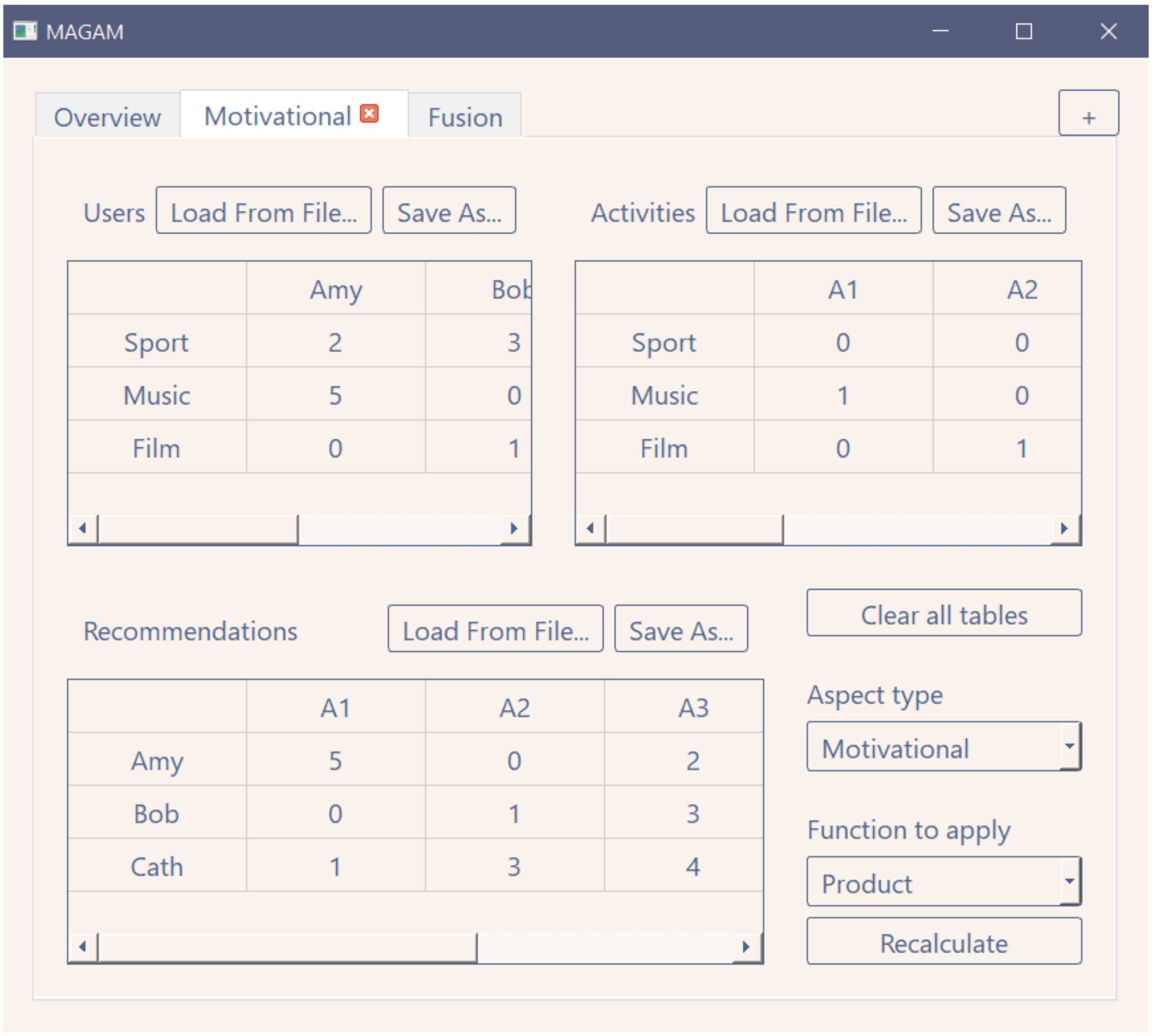
From top to bottom, we first have a list of items that are different *aspect types*. The user can select the one they want. There is a particularity about one of these types, but we will come to it later.

Second and third are the user and activity matrices, to be uploaded from the user's computer.

The last element is a drop-down menu from which the user can select the function that they want to apply to the two matrices.

Once those four items have been selected, the *Calculate recommendations* button is enabled, and the user can compute the recommendation matrix.

This will open a new tab, titled with the corresponding aspect type selected by the user.



MAGAM

Overview | Motivational | Fusion

Users — Load From File... — Save As...

	Amy	Bob
Sport	2	3
Music	5	0
Film	0	1

Activities — Load From File... — Save As...

	A1	A2
Sport	0	0
Music	1	0
Film	0	1

Recommendations — Load From File... — Save As...

	A1	A2	A3
Amy	5	0	2
Bob	0	1	3
Cath	1	3	4

Clear all tables

Aspect type: Motivational

Function to apply: Product

Recalculate

Figure 11: A filled tab

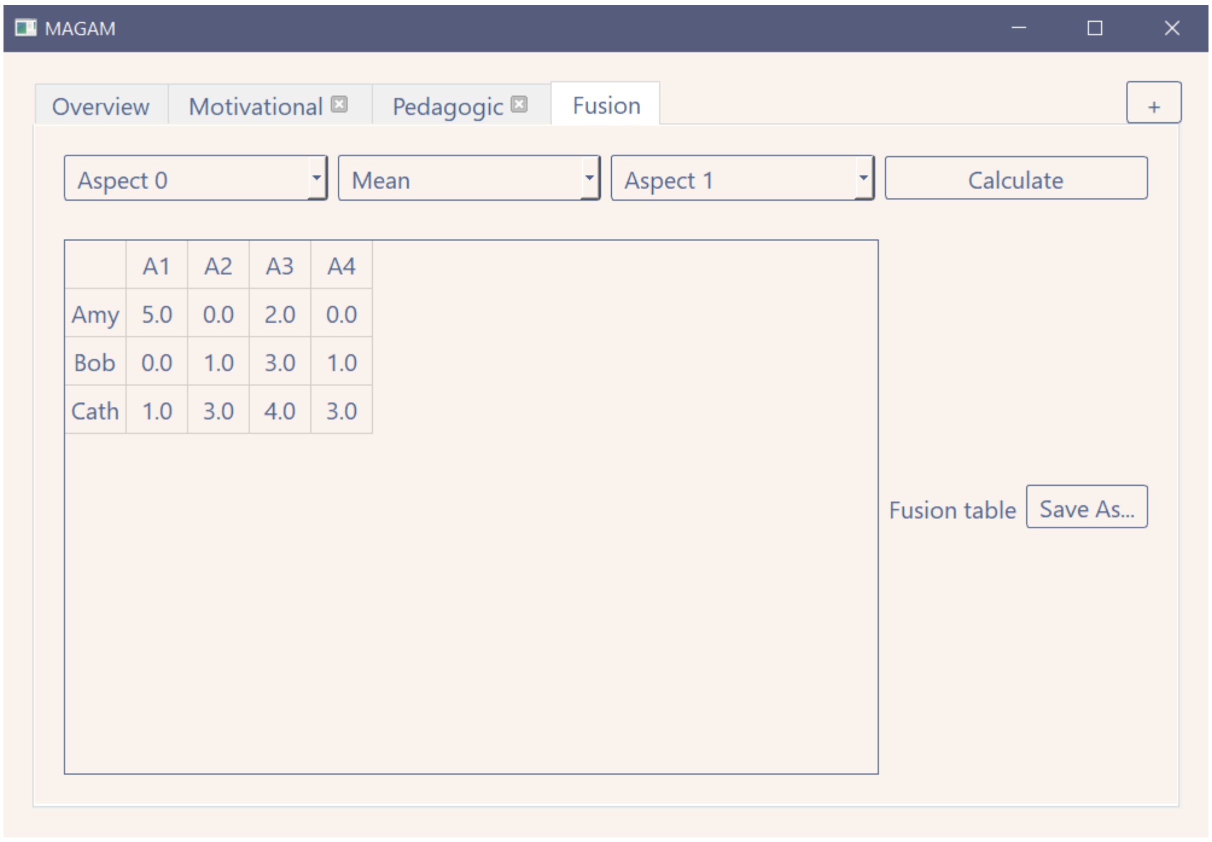
This tab is divided into three sections. The top left section contains the user matrix represented as a table. The top right section is similar but contains the activities matrix.

These two tables are there for the user to verify that they uploaded the right matrices. They are also editable, in case of a typo or a missing number, and the user can then save the edits as a new CSV file.

The bottom section is what will interest most users. It contains the recommendations table that the user wanted to compute in the first place. It is not editable, but can of course be saved as a CSV file. On the right side of this bottom section, we can find the aspect type and the function that has been applied to obtain the recommendations.

Different user and activity matrices can be uploaded by the user, and a new aspect type and function can be applied as well.

Now let's go back to the particularity of one of the aspect types, which is the *Didactic* type. With this type, the user has the choice to create a user matrix using the BKT tool (called through pyBKT). This user matrix models the mastering of each knowledge component by the students.



	A1	A2	A3	A4
Amy	5.0	0.0	2.0	0.0
Bob	0.0	1.0	3.0	1.0
Cath	1.0	3.0	4.0	3.0

Figure 12: The final fusion tab

To realise this interface, we used the GUI module PyQt5. As stated above, we discovered it this year in the context of our HCI class.

We used combo-boxes, slots and signals, QTables, and QSS. When we were almost done with the interface itself, we used QtDesigner, "a tool that acts as a graphical user interface"[8], to try and come up with adequate design and colour schemes for our soon-to-be application.

One of the more complicated points of the interface is that we do not know the size of the users' computer screens, and this makes it difficult to judge what size we must set our application to be. We would have liked to implement a self-adjusting interface, but did not have the time to actually code it, so it is something to be done in the future.

As it is, the font, font size, window size and button width are set using QSS, though they can be modified by anyone who knows some Python and CSS.

3.3 The Controller

The controller was implemented relatively late in the project because of a lack of experience. It was our first time implementing one and we did not think to assign the responsibility of its implementation to either of us.

As such, when we started linking the model and the view together, we quickly fell into the trap of directly calling the model from the view, as it seemed more straightforward initially than writing a "middle-man" that would communicate between them. The view also ended up storing some permanent data that it would send to the model when necessary.

Along the way, we discussed our struggles with implementing the MVC pattern with Mrs Yessad and our HCI lecturer Mr Bailly. These exchanges helped us better understand the role of the controller and since we had by now coded a good portion of the application, we also had a better understanding of how we could reorganise things.

We then decided to reorganise and factorise some of the code in the view and in the model, and implemented the controller. We went back and forth a few times, unsure about which component should know which. Eventually, we settled for an architecture where the view and controller both know each other, and the controller communicates changes made to the data by the user to the model.

Phase 4: Tests, Documentation and Evaluation

At every step of the implementation, regular tests were being made. The model was tested separately from the view, using test data that came from Mrs Yessad's MAGAM presentation slides. The pyBKT-related code was first written and tested on Colab before being integrated into the model. This allowed us to test it with big sets of data like the EEDI database.

The view was also tested with different inputs under different conditions.

With the application now functional, we started to write the user and developer's manuals.

Soon, the evaluation phase will start. External users will be able to test the application and give us feedback on the interface and features.

Challenges

As mentioned above, one of our main challenges with this project was learning how to implement the MVC pattern as it was our first time applying it. Another challenge we faced due to inexperience was in designing the user interface and in programming with PyQt. In particular, we were constantly faced with the question of how to make our application more user-friendly, and as accessible as possible to users without much background knowledge on MAGAM. Our main method of overcoming these challenges was through trial and error, and through discussions with our supervisor and friends that helped give us new perspectives. We will also conduct an evaluation phase where the application will be tested by actual users. This process will also give us concrete feedback on potential improvements to the interface.

One challenge we faced that was due to external factors was the connection of MAGAM to pyBKT. We initially encountered some issues in using the library as it turns out our computers were not powerful enough to run pyBKT in Python, and it took us a while to realise the source of the issue. It was only after comparing the performances of pyBKT in Windows, on Google Colab and in Ubuntu that we realised the better performance in Colab was not only due to the higher RAM but also due to the fact that pyBKT is installed with a C++ extension when a C++ compiler is available.

Another challenge we faced with pyBKT was the lack of documentation for users wishing to use the library beyond its basic use cases. We overcame this challenge by reading the source code, reading the original paper on pyBKT and discussing test results with our supervisor who helped us understand what certain outputs were.

What we've learned

Working on this project as a pair has been a pleasure and a learning experience. We had previously worked together before but not for such a long period of time. This project taught us to better organise ourselves in terms of dividing up responsibilities and coordinating to merge codes.

We also implemented our first project that follows the MVC pattern. This experience has been very valuable as it taught us to write more modular code.

Finally, it has been very interesting to learn about MAGAM, BKT and adaptive systems in the context of learning environments. Using technology for education is a topic we were both interested in, and this project was an opportunity to make a first step into exploring this field of work.

Future perspectives

In the short term, there are small details in the user interface that could be changed to improve the user experience. For example, adding shortcuts such as *Ctrl-C* or *Ctrl-V* when the user wishes to edit a table or copy it to a new document.

In the longer term, we believe it is important for the application to provide support for regular users. For example, a simple feature could be the option to save and continue a session. This could be further developed into a feature whereby an educator can easily update an existing calculation to obtain new recommendations after doing one round of activities with their students.

An ambitious feature that could be very useful for educators would be the option to connect MAGAM to existing learning management systems such as Moodle, or other educational tools or websites that aid with learning like Khan Academy or serious games.

Conclusion

Our main objective with this project was to implement a new version of MAGAM that would be connected to pyBKT. We achieved this through close collaboration with our supervisor, Mrs Yessad, whom we thank for her patient and kind guidance throughout the semester.

The project was carried out in several phases, from the definition of the project to its conception and implementation, then to the final testing and documentation phase.

The application we have created allows users to input data on students and activities, and obtain recommendations on which of those activities are suitable for which students. Users can input this data either by loading it from CSV files or by hand. For the didactic aspect, users also have the choice to calculate the user matrix using BKT.

The result of our work is a functional application that does what it is meant to do. The code base is organised in such a way that users with some Python experience can easily add new calculation functions without having to understand how the rest of the application is coded. Because it follows the MVC pattern, parts of the code can also be easily reused. For example, the interface could be updated in the future with a new graphics library or a more recent version of PyQt without having to change anything within the model.

In its current state though, there is little support for users that are more experienced and thus could make use of a more streamlined interface. There is clearly still a lot of work that can be done to turn the application into a more fluid and convenient tool that would be comfortable to use on a weekly or even daily basis. As mentioned in the Future Perspectives section, creating an API that would allow MAGAM to connect to other educational tools would also make the application much more powerful and potentially be the key to turning it into a viable product.

In conclusion, we are proud of the work that we have achieved despite the challenges that we faced. We would have liked to go further but the time constraints did not permit us to do so. Nevertheless, we are hopeful that our work provides a solid base to be built upon to turn MAGAM into a tool that can empower educators from all walks of life to help students learn and grow in the way that best suits them.

References

[1] Baptiste Monterrat, Amel Yessad, François Bouchet, Élise Lavoué, and Vanda Luengo. 2017. MAGAM: A Multi-Aspect Generic Adaptation Model for Learning Environments. In *Data Driven Approaches in Digital Education*, Élise Lavoué, Hendrik Drachsler, Katrien Verbert, Julien Broisin and Mar Pérez-Sanagustín (eds.). Springer International Publishing, Cham, 139–152. DOI:https://doi.org/10.1007/978-3-319-66610-5_11

[2] Albert T Corbett and John R Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge.

[3] Anirudhan Badrinath, Frederic Wang, and Zachary Pardos. 2021. pyBKT: An Accessible Python Library of Bayesian Knowledge Tracing Models. *ArXiv210500385 Cs* (May 2021). Retrieved February 27, 2022 from http://arxiv.org/abs/2105.00385

[4] Yudelson, Michael V., Kenneth R. Koedinger, and Geoffrey J. Gordon. 2013. Individualized bayesian knowledge tracing models. International conference on artificial intelligence in education. Springer, Berlin, Heidelberg.

[5] The Qt Company Ltd. Qt for Python. Retrieved from https://doc.qt.io/qtforpython-5/.

[6] NumPy. NumPy documentation. Retrieved from https://numpy.org/doc/stable/.

[7] The pandas development team. Pandas documentation. Retrieved from https://pandas.pydata.org/docs/.

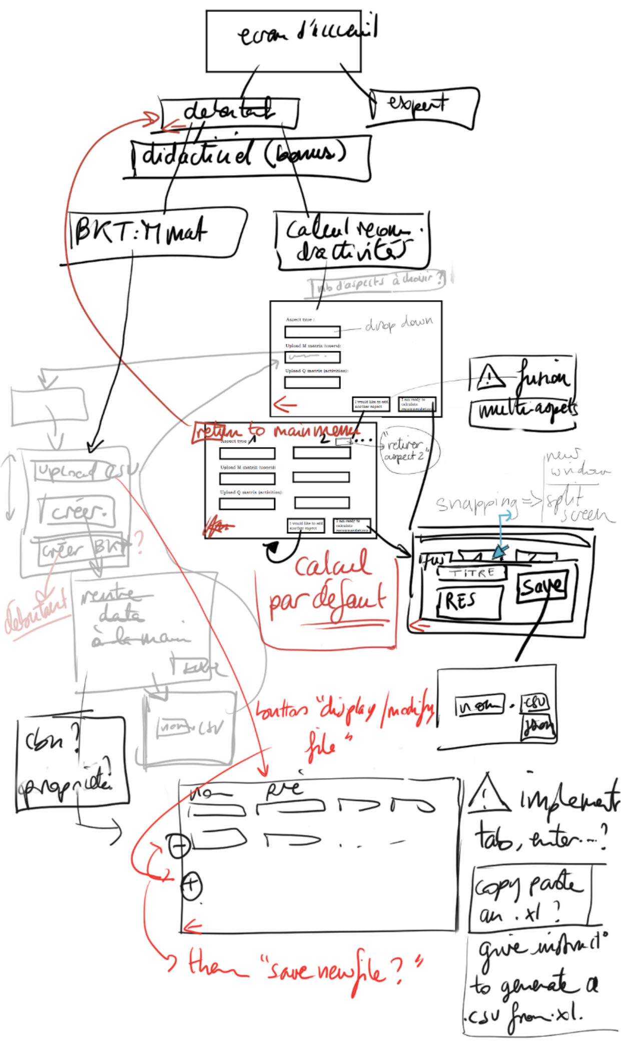
[8] Python PyQt. What is PyQt? Retrieved from https://pythonpyqt.com/what-is-pyqt/.

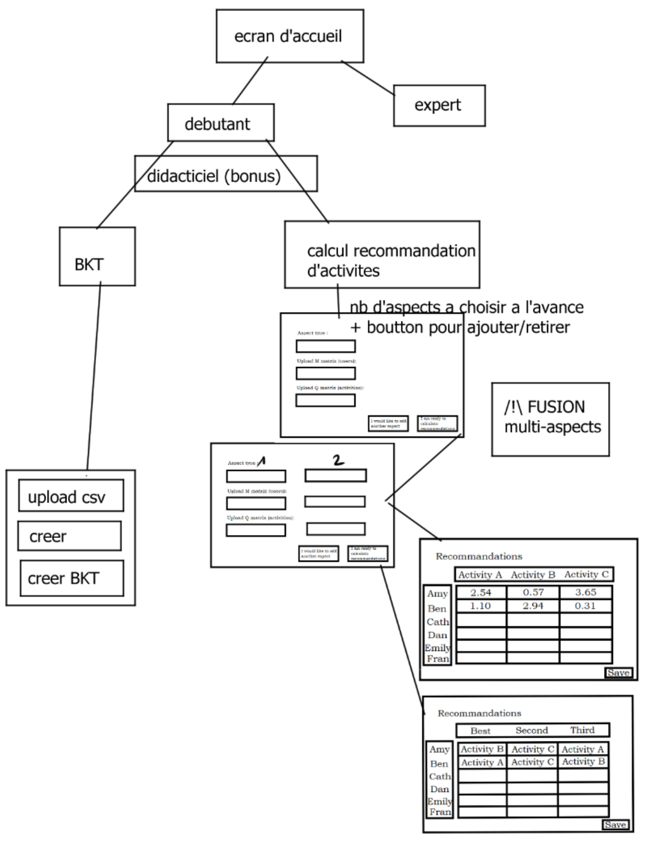
[9] Stack Overflow. Using variables in qt StyleSheets. Retrieved from https://stackoverflow.com/questions/10898399/using-variables-in-qt-stylesheets.

[10] Stack Overflow. Search and replace between two files with search phrase in first file: Python. Retrieved from https://stackoverflow.com/questions/26172875/search-and-replace-between-two-files-with-search-phrase-in-first-file-python.

[11] Stack Overflow. 'staticmethod' object is not callable. Retrieved from https://stackoverflow.com/questions/41921255/staticmethod-object-is-not-callable.

Annexe 1: Other prototype images




ecran d'accueil
expert
debutant
didacticiel (bonus)
BKT
calcul recommandation d'activites
nb d'aspects a choisir a l'avance
+ boutton pour ajouter/retirer
/!\ FUSION multi-aspects
upload csv
creer
creer BKT
Aspect type :
Upload M matrix (users):
Upload Q matrix (activities):
I would like to add another aspect
I am ready to calculate recommendations
1
2
Recommandations
Save

Recommandations

	Activity A	Activity B	Activity C
Amy	2.54	0.57	3.65
Ben	1.10	2.94	0.31
Cath			
Dan			
Emily			
Fran			

Save

Recommandations

	Best	Second	Third
Amy	Activity B	Activity C	Activity A
Ben	Activity A	Activity C	Activity B
Cath			
Dan			
Emily			
Fran			

Save

Annexe 2: Extract of the Specifications

Définition des besoins

Le client souhaite que les utilisateurs puissent soumettre un certain nombre de données sur des élèves, sur des activités à recommander aux élèves et sur des aspects d'adaptation afin d'obtenir des recommandations d'activités adaptées à chaque élève.

Les utilisateurs de la solution seront des chercheur.se.s et potentiellement des enseignants (du second degré ou du supérieur). Avec notre nouvelle implémentation, un besoin clé est que le code du logiciel soit facile à prendre en main par des utilisateurs non informaticiens.

Les utilisateurs doivent pouvoir rentrer à la main des données de type *profils d'élèves, types d'activités, propriétés de ces activités,* ou charger un fichier .csv à soumettre à BKT.

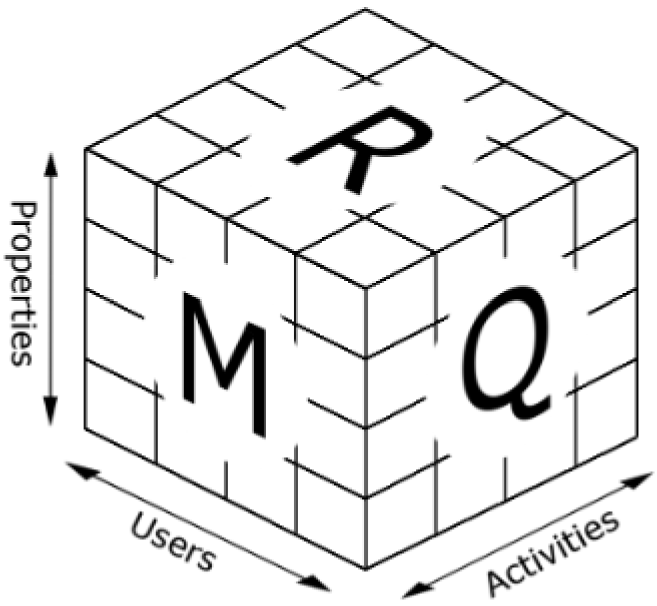
Pour les profils d'élèves, les aspects d'adaptations possibles sont (i.e. ce à quoi peut s'adapter MAGAM) :

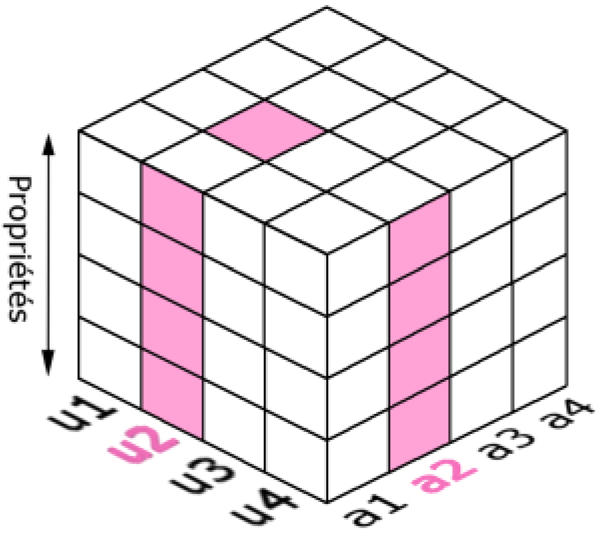
- l'aspect didactique,
- l'aspect pédagogique,
- l'aspect stratégique,
- l'aspect motivationnel et émotionnel
- les styles d'apprentissage, et
- l'aspect ludique.

Le système devra pouvoir s'adapter aux choix des utilisateurs sur le type de calcul souhaité et les priorités des différents aspects dans la fusion.

A partir des données que l'utilisateur aura soumis, le système sera en capacité de calculer des recommandations à partir de deux matrices, et éventuellement faire la fusion des recommandations si plusieurs aspects sont choisis.

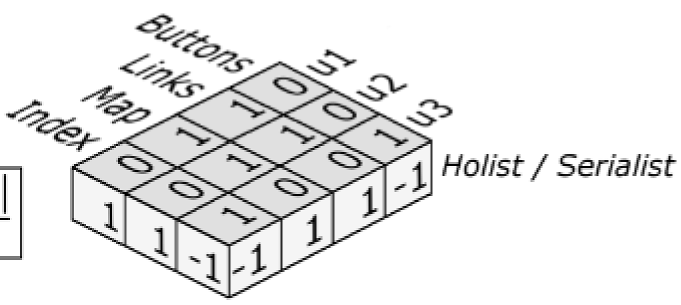
Les schémas ci-dessous montrent comment les données sur les élèves et celles sur les activités peuvent être mises en commun par les propriétés pour calculer les recommandations.



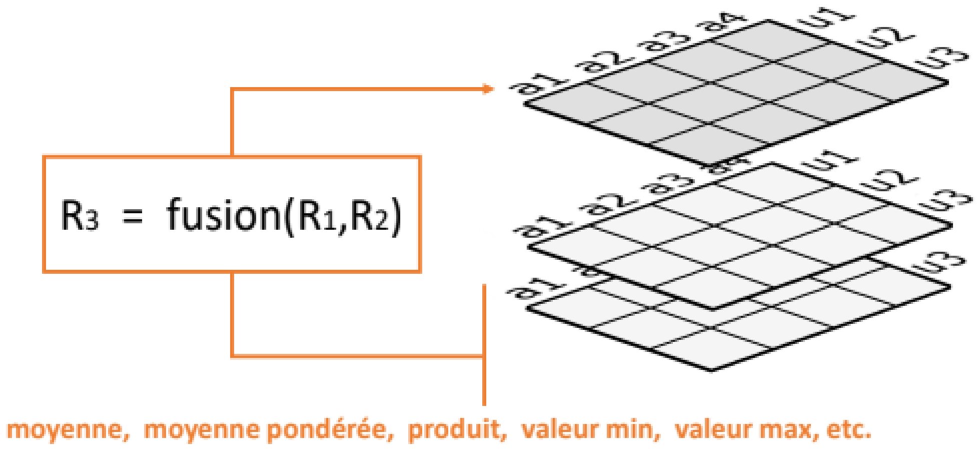


Pour prendre un exemple très simple, le schéma ci-dessous montre comment un élève qui a pour personnalité d'être holiste sera recommandé des activités qui permettent de voir l'ensemble du contenu sous la forme d'une carte hiérarchique.

$$\boxed{\forall u \forall a,\ R_{ua} = \Sigma_{p=1}^{pmax} \frac{|Q_{pa} + M_{up}|}{2}}$$



Pour la fusion, il y a différents types de calculs possibles parmi lesquels l'utilisateur pourra choisir, dont la moyenne, la moyenne pondérée, le produit, la valeur minimale ou la valeur maximale. Le schéma ci-dessous montre comment ce calcul est appliqué pour obtenir une matrice finale de recommandation.


R3 = fusion(R1,R2)
a1 a2 a3 a4
u1 u2 u3
moyenne, moyenne pondérée, produit, valeur min, valeur max, etc.

Le logiciel devra avoir une interface graphique qui est simple à comprendre et utiliser, où l'utilisateur pourra soumettre les données, faire les choix nécessaires et visualiser les recommandations générées.

Solution envisagée

Choix de langage

Nous envisageons d'utiliser le langage Python pour implémenter MAGAM. Il y a plusieurs raisons pour cela. Premièrement, il y a de très bonnes bibliothèques pour la manipulation de données et de calcul matriciel en Python. Deuxièmement, la bibliothèque PyBKT est en Python, et cela facilitera la connexion de celle-ci avec notre logiciel. Python est aussi un langage facile à prendre en main et proche du pseudo-code, ce qui peut permettre à des enseignants sans expérience en programmation de comprendre l'implémentation et de faire des modifications légères avec relativement peu de temps investi.

Pour l'interface graphique, nous utiliserons la bibliothèque PyQt qui permet de créer des interfaces qui s'adaptent aux différents systèmes d'exploitation.

Documentation

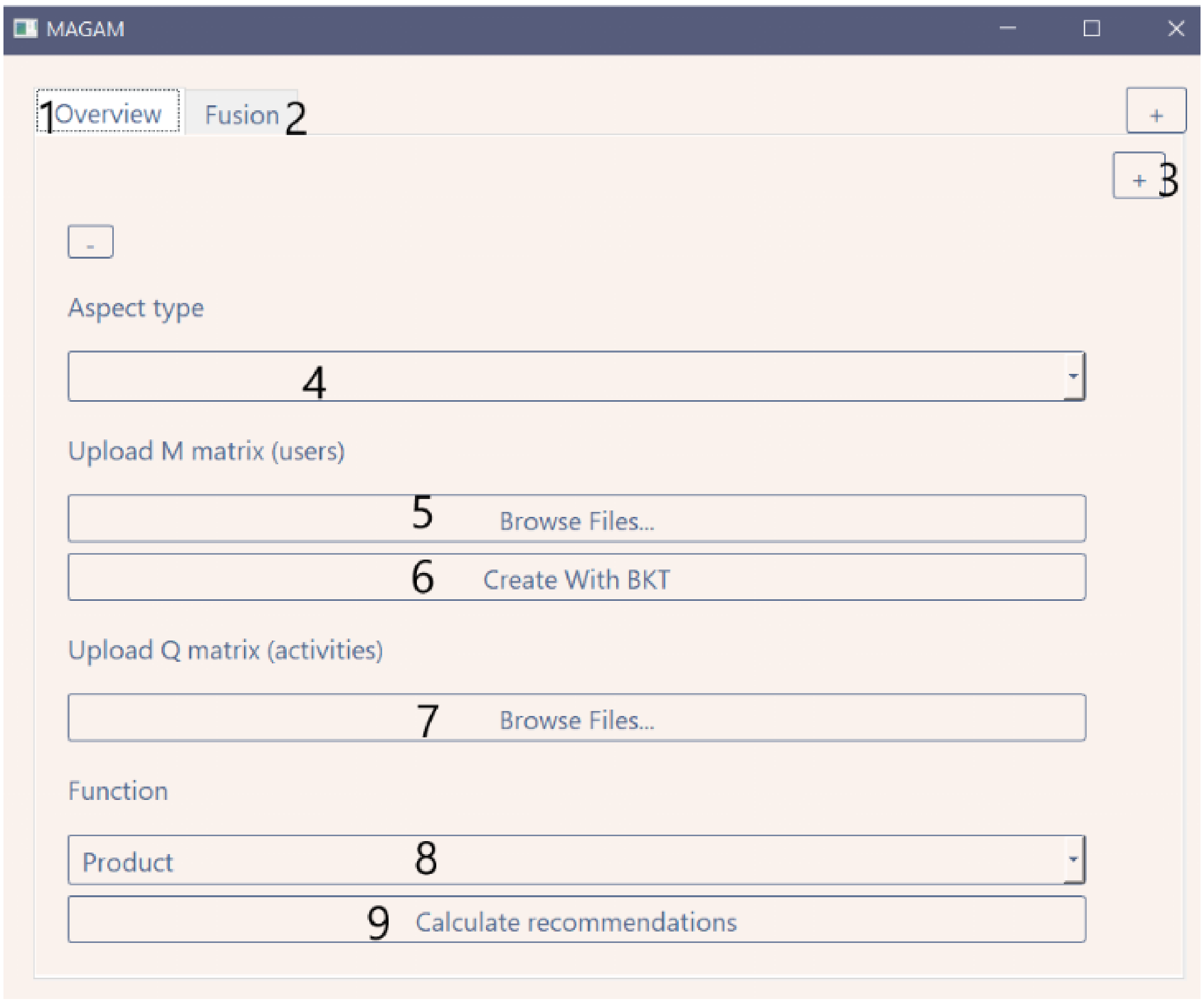
La solution finale contiendra un manuel d’utilisateur et/ou un tutoriel pour expliquer la prise en main du logiciel. Elle contiendra également un manuel pour des développeurs qui souhaiteraient modifier ou étendre le logiciel.

Objectifs secondaires si le temps le permettra :

- Différentes manières pour les utilisateurs de saisir des données : JSON, CSV, XLS
- Générer un exécutable pour les utilisateur.rice.s qui n’ont pas Python
- Créer une API qui permettrait à des développeurs de connecter MAGAM à des outils comme Moodle ou Khan Academy pour avoir un feedback direct sur les performances des élèves sur des exercices

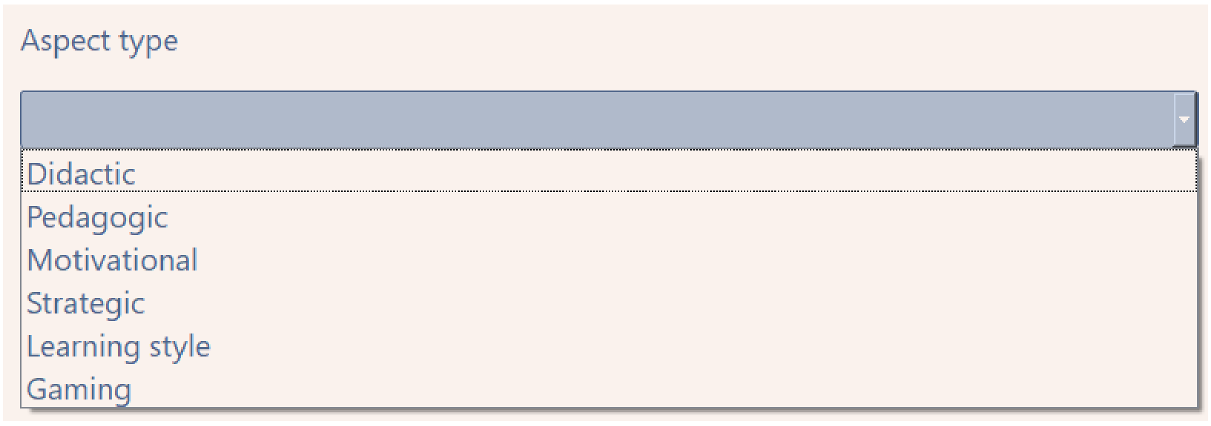
Annexe 3: User manual

This application might require you to upload CSV files. Examples of the required formatting can be found here: https://github.com/dreamlumos/MAGAM/tree/main/data.
Be sure to follow this exact format, or you risk getting incorrect results.
Once you've launched the app, this window will appear.

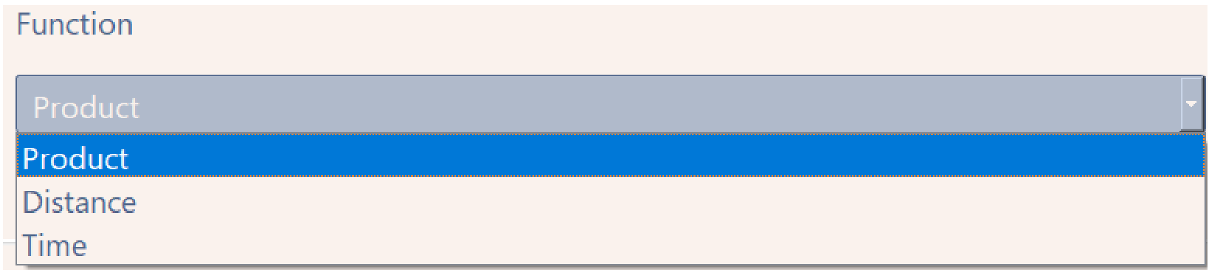


The tab (1) you see is the *Overview* tab. You cannot close this tab, as it is akin to the main menu.
Tab (2) is the *Fusion* tab. We will describe it in more detail later in this manual.
Button (3), when clicked, adds most of the same drop menus underneath.

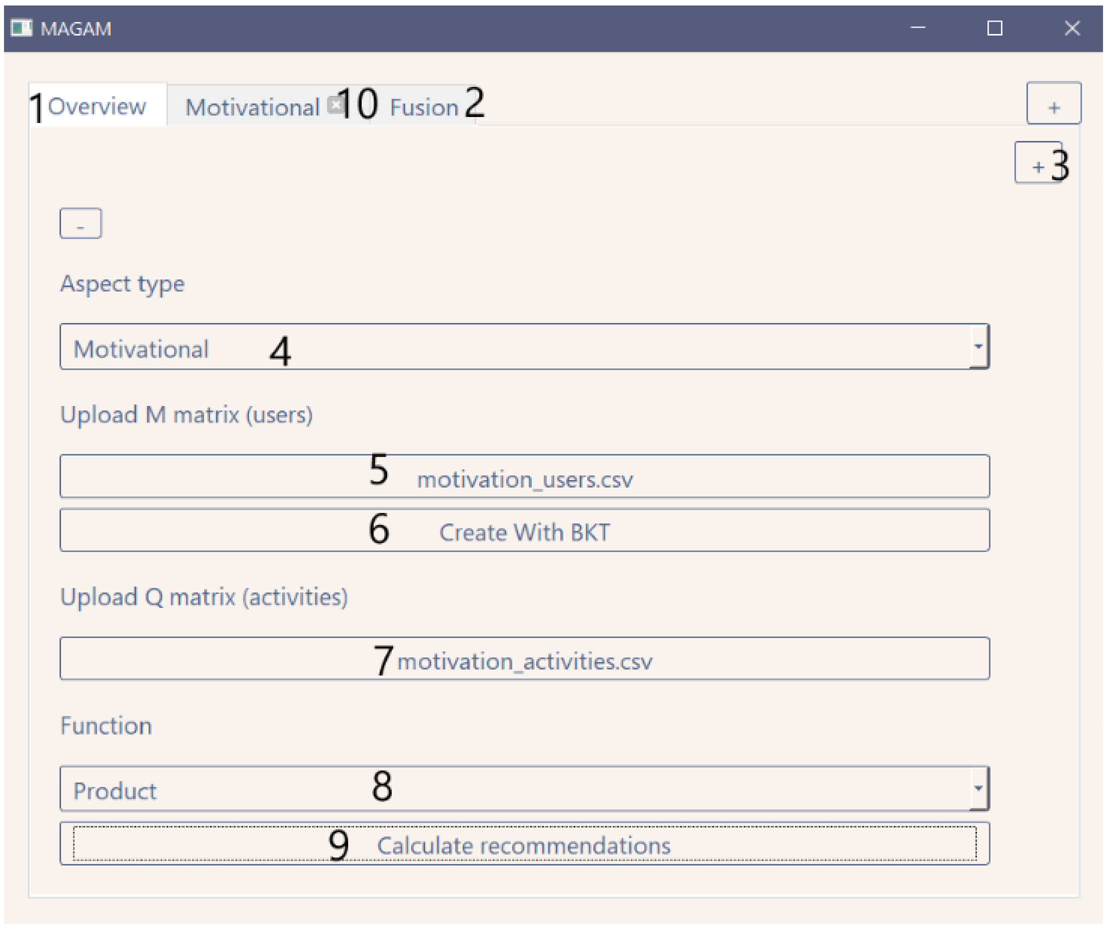
Now, for the actual aspect. (4) is where you will select the aspect type, like so:



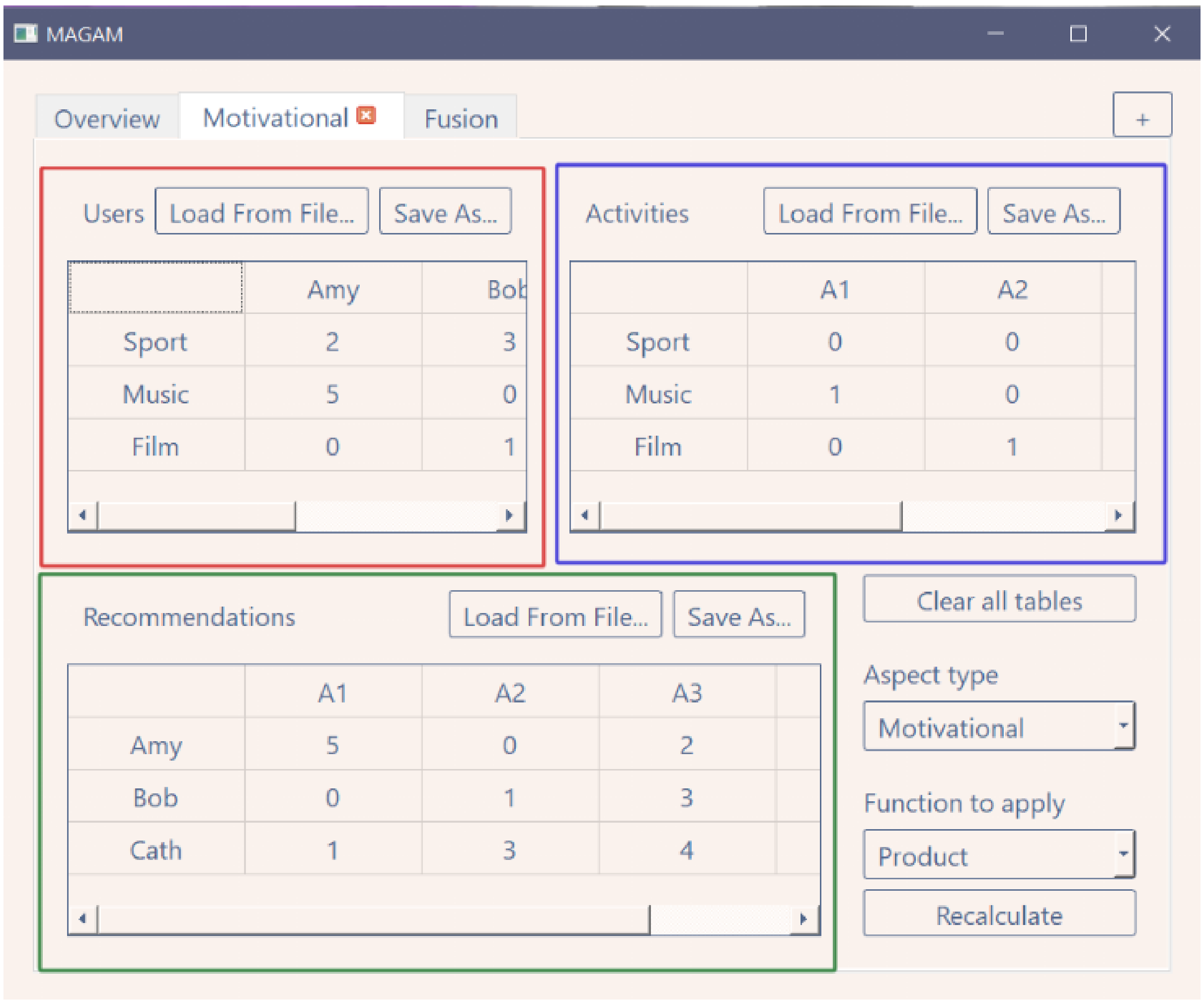
(5) is where you can browse your computer for a CSV file containing a *Users matrix*.
(6) can only be clicked on if the *aspect type* you picked in (4) is the *Didactic* aspect, we will also go into more detail regarding this later in the manual.
(7) is the same as (5), but this time for the *Activities matrix*.
(8) is where you can select the function you want applied to your two matrices, like so:



It is by default set to *Product*. Three basic functions are implemented, and you can refer to the Developer's Manual if you wish to add your own.
And finally, once you have uploaded your matrix and selected the aspect type and the function, you can click on (9).



Your interface should now look like this, with the names of the files you selected showing. A new tab, (10), was created when you clicked on (9).



Users

	Amy	Bo
Sport	2	3
Music	5	0
Film	0	1

Activities

	A1	A2
Sport	0	0
Music	1	0
Film	0	1

Recommendations

	A1	A2	A3
Amy	5	0	2
Bob	0	1	3
Cath	1	3	4

Aspect type: Motivational

Function to apply: Product

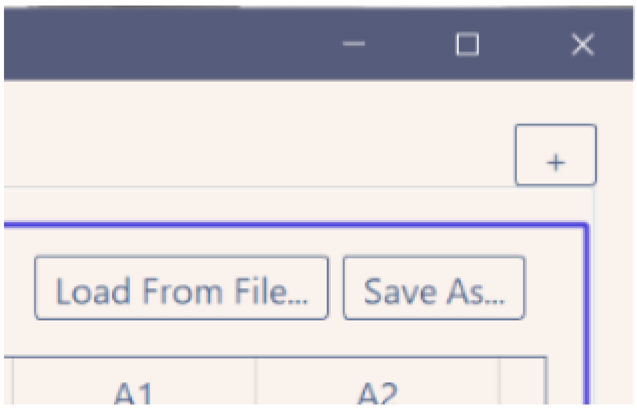
Here we have what the tab displays after having been created. Outlined in red is the *Users matrix* that you uploaded in (1), and in blue is the *Activities matrix*. Those two sections serve the same purpose. The two tables in which the matrices are displayed can be edited, in case of a typo.

The sections both include two buttons. The "Load From File…" button exists in case you realise here that you uploaded the wrong matrix. You can use the "Save As…" button if you edited the tables and want to save the new matrix for later use. This will save the matrix as a CSV file.

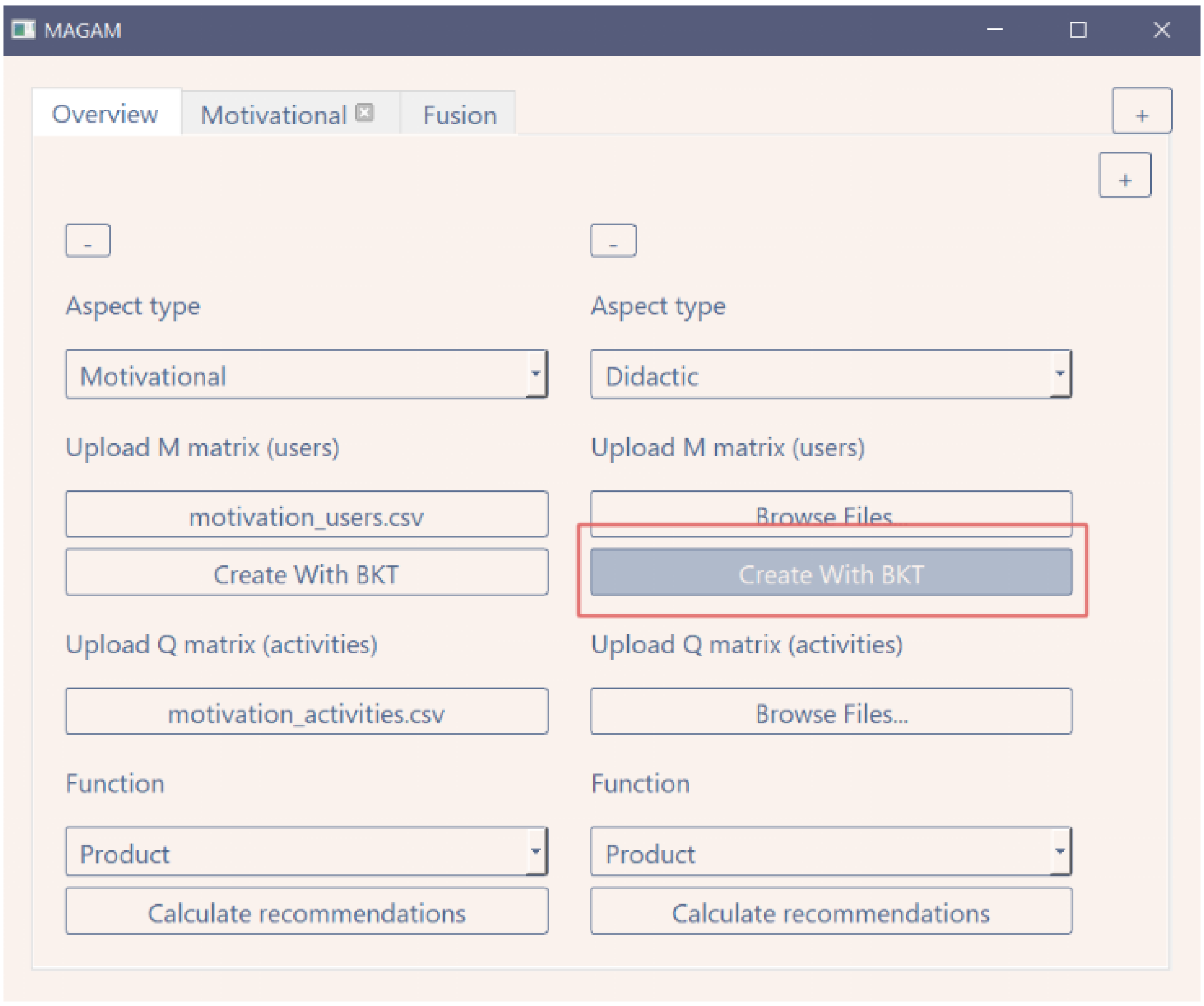
The green section displays the *Recommendations matrix.* It is the result of the function you applied to the *Users* and *Activities matrices.* The table is non-editable, however you can still save the matrix as a CSV file using the "Save As…" button. The "Load From File…" button enables you to directly upload a *Recommendations matrix* that you might have already saved as a CSV file. This will be useful for the *Fusion* tab later.

To the right of this section you can find the *Aspect type* that you picked earlier, as well as the function you applied. There is also a "Clear all tables" button that enables you to clear all data displayed on this tab.

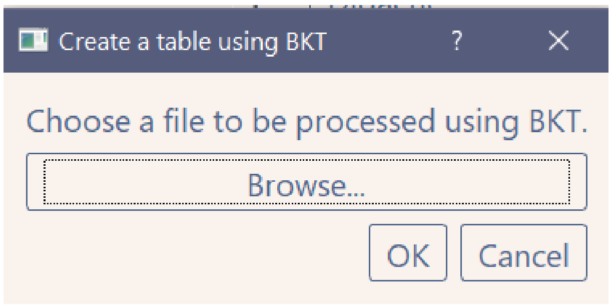
To create a new, *empty* tab, you can also click on the "+" button in the upper right corner of the window.



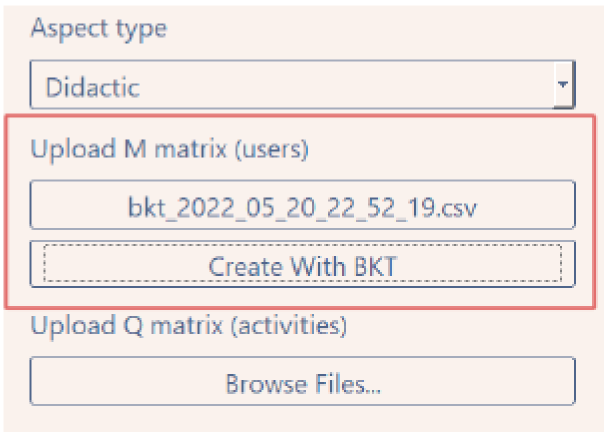
Now, if in the *Overview* tab you picked the *Didactic* aspect, here are your options.



Outlined in red is the "Create With BKT" button. If you click on it, a pop-up window will appear.

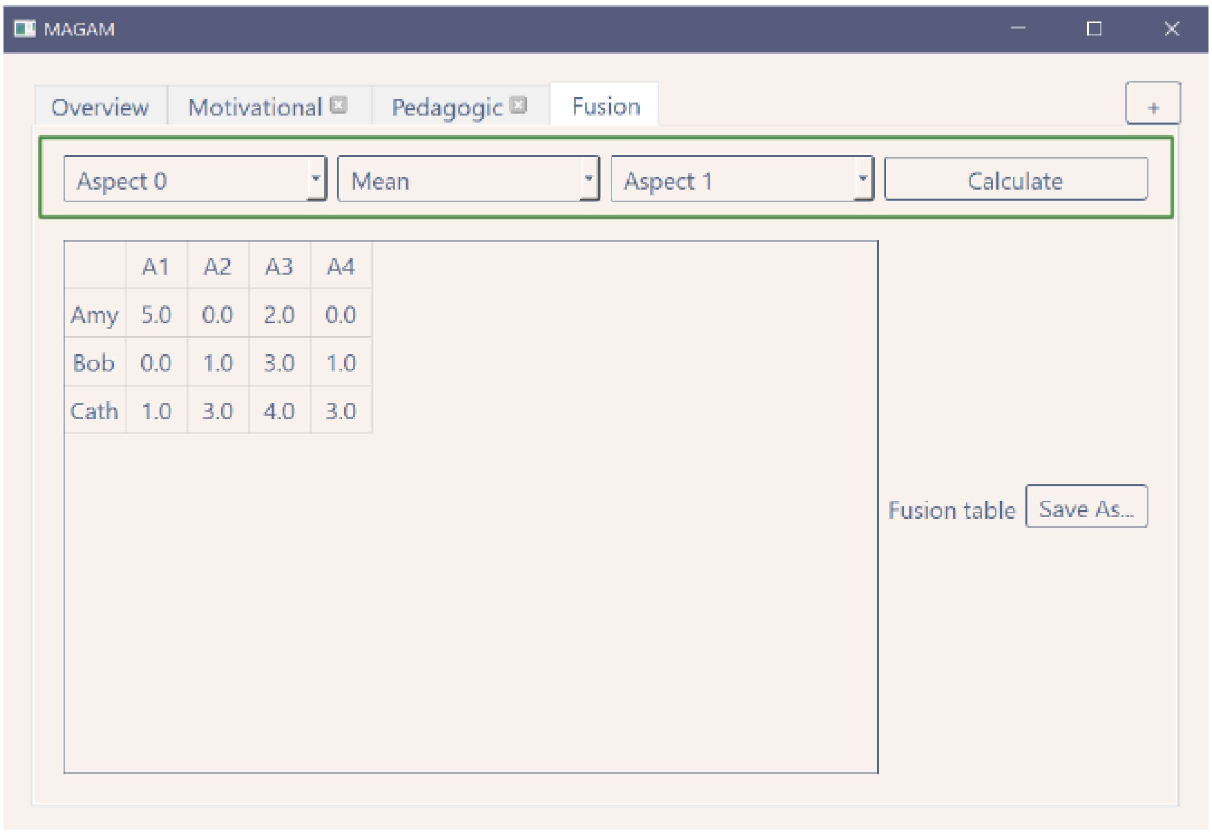


You then need to upload a CSV file with correctly named headers (see this file) and BKT will process it. It will then output a new file, save it as a CSV where your input file is, and automatically upload it to the interface.



This BKT option can only be used when in the *Overview* tab.

Next is the *Fusion* tab.



	A1	A2	A3	A4
Amy	5.0	0.0	2.0	0.0
Bob	0.0	1.0	3.0	1.0
Cath	1.0	3.0	4.0	3.0

Outlined in green, from left to right, we have three dropdowns and one "Calculate" button. The first and third dropdowns are the *Recommendations matrices* that you can choose to *fuse.* The second dropdown is the fusion function that you want to apply. The resulting matrix is displayed below all these.

The options you are able to choose from in the dropdowns are the *Recommendations* that you calculated since you opened the application. Once you have computed a fusion however, you can see that the resulting matrix will be available in the dropdowns. You can thus fuse more than two matrices.