Sorbonne Université

Environnements Virtuels Hautement Interactifs

# EVHI : Projet

*Auteurs :*
BIEGAS Emilie (3700036)
YANG Zitong (3872648)

01 Décembre 2021

# 1 Introduction de l'application

Nous proposons de développer une application d'apprentissage de langue sous forme de série de questions de vocabulaire (sous forme de question à choix multiples (QCM) ou en demandant à l'utilisateur d'écrire la réponse en entier). C'est un jeu sérieux destiné à toute personne voulant améliorer son vocabulaire en anglais.

Nous allons pour cela modéliser les connaissances de l'utilisateur afin de pouvoir lui proposer des questions adaptées (que ce soit au niveau de la forme de la question (QCM ou autre) ou au niveau du mot de vocabulaire demandé), notamment en s'appuyant sur la "Power Law of Practice". En effet, chacun des mots de vocabulaire va être associé à une valeur entre 0 et 1 qui quantifie la probabilité d'acquisition de ce mot par l'utilisateur (1 lorsque le mot est totalement acquis et 0 s'il n'est pas acquis ou que l'on n'a pas encore testé l'utilisateur sur ce point). Cette valeur va évoluer au cours du temps et en particulier va diminuer en suivant la "Power Law of Practice" (qui prendra également en compte le nombre de fois où le mot en question a été demandé).

Nous allons également utiliser le "Keystroke level Model" (afin de mesurer la vitesse du clic ou de l'entrée de texte de l'utilisateur en comparaison à ce modèle) ainsi que les traces récoltées par l'oculomètre afin de déterminer si l'utilisateur semble avoir répondu au hasard ou s'il connaissait réellement la réponse. En effet, on a tendance à regarder plusieurs fois toutes les réponses les unes après les autres et a entrer la réponse plus lentement lorsque l'on hésite et plutôt à regarder une seule réponse plus longtemps et à la sélectionner rapidement lorsque l'on est sûr de nous. Cette estimation d'hésitation va, entre autre, permettre d'adapter les probabilités d'acquisition du mot de vocabulaire.

En fonction des connaissances de l'utilisateur et des traces recueillies, notre système va adapter les questions posées à l'utilisateur et proposer un retour à celui-ci sous forme de fiche explicative ciblée sur l'origine de ses lacunes, c'est-à-dire, par exemple, lorsqu'il semble confondre deux notions proches ou lorsqu'il confond la nature des mots (s'il écrit un verbe alors que l'on lui a demandé un adjectif par exemple).

# 2 Cahier des charges

Afin de mener à bien ce projet, listons les principales tâches à effectuer :

- ☐ Modéliser les connaissances de vocabulaire de l'utilisateur en associant à chaque mot de vocabulaire une valeur entre 0 et 1 déterminant la probabilité d'acquisition de ce mot par l'utilisateur ainsi qu'une valeur entière supérieure ou égale à 0 désignant combien de fois le mot lui a été demandé afin de déterminer l'ancrage de cette connaissance (selon la "Power Law Of Practice") et de pouvoir différencier une probabilité d'acquisition de 0 due à des lacunes de l'utilisateur et une probabilité d'acquisition initialisée à 0 lorsque cette connaissance n'a pas encore été testée par notre système.
  La probabilité d'acquisition d'un mot va diminuer progressivement en suivant la "Power Law Of Practice" (en fonction du nombre de fois où le mot a été demandé, puisque l'on a vu que la diminution se faisait de manière plus drastique lors des premières apparitions du mot que par la suite) jusqu'à ce que l'on questionne l'utilisateur sur ce mot. Lorsque l'on questionne l'utilisateur sur ce mot, la probabilité d'acquisition va alors être mise à jour à 0 si l'utilisateur s'est trompé et n'a pas hésité, 0.25 si l'utilisateur s'est trompé mais a hésité avec la bonne réponse, 0.75 si l'utilisateur a bien répondu mais a hésité, et 1 si l'utilisateur a entré la bonne réponse sans hésitation. Le fait de savoir si l'utilisateur a hésité ou non fait partie de nos prochaines tâches.

- ☐ Mettre en place un moyen de différencier les natures de mot et modéliser les connaissances de l'utilisateur concernant la nature des mots en anglais afin de déterminer si l'apprenant ne confond pas différentes natures de mot (adjectif et nom par exemple). Les liens entre les différentes classes de mots (nom commun, adjectif, verbe, etc.) vont être décrits de façon symbolique sous forme de règles et vont permettre de voir si l'apprenant maîtrise ces notions (grâce à un raisonnement déductif). Par exemple, si un mot se finit par dom, c'est sûrement un nom (l'adjectif libre, en anglais free, se transforme en nom (liberté) en lui ajoutant le suffixe -dom pour créer le mot freedom). Il y aura donc une règle qui dira que si le mot se finit par "dom" alors c'est un nom. Autre exemple : la transformation d'un verbe en adjectif peut s'effectuer en lui ajoutant le suffixe -able (le verbe casser, en anglais break, devient un adjectif (cassable) en lui ajoutant le suffixe -able pour créer le mot breakable). Il y aura donc une règle qui dira que si le mot se finit par "able" alors c'est un adjectif. Nous allons ainsi mettre en place des règles évaluant la nature d'un mot en fonction de son suffixe en anglais. Ainsi, si l'utilisateur s'est trompé mais qu'il a entré un mot similaire (c'est à dire un mot dont le préfixe est semblable au préfixe du mot attendu), connaissant la nature du mot demandé à l'utilisateur, on pourra déterminer, en fonction du suffixe entré par l'utilisateur, quelles natures de mot il confond. Les connaissances de l'utilisateur sur les natures de mots vont être représentées sous forme de tableau dans lequel chaque case contiendra une valeur entre 0 et 1 caractérisant le fait que l'utilisateur sait bien faire la différence entre ces deux natures (1) ou pas du tout (0). Notons que ce tableau sera symétrique (une seule moitié est à considérer).

□ Modéliser le comportement oculaire de l'utilisateur afin de déterminer s'il fait face à une hésitation. Pour cela, nous allons prendre en compte à la fois le temps total que l'utilisateur passe sur chacun des mots (donc une durée par mot proposé), ainsi que le nombre de fois où il a regardé chaque mot (donc un nombre par mot proposé). Afin de ne pas avoir à comparer des vecteurs sur trop de critères (si on gardait les données telles quelles, on devrait comparer des vecteurs de taille 2 fois le nombre de proposition dans le QCM) et afin de pouvoir proposer des QCM avec un nombre de choix différents (si on gardait les données telles quelles, on ne pourrait pas faire de comparaisons entre des données provenant de deux QCM avec un nombre de choix de réponse différent), nous allons représenter les données oculaires sur seulement deux critères : la différence maximum de temps passé sur un mot entre deux mots et le nombre de fois total où l'utilisateur a bougé son regard sur un autre mot divisé par le nombre de propositions (donc la somme du nombre de fois où il a regardé chaque mot divisé par le nombre de propositions). Notons que la division par le nombre de propositions va permettre de comparer les données de deux QCM avec un nombre de choix de réponse différent. Pour le premier critère, nous écarterons le temps d'une réponse si celle-ci est nettement moins regardée que les autres. Nous considérerons que l'utilisateur est sûr que cette réponse n'est pas la bonne. En effet, sinon, le premier critère serait faussé si, par exemple, l'utilisateur hésite entre seulement deux réponses parmi les trois proposées (la plus grande différence de temps passé sur deux réponses sera alors grande alors que le temps passé sur les réponses sur lesquelles il hésitait était très proche). En effet, dans ce cas, on veut estimer que l'utilisateur hésite. De la même manière, si l'on écarte une réponse pour le premier critère, nous l'écarterons pour le second critère également (que ce soit dans le comptage ou dans la division à effectuer).
Le choix de cette représentation va permettre d'illustrer le fait qu'en général si l'utilisateur passe le même nombre de temps, un temps plutôt long (que l'on calculera à la prochaine étape), et un nombre conséquent de fois sur chacun des mots, c'est qu'il hésite fortement. Ainsi, plus le premier critère est faible et plus le second critère est élevé, plus on considérera que l'utilisateur hésite.
Nous allons ainsi pouvoir approximer son comportement en fonction de son hésitation. Nous allons pour cela représenter deux comportement : l'utilisateur est sûr de lui ou l'utilisateur hésite. Ces deux comportement vont donc être représentés par des données sur les deux critères vus plus haut. Nous allons initialiser ces valeurs pour chacun des deux types de comportement grâce aux premières questions d'initialisation posées à l'utilisateur (nous verrons plus loin, dans la section **5.1** d'adaptation des questions posées, plus de détails). Les valeurs définissant chaque classe vont ensuite s'affiner au cours du temps grâce à l'algorithme de Machine Learning décrit juste après.
Nous allons ensuite pouvoir déterminer dans quel cas l'utilisateur se trouve en utilisant une formule de maximum a posteriori (on va déterminer de quel comportement il est le plus proche sachant les traces récoltées à ce tour). Nous allons pour cela utiliser un célèbre algorithme de Machine Learning : la méthode du plus proche voisin (single linkage) en classification supervisée (supervisée grâce aux premières questions d'initialisation) afin de classifier le comportement de l'utilisateur (parmi les classes sûr et hésite) et de mettre à jour les données caractérisant ces comportement à chaque nouvelle trace récoltée (en prenant, pour chaque classe et pour chaque critère, la moyenne de toutes les données classées dans cette classe).

□ Modéliser le comportement de l'utilisateur au niveau de sa vitesse de sélection. Pour cette partie, nous allons utiliser le "Keystroke level Model". Afin d'adapter le modèle, c'est-à-dire de le mettre à l'échelle, nous attribuerons un niveau à l'utilisateur (rapide, moyen, lent). Pour ce faire, lorsque l'utilisateur va lancer le jeu, nous allons calculer le nombre de temps qu'il met entre le clic dans le premier menu et celui dans le second. Nous pourrons ainsi déterminer si l'utilisateur est très familier avec la souris (et qu'il sélectionne rapidement) ou non en comparant le temps enregistré avec les temps prédits par le "Keystroke level Model" pour chaque niveau. Nous attribuerons à l'utilisateur le niveau dont le temps prédit est le plus proche du temps enregistré. Nous considérerons donc les valeurs du modèle adapté au niveau de l'utilisateur.
Ensuite, lorsque l'utilisateur répondra à une question à choix multiples (QCM), on comparera sa vitesse de clic réelle avec la vitesse de clic prédite par le modèle (en prenant en compte le niveau de l'utilisateur). On comparera donc la vitesse réelle avec la prédiction du modèle quant au temps de lecture plus le temps de réflexion plus la vitesse de pointage de la souris sur la réponse souhaitée plus le temps nécessaire pour effectuer un clic (presser et relâcher le bouton). Si la différence (vitesse réelle - vitesse prédite, pas en valeur absolue car si l'utilisateur met moins de temps que ce qu'avait prédit le modèle, il faut estimer qu'il était sûr de lui) entre ces deux temps est faible (c'est à dire inférieure à 15% du temps prédit), nous dirons, selon cette modélisation, que l'utilisateur est sûr de lui et si celle-ci est trop grande (supérieure à 15% du temps prédit) nous dirons que l'utilisateur hésite. Notons que nous avons choisi de caractériser la différence en prenant en compte un pourcentage du temps prédit afin de ne pas donner un temps de différence fixe qui aurait été plus difficile à atteindre pour un utilisateur novice que pour un expert. On est ainsi plus exigeant avec les personnes censées répondre rapidement. Considérons, par exemple, que la vitesse de sélection de l'utilisateur débutant est estimé à 3 secondes et que celle de l'utilisateur expert est estimé à 1 seconde. Dans ce cas, si l'on donnait un temps fixe de différence entre réalité et prédiction (disons 0.5 seconde) à respecter pour

estimer l'hésitation de l'apprenant, ce temps ne représenterait, proportionnellement au temps total mis pour la sélection, pas grand chose pour l'utilisateur lent (seulement 17% de son temps de sélection) alors qu'il représenterait 50% du temps de l'utilisateur rapide. Cette méthode favoriserait alors les utilisateurs rapides en leur donnant une grande marge et défavoriserait les utilisateurs lents.
De plus, à chaque réponse à laquelle l'utilisateur a correctement répondu et sans hésitation, nous testerons si le temps réel mis par l'utilisateur est nettement inférieur au temps prédit selon le niveau que l'on lui a attribué et le classerait ainsi dans un autre niveau (c'est à dire si l'utilisateur met un temps plus représentatif d'un niveau plus expert que celui que l'on lui a attribué). Si ce temps réel est inférieur au temps prédit d'un (ou de plusieurs) niveau(x) plus expert, nous attribuerons ce niveau plus expert (le plus expert dans le cas où il y en avait plusieurs) à l'utilisateur afin de ne pas le classer trop souvent dans la classe d'hésitation "sûr" (si l'utilisateur s'est fait passé pour un novice alors qu'il est expert par exemple).
Notons que le temps où l'apprenant ne regarde plus l'écran ne sera pas pris en compte dans le temps réel de vitesse de sélection. En effet, si l'utilisateur détourne le regard de l'écran, on arrête le chronomètre de vitesse de clic puisque l'on considère que l'utilisateur fait autre chose s'il ne regarde pas l'écran et que ce temps ne fait pas partie de son temps de réflexion (l'utilisateur est distrait). On reprendra le timer lorsque l'utilisateur regardera à nouveau l'écran.

- ☐ Modéliser le comportement de l'utilisateur au niveau de sa vitesse d'entrée de texte au clavier (en fonction du nombre de caractères entrés bien sûr et en suivant, de nouveau le "Keystroke level Model"). Afin d'adapter le modèle, c'est-à-dire de le mettre à l'échelle, nous attribuerons un niveau à l'utilisateur (bon dactylographe, dactylographe moyennement qualifié, ou pire dactylographe qui est peu familier avec le clavier). Pour ce faire, lorsque l'utilisateur va lancer le jeu, nous allons lui demander d'entrer un mot au clavier (par exemple "apprendre") afin de calculer le nombre de temps qu'il met pour entrer ce mot. Nous pourrons ainsi déterminer si l'utilisateur est très familier avec le clavier (et qu'il écrit rapidement) ou non en comparant le temps enregistré avec les temps prédits par le "Keystroke level Model" pour chaque niveau. Nous attribuerons à l'utilisateur le niveau dont le temps prédit est le plus proche du temps enregistré. Nous considérerons donc les valeurs du modèle adapté au niveau de l'utilisateur.
  Ensuite, lorsque l'utilisateur répondra à une question (attention, ici on ne considère que les questions dans lesquelles on demande à l'utilisateur d'entrer la réponse au clavier, ce n'est pas un QCM !), on comparera sa vitesse d'entrée de texte réelle avec la vitesse d'entrée de texte prédite par le modèle (en prenant en compte le niveau de l'utilisateur). On comparera donc la vitesse réelle avec la prédiction du modèle quant au temps de lecture plus le temps de réflexion plus la vitesse à laquelle l'utilisateur déplace ses mains sur le clavier (on considère que l'utilisateur n'a pas besoin de cliquer sur la zone de texte et que lorsqu'il tape sur le clavier, sa réponse est entrée dans la barre de texte) plus le temps nécessaire pour taper chacune des lettre que l'utilisateur a tapé (on ne considère pas le nombre de lettres du mot attendu mais bien le nombre de lettres du mot que l'utilisateur a entré) plus le temps nécessaire pour appuyer sur la touche "entrée". Si la différence (vitesse réelle - vitesse prédite, pas en valeur absolue car si l'utilisateur met moins de temps que ce qu'avait prédit le modèle, il faut estimer qu'il était sûr de lui) entre ces deux temps est faible (c'est à dire inférieure à 15% du temps prédit), nous dirons, selon cette modélisation, que l'utilisateur est sûr de lui et si celle-ci est trop grande (supérieure à 15% du temps prédit) nous dirons que l'utilisateur hésite. Notons que nous avons choisi de caractériser la différence en prenant en compte un pourcentage du temps prédit afin de ne pas donner un temps de différence fixe qui aurait été plus difficile à atteindre pour un utilisateur novice que pour un expert. On est ainsi plus exigeant avec les personnes censées répondre rapidement. Considérons, par exemple, que la vitesse d'entrée de texte de l'utilisateur débutant est estimé à 3 secondes et que celle de l'utilisateur expert est estimé à 1 seconde. Dans ce cas, si l'on donnait un temps fixe de différence entre réalité et prédiction (disons 0.5 seconde) à respecter pour estimer l'hésitation de l'apprenant, ce temps ne représenterait, proportionnellement au temps total mis pour l'entrée de texte, pas grand chose pour l'utilisateur lent (seulement 17% de son temps d'entrée de texte) alors qu'il représenterait 50% du temps de l'utilisateur rapide. Cette méthode favoriserait alors les utilisateurs rapides en leur donnant une grande marge et défavoriserait les utilisateurs lents.
  De plus, à chaque réponse à laquelle l'utilisateur a correctement répondu et sans hésitation, nous testerons si le temps réel mis par l'utilisateur est nettement inférieur au temps prédit selon le niveau que l'on lui a attribué et le classerait ainsi dans un autre niveau (c'est à dire si l'utilisateur met un temps plus représentatif d'un niveau plus expert que celui que l'on lui a attribué). Si ce temps réel est inférieur au temps prédit d'un (ou de plusieurs) niveau(x) plus expert, nous attribuerons ce niveau plus expert (le plus expert dans le cas où il y en avait plusieurs) à l'utilisateur afin de ne pas le classer trop souvent dans la classe d'hésitation "sûr" (si l'utilisateur s'est fait passé pour un novice alors qu'il est expert par exemple).
  Notons que le temps où l'apprenant ne regarde plus l'écran ne sera pas pris en compte dans le temps réel de vitesse d'entrée de texte. En effet, si l'utilisateur détourne le regard de l'écran, on arrête le chronomètre d'entrée de texte puisque l'on considère que l'utilisateur fait autre chose s'il ne regarde pas l'écran et que ce temps ne fait pas partie de son temps de réflexion (l'utilisateur est distrait). On reprendra le timer lorsque

l'utilisateur regardera à nouveau l'écran. On pourrait dans ce cas ajouter au temps prédit la vitesse à laquelle l'utilisateur déplace ses mains sur le clavier si l'on considère que l'utilisateur enlève ses mains du clavier lorsqu'il ne regarde plus l'écran.

- □ Déterminer le niveau d'hésitation de l'utilisateur en fonction des modélisations ci-dessus. Rappelons que nous avons deux cas à traiter : soit la question posée était un QCM et dans ce cas, nous devons prendre en compte les classifications du comportement d'hésitation données par la vitesse de clic et l'oculomètre ; soit la question posée nécessitait une réponse entrée au clavier et dans ce cas, nous devons prendre en compte la classification donnée par la vitesse d'entrée de texte (nous évaluerons alors le niveau d'hésitation de l'utilisateur simplement par cette classification). Dans le premier cas, nous dirons que l'utilisateur hésite si les deux modèles prédisent qu'il hésite, et que l'utilisateur est sûr de lui si les deux modèles prédisent qu'il est effectivement sûr de lui. Lorsque le modèle oculaire estime que l'utilisateur hésite mais pas le modèle de vitesse de clic ou lorsque le modèle oculaire estime que l'utilisateur est sûr de lui mais pas le modèle de vitesse de clic, cela est plus délicat mais nous dirons que l'utilisateur hésite. Nous définirons cette estimation d'hésitation différemment (c'est-à-dire qu'on dira dans ce cas que l'utilisateur est sûr de lui) si un de ces cas de figures se produit lors de l'initialisation avec les premières questions alors que l'utilisateur était sûr de lui (nous verrons plus loin, dans la section **5.1** d'adaptation des questions posées, comment le savoir). Nous pourrons également affiner l'estimation d'hésitation de ces deux derniers cas grâce à des tests (il vaut peut-être mieux estimer l'hésitation de l'utilisateur par l'estimation donnée par le modèle de vitesse de clic s'il y a un conflit avec le comportement estimé par l'oculomètre).

- □ Implémenter des algorithmes permettant d'adapter les questions posées à l'apprenant, et le contenu des fiches proposées (en fonction de si l'utilisateur confond les : natures des mots, faux amis, mots proches dans le sens, mots proche dans l'écriture, mots proche dans la prononciation, etc.). Les détails d'implémentation de ces algorithmes sont présentés dans la section **5** intitulée Forme d'adaptation choisie.

# 3 Autre forme d'interaction : l'oculomètre

L'oculomètre va principalement permettre de déterminer si l'utilisateur hésite dans sa réponse ou non lorsqu'il fait face à un QCM, comme nous l'avons détaillé dans la section précédente, plus précisément dans le point "Modéliser le comportement oculaire de l'utilisateur afin de déterminer s'il fait face à une hésitation".

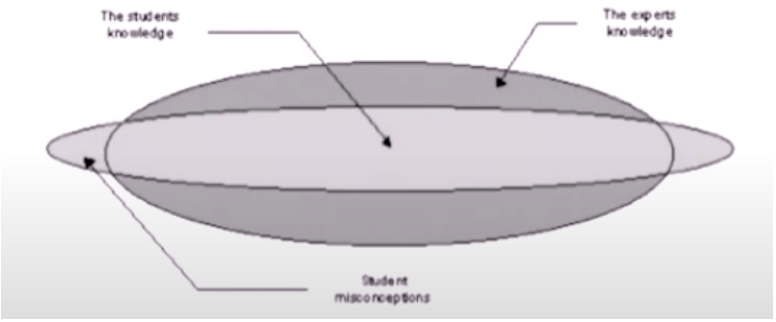
Il va également permettre de savoir si l'utilisateur regarde bien l'écran ou non. Si l'apprenant ne regarde plus l'écran, on va, comme décrit précédemment, arrêter le chronomètre de vitesse de clic ou de vitesse d'entrée de texte puisque l'on considère que l'utilisateur fait autre chose s'il ne regarde pas l'écran et que ce temps ne fait pas partie de son temps de réflexion (l'utilisateur est distrait).

# 4 Modélisation de l'utilisateur

Nous avons vu dans la section **2** intitulée Cahier des charges que l'on va modéliser plusieurs aspects de l'utilisateur. Nous allons, en effet, modéliser les connaissances de vocabulaire de l'utilisateur, les connaissances de l'utilisateur concernant la nature des mots en anglais, le comportement oculaire de l'utilisateur, le comportement de l'utilisateur au niveau de sa vitesse de sélection, et le comportement de l'utilisateur au niveau de sa vitesse d'entrée de texte au clavier. Nous avons décrit, en détails, le modèle (c'est-à-dire la structure et les calculs qui permettront de mettre à jour la modélisation) de chacune des modélisations précédemment citées dans la section **2** intitulée Cahier des charges.

Présentons la modélisation de l'utilisateur de façon moins détaillée (et plus abstraite) que vue précédemment. Nous allons mettre en place un modèle descriptif de l'apprenant dans le but de représenter des données historiques (tirées de ses anciennes réponses et anciens comportement de réponse au niveaux du regard et de la vitesse de sélection ou d'entrée de texte) de manière interprétable, de décrire et d'expliquer les phénomènes liés à l'apprentissage d'un apprenant (interprétation, explication des comportements) afin d'adapter les questions posées et les fiches proposées à l'apprenant, lui permettant d'améliorer plus efficacement ses connaissances et capacités. Notons tout d'abord que nous allons modéliser un apprenant sans prendre en compte la notion de groupe et que l'on va se concentrer sur la modélisation de ses connaissances et de ses comportements (et non pas de son savoir-faire, ses compétences ou autre). Nous allons devoir représenter les comportement d'hésitation de l'apprenant en fonction des traces de l'oculomètre et de sa vitesse de clic ou d'entrée de texte et nous mettrons à jour nos modèles à chaque réponse de l'apprenant, en prenant en comptes ces traces, l'estimation d'hésitation qui en est ressorti et la réponse que l'apprenant a effectivement sélectionnée, comme détaillé dans la section **2** intitulée Cahier des charges.

Nous utiliserons un modèle étendu pour représenter les connaissances de l'utilisateur en terme de mots de vocabulaire et de natures de mot puisque l'on a décidé de pouvoir faire un retour à l'utilisateur en fonction de connaissances qu'il aurait en plus de l'expert comme par exemple le vocabulaire français qui peut porter à confusion et donner de faux amis par rapport à l'anglais (de même, il peut confondre des natures de mots en anglais à cause des notions françaises). On considère que les connaissances de l'apprenant ne forment pas un sous ensemble des connaissances expertes puisque l'apprenant à des connaissances supplémentaires (en l'occurrence, le vocabulaire français) comme représenté sur le schéma ci-contre, les misconceptions étant les erreurs de l'apprenant dues à ses connaissances en français :




Nous allons utiliser une modélisation symbolique pour représenter les connaissances des natures des mots. En effet, les liens entre les différentes classes de mots (nom commun, adjectif, verbe, etc.) vont être décrits de façon symbolique grâce au model tracing (représentation sous forme de règles de production qui nécessite d'expliciter toutes les règles permettant de décrire les connaissances dont on a besoin pour déterminer la nature d'un mot) et vont permettre de voir si l'apprenant maîtrise ces notions (grâce à un raisonnement déductif). Nous avons choisi cette approche car les stratégies de résolutions sont bien définies et que l'on souhaite une rétroaction (concernant la nature des mots) compréhensive et immédiate.

Puis, nous allons également modéliser les comportements de l'utilisateur au niveau du mouvement de ses yeux, de sa vitesse de clic et d'entrée de texte de la même manière en établissant des règles de classification (les classes étant "utilisateur sûr" et "utilisateur hésitant") déterminant si l'utilisateur à répondu au hasard ou non.

Tous les détails se trouvent dans la section **2** intitulée Cahier des charges.

# 5 Forme d'adaptation choisie

Notre projet comporte deux formes d'adaptation : l'adaptation des questions posées à l'utilisateur et l'adaptation du contenu de la fiche d'aide proposée à l'utilisateur.

Les modélisations de comportement de l'utilisateur (oculaire et au niveau de la vitesse de clic et d'entrée de texte) vont permettre de classifier l'hésitation de l'utilisateur et ainsi d'adapter la probabilité d'acquisition donnée à chacun des mots de vocabulaire, comme vu précédemment. Ces probabilités vont ensuite permettre de déterminer les questions à poser à l'apprenant les plus pertinentes. Nous détaillons les mécanismes informatiques pour l'adaptation dans la section suivante.

## 5.1 Adaptation des questions posées

Initialement, on commence par poser des questions à l'utilisateur afin de cibler son niveau, de voir s'il sait bien faire la différence entre deux mots de la même famille mais de nature différente et afin de commencer à mettre en place des reconnaissances de ses comportement (comportement oculaire, vitesse de clic et d'entrée de texte). Pour cela, on va demander à l'utilisateur de répondre à une question de vocabulaire en QCM (on lui proposera un QCM différent jusqu'à ce qu'il donne la bonne réponse) puis on lui reposera la même question mais sous forme d'écriture complète, on va ainsi pouvoir déterminer si l'utilisateur hésitait lors du QCM (c'est le cas s'il n'entre pas la bonne réponse au clavier) ou s'il connaissait vraiment la réponse. Nous répéterons ce processus jusqu'à avoir rencontré au moins deux fois chacun des deux cas (l'utilisateur entre le bon mot au clavier ou pas) afin de pouvoir avoir une initialisation représentative. Cette première manipulation de l'application va permettre de calibrer la vitesse de clic de l'utilisateur ainsi que sa vitesse d'entrée de texte en considérant évidemment les deux cas de figure à savoir : l'utilisateur connaît la réponse ou l'utilisateur hésite. Elle permettra également de classifier les données récoltées par l'oculomètre en fonction des cas. De plus, on demandera à l'utilisateur de traduire deux mots de la même famille mais de nature différente afin de déterminer s'il fait bien la distinction entre les deux.

Après cette initialisation, il va falloir poser des questions pertinentes à l'utilisateur afin qu'il puisse améliorer ses connaissances sans pour autant le noyer d'informations. En effet, le but n'est pas d'avoir des valeurs différentes de 0 dans toute la base de connaissance le plus vite possible (auquel cas, on poserait des questions sur des mots

différents à chaque fois) mais plutôt de privilégier l'ancrage des connaissances profondément dans la mémoire de l'utilisateur (de sorte à ce qu'il n'oublie pas facilement ce qu'il a appris en utilisant notre logiciel).

En suivant la "Power Law Of Practice", la probabilité d'acquisition de chacun des mots va baisser au cours du temps. Notons que la courbe de décroissance considérée dépendra du nombre de fois où l'utilisateur à répondu à ce mot de vocabulaire. En effet, comme on le voit dans la "Power Law Of Practice", lors des premières apparitions du mot, il est préférable de faire un rappel plus fréquemment que lorsque l'on a déjà vu le mot plusieurs fois. De surcroît, plus le mot a été rencontré, plus on demandera à l'apprenant d'entrer le mot au clavier plutôt que de lui proposer un QCM.

Notre algorithme permettra de trouver un compromis entre approfondissement (apprentissage de nouveau mots) et renforcement (ancrer les mots déjà rencontrés dans la mémoire). La proportion de mot non encore rencontré va évoluer au cours du temps. En effet, il est préférable d'apprendre beaucoup de nouveaux mots lors de la première utilisation de l'application puis, au fur et à mesure, de privilégier le renforcement, notamment lorsque la base de connaissance de l'utilisateur commence à être très grande. Au début, notre algorithme va donc, après les premiers mots d'initialisation, alterner un mot déjà rencontré et un mot non encore rencontré puis cette proportion va baisser jusqu'à atteindre une question sur un mot non encore rencontré tous les dix mots. Ces valeurs évolueront en fonction des performances de l'apprenant. En effet, si l'utilisateur a déjà du mal avec la base de mot qu'il a déjà rencontré, on va préférer renforcer ses connaissances pendant un temps. Notre algorithme prendra donc un paramètre déterminant le nombre de questions sur des mots déjà rencontrés avant une nouvelle question (et donc initialisé à 1). Ce paramètre augmentera dès que l'utilisateur se trompe sur un mot qu'il a déjà rencontré et si ce paramètre atteint la valeur 11, on effectuera un renforcement de connaissances jusqu'à ce que l'apprenant donne 5 bonnes réponses consécutives.

Notons que, concernant les nouveaux mots proposés à l'utilisateur, nous avons deux choix : soit on lui propose un mot proche du lexique qu'il a déjà rencontré afin qu'il puisse lier les notions, soit on lui propose de nouveaux mots très différents afin d'élargir sa palette de connaissances. Étant donné que ces deux options semblent cohérentes, nous avons décider de trouver un compromis en choisissant le nouveau mot aléatoirement dans la base de donnée afin qu'il puisse bénéficier des deux avantages précédemment cités. Nous pouvons tout de même prendre en compte la nature du mot proposé et poser une question sur un mot dont la nature pose problème à l'utilisateur (c'est-à-dire dont la nature a le score (la somme sur sa ligne de score) le plus faible dans la représentation des natures des mots).

Il reste un point à éclaircir : comment choisir sur quel mot déjà rencontré l'utilisateur va-t-il être interrogé. Pour cela notre algorithme va choisir, parmi les mots déjà rencontrés (c'est à dire les mots dont la valeur désignant le nombre de fois où le mot lui a été demandé est supérieure strictement à 0 dans la représentation des mots de vocabulaire) celui qui a la probabilité d'acquisition la plus basse. S'il y en a plusieurs à égalité, nous choisirons le mot qui a été rencontré le moins de fois (afin de renforcer son ancrage) et si tous les mots ont une valeur supérieure à 0.75, nous allons choisir exceptionnellement un nouveau mot (nous considérons que l'utilisateur connaît assez bien tous les mots sur lesquels il a été interrogé).

## 5.2 Adaptation du contenu de la fiche d'aide

Tout d'abord, au niveau de l'interface, notons qu'une fiche d'aide est toujours proposée et accessible par un clic. Après chaque question, l'utilisateur peut choisir de consulter cette fiche avant de continuer. On peut alors prendre en compte le fait qu'il a consulté la fiche ou non pour estimer l'hésitation de l'utilisateur lors de sa réponse (et ainsi mettre à jour les probabilités d'acquisition du mot) et pour mettre à jour les caractéristiques des classes d'hésitation vues plus haut (notamment dans le cadre d'un QCM, si l'une des deux modélisations (oculomètre ou vitesse de sélection) indique que l'utilisateur est sûr de lui et que l'autre indique que l'utilisateur hésite, si l'utilisateur consulte la fiche, on considérera qu'il a hésité avant de répondre et on pourra alors classer son hésitation à la question, les données récoltées par l'oculomètre et par sa vitesse de réponse dans la classe "hésite").

Questionnons-nous désormais sur le contenu de la fiche d'aide. L'apprenant peut avoir besoin d'aide pour plusieurs raisons, il peut confondre les : natures des mots, faux amis, mots proches dans le sens, mots proche dans l'écriture, mots proche dans la prononciation, etc. Si notre système n'a pas d'information sur les lacunes de l'utilisateur, la fiche va simplement proposer une définition du mot et un exemple d'utilisation (une phrase où le mot serait dans son contexte). De plus, si l'utilisateur sélectionne ou entre une mauvaise réponse, on indiquera dans la fiche également la traduction (et peut-être même la définition) du mot qu'il a entré ou sélectionné afin d'aider l'utilisateur à faire la différence. Sinon, en plus de ces deux (ou trois) éléments, la fiche d'aide va présenter, selon les lacunes de l'utilisateur détectées par le système :

-les différents suffixes en fonction des natures des mots (on a vu plus haut comment identifier que l'utilisateur confondait des natures de mot)

-la traduction (et peut-être même la définition) des faux amis : si l'utilisateur entre un mot en anglais proche du mot demandé en français, on va lui donner la traduction (et peut-être même la définition) de chacun des deux mots pour qu'il puisse faire la différence (par exemple, si on demande à l'utilisateur de traduire le verbe

achever (to complete) et qu'il répond to achieve (réaliser), la fiche va présenter chacun des deux mots avec leur traduction et définition respective)
On pourrait également aider l'utilisateur s'il confond deux mots proches dans le sens, dans l'écriture, ou dans la prononciation en donnant la traduction (et peut-être même la définition) des deux mots, de la même manière qu'avec les faux amis.

# 6 Format et système de collecte de traces

Tout d'abord, rappelons que nous devons récolter plusieurs traces : les traces oculaires, les traces au niveau de la vitesse de sélection, et les traces au niveau de la vitesse d'entrée de texte.

Les traces de mouvement des yeux vont être récoltées à l'aide d'un oculomètre. Comme nous l'avons vu dans le cahier des charges, les données conservées vont être la différence maximum de temps passé sur le mot entre deux mots (c'est à dire un flottant) et le nombre de fois total où l'utilisateur a bougé son regard sur un autre mot divisé par le nombre de propositions (c'est à dire un flottant). Chaque donnée récoltée par l'oculomètre va donc être un point à deux dimension dont chaque coordonnée est flottante.

Les traces au niveau de la vitesse de sélection vont être récoltées grâce à un chronomètre qui débutera lorsque la question apparaîtra à l'écran et qui s'arrêtera lorsque l'utilisateur cliquera sur une réponse (avec des pauses dans le chronomètre si l'apprenant détourne le regard de l'écran). Ces traces seront donc des flottants représentant un temps de réponse.

Les traces au niveau de la vitesse d'entrée de texte vont être récoltées grâce à un chronomètre qui débutera lorsque la question apparaîtra à l'écran et qui s'arrêtera lorsque l'utilisateur appuiera sur la touche "entrée" (avec des pauses dans le chronomètre si l'apprenant détourne le regard de l'écran). Ces traces seront donc des flottants représentant un temps de réponse. Il faudra également sauvegarder la taille du mot (en nombre de caractères) entré par l'utilisateur afin de pouvoir estimer le temps d'entrée de texte avec le "Keystroke level Model".

Tous les détails concernant le format, le système de collecte ou encore l'analyse des traces se trouvent dans la section **2** intitulée Cahier des charges.