Gérer et exploiter des connaissances produites par une communauté en ligne – Application au raisonnement à partir de cas

Emmanuelle Gaillard

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Gérer et exploiter des connaissances produites par une communauté en ligne

Application au raisonnement à partir de cas

THÈSE

présentée et soutenue publiquement le 22 juin 2016

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Emmanuelle Gaillard

Composition du jury

Rapporteurs :	Sylvie Despres	Professeure à l'Université Paris 13
	Alain Mille	Professeur émérite à Université Claude Bernard Lyon 1
Examinateurs :	Anne Boyer	Professeure à l'Université de Lorraine
	Patrick Gallinari	Professeur à l'Université Paris 6
Directeur de thèse :	Jean Lieber	Maître de conférences HDR à l'Université de Lorraine
Encadrant :	Emmanuel Nauer	Maître de conférences à l'Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Je tiens à remercier les membres de mon jury d'avoir accepté d'en faire partie et de s'être investis de la sorte. Merci à Anne Boyer d'avoir accepté de présider ce jury et d'avoir été ma référente tout au long de cette thèse. Merci à Sylvie Despres et Patrick Gallinari d'en avoir été les rapporteurs et d'avoir fait part d'enrichissantes remarques. Merci à Alain Mille et son débat sur Pitrat.

Je souhaite remercier tout particulièrement mon directeur de thèse Jean Lieber et mon co-encadrant Emmanuel Nauer qui, au fil de cette thèse, sont devenus plus que des encadrants mais des amis. Un grand merci à Jean pour la rigueur scientifique et la richesse théorique qu'il m'a enseignées. Je suis également reconnaissante du temps consacré à la transmission de sa culture musicale (pas vraiment réussie), en espérant que ma descendance fasse mieux. Je remercie chaleureusement Emmanuel pour son soutien, la quantité de temps investi durant les longs débats amenant à des remaniements, mais aussi pour les nombreuses relectures de ma thèse. J'espère avoir fait honneur au statut de première thésarde. Merci également pour les discussions et les échanges d'opinions autour de sujets variés.

J'adresse mes remerciements aux membres de l'équipe Orpailleur avec qui j'ai eu le plaisir de travailler ainsi qu'Amedeo Napoli de m'avoir accueillie au sein de cette équipe. Je remercie également toutes les personnes avec qui j'ai pu partager scientifiquement et amicalement au LORIA ou en conférences, notamment Alice, Laura, Fadi, Felipe, Jérémie, Julien, Matthieu, Stéphane, Valmi, Yannick et bien d'autres.

Merci à Yannick Naudet et Younes Djaghloul de m'avoir orientée vers ce sujet de thèse et de m'avoir accueillie au sein du CRP Henri Tudor.

Je souhaite aussi remercier la communauté internationale de raisonnement à partir de cas de m'avoir conseillée lors de mes présentations et notamment David Wilson lors du *Doctoral Consortium*.

Un grand merci à l'IUT de Nancy Charlemagne et notamment Laurent Dupont et Laurent Andrey de m'avoir fait confiance en me donnant des responsabilités. Merci également à l'IUT de Saint-Dié-des-Vosges et ses enseignants, spécialement Jonathan Weber et Jean-Luc Husson, de m'avoir chaleureusement accueillie en me permettant de finir ma thèse.

Merci aux enseignants du master de sciences cognitives et aux amis que j'ai rencontrés durant mes études et qui m'ont permis de m'épanouir dans ce que je faisais. Je pense particulièrement à Emilie Bigot et Nicolas Renoir Nosal.

Je remercie également Amélie Cordier pour son aide précieuse lors de mes évaluations ainsi que pour les corrections de mon accent lors de mes répétitions en anglais, mais aussi pour son amitié.

Merci à Valmi Dufour-Lussier pour sa « patience » lors de séances de débogages, pour ses partages enthousiastes quant à ses opinions sur les français et ses analyses psychologiques profondes.

Je tiens à remercier tendrement ma mère et mon père d'avoir cru en moi et de m'avoir inculqué la persévérance et l'optimisme, ce en quoi je n'aurais pas pu finir ni même commencer cette thèse. Merci également à mon frère d'avoir tout simplement toujours été présent. Merci à mon fils Aleko de m'avoir inspirée avant même d'être venu au monde et de m'avoir laissée finir ma thèse à temps. Enfin, merci à mon compagnon Lubo qui m'a soutenue, a sacrifié week-ends et soirées et a subi les nombreuses nuits blanches à travailler.

À mon compagnon et mon fils, mon nouvel équilibre.

Sommaire

Table des figures

Liste des tableaux

Introduction

Tom veut faire un dessert aux fruits à son fils Eliot qu'il garde pour l'après-midi. Tom veut utiliser la bouteille de lait qu'il a dans son réfrigérateur. De plus, Tom ne veut utiliser que les fruits qu'il a chez lui : des bananes et/ou des abricots. Il décide de réaliser une recette de clafoutis qu'il a trouvé sur un site de cuisine en ligne qui contient du lait. Cette recette a été partagée par une utilisatrice qui a l'habitude de proposer de bonnes recettes. Cependant, la recette contient des cerises que Tom n'a pas chez lui.

Un commentaire apporté par Pierre sur la recette, un utilisateur du site, indique que les cerises peuvent être remplacées par des bananes. Néanmoins Tom n'a pas confiance en ce commentaire ; il pense que les bananes rendraient le clafoutis trop pâteux. Il préfère suivre un autre commentaire apporté par Mercotte, une utilisatrice qui indique que les cerises peuvent être remplacées par un autre fruit à noyau comme l'abricot.

Tom va réaliser sa recette de clafoutis en remplaçant les cerises par des abricots. Eliot va pouvoir manger ce succulent clafoutis et se régaler !

Ce scénario est typique de la rencontre entre deux domaines : le raisonnement à partir de cas (RÀPC) qui est un paradigme de résolution de problèmes par analogie [Riesbeck et Schank, 1889] et l'utilisation de connaissances provenant d'utilisateurs du Web. Le RÀPC utilise des expériences passées similaires à un nouveau problème et les adapte pour résoudre ce nouveau problème. Dans ce scénario, les expériences sont des recettes de cuisine apportées par les utilisateurs d'un site de cuisine en ligne. Les contraintes du problème sont de réaliser un dessert avec du lait et des fruits qui peuvent être des bananes ou des abricots. Le site de cuisine en ligne ne contient pas de recettes qui satisfassent ces contraintes mais il existe une recette similaire, créée par une utilisatrice qui a une bonne réputation. Cette recette similaire est une recette de clafoutis contenant du lait et des fruits qui sont des cerises — et non pas des bananes ou des abricots. Pour satisfaire totalement les contraintes du problème (c'est-à-dire, avoir une recette contenant comme seuls fruits des bananes ou des abricots), Tom a le choix entre deux adaptations proposées par des utilisateurs du site. Tom préfère ne pas utiliser la première adaptation car il ne croit pas qu'adapter le clafoutis en remplaçant la cerise par de la banane produise un bon résultat (ou, du moins, un résultat qui lui plaise). En revanche, il est enclin à la deuxième adaptation car il croit que remplacer la cerise par un autre fruit à noyau comme l'abricot est une bonne adaptation. Finalement, Tom adapte la recette de clafoutis en substituant les cerises par des abricots.

Le raisonnement à partir de cas

Le raisonnement à partir de cas (RÀPC) est un domaine de l'intelligence artificielle, et plus spécifiquement de la résolution de problèmes. Les fondements du RÀPC proviennent des théories sur la compréhension de la pensée humaine et notamment de la notion de script introduite par [Schank et Abelson, 1977]. Un script est une structure en mémoire qui décrit une situation stéréotype de la vie quotidienne afin de servir de référence pour comprendre une nouvelle situation et agir de façon adaptée. Lorsqu'une personne rencontre une nouvelle situation, elle sélectionne dans sa mémoire un script qui est pertinent pour aborder la nouvelle situation. Le script sélectionné est ensuite adapté, pour correspondre exactement aux besoins de la nouvelle situation, avec des connaissances déclaratives et procédurales précédemment acquises.

Fondé sur ce principe, le RÀPC sélectionne dans une base de cas, un cas étant une expérience passée, le, éventuellement les, cas le(s) plus similaire(s) au nouveau problème grâce à des connaissances de similarité. Dans cette phase, qui est appelée phase de remémoration, le cas le plus similaire est appelé cas source tandis que le nouveau problème est appelé cas cible. Le cas source est ensuite adapté pour résoudre le problème cible en s'appuyant sur des connaissances du domaine et des connaissances d'adaptation ; cette phase correspond à la phase d'adaptation. Si le cas source adapté est considéré comme une bonne solution pour l'utilisateur, il peut être retenu dans la base de cas comme un nouveau cas ; cela correspond à la phase de mémorisation. D'autres phases existent et seront présentées.

Dans l'exemple précédent lié au domaine culinaire, le problème cible est de trouver une recette de dessert au lait et aux fruits qui sont des bananes ou des abricots, et le cas source remémoré est la recette de clafoutis. Une solution du problème cible doit satisfaire trois contraintes imposées par le problème cible : être une recette de dessert, contenir du lait et contenir des fruits qui sont des bananes ou des abricots. Les connaissances utilisées pour remémorer et adapter le cas source en satisfaisant les contraintes du problème cible sont (1) les connaissances liées au cas source (type de plat produit par la recette, ingrédients contenus), (2) le fait que la cerise, la banane et l'abricot sont des fruits, et que la cerise et l'abricot sont plus particulièrement des fruits à noyau, (3) les connaissances de similarité entre le cas source et le problème cible (un problème cible demandant des fruits est similaire à un cas source contenant des fruits), et (4) les connaissances d'adaptation qui disent de remplacer les cerises par des bananes ou des abricots. Comme la recette adaptée de clafoutis avec du lait et des abricots est une bonne recette pour Tom, alors elle sera stockée comme un nouveau cas dans la base de cas et pourra être utilisée ultérieurement lors de la résolution d'un nouveau problème.

À l'instar de la plupart des systèmes de RÀPC classiques, cet exemple exploite des unités de connaissances (UC) d'une base de connaissances qui sont habituellement organisées dans quatre conteneurs de connaissances [Richter, 1995] : la base de cas, les connaissances du domaine, les connaissances de similarité et les connaissances d'adaptation. Un système de RÀPC prend en entrée un problème cible (par exemple, « Je veux un dessert contenant du lait et des fruits qui sont des bananes ou des abricots. ») et retourne des réponses calculées par le moteur du système de RÀPC qui utilise les quatre conteneurs de connaissances pour raisonner.

Dans certains systèmes de RÀPC, les connaissances du domaine sont représentées à l'aide d'une ontologie du domaine. Une ontologie du domaine est une organisation formelle d'un ensemble de classes d'un domaine spécifique, reliées entre elles par des relations. La plus utilisée des relations est une relation hiérarchique de généralisation-spécialisation appelée également relation de subsomption. Par exemple, dans l'ontologie du domaine de la cuisine, on peut traduire la proposition « L'abricot est un fruit à noyau » par le fait que la classe des abricots est plus spécifique que (c'est-à-dire, la classe des abricots est subsumée par) la classe des fruits à noyau. La

classe la plus spécifique (subsumée) est aussi appelée *sous-classe* tandis que la classe générique (subsumante) est appelée *superclasse*. Dans l'exemple, la classe des fruits à noyau est une superclasse de la classe des abricots. Dans un système de RÀPC, la formalisation des connaissances du domaine via une ontologie peut avoir plusieurs utilités. L'ontologie du domaine permet de définir les classes pour représenter les cas de la base de cas et les problèmes cibles. De plus, le calcul de la similarité durant la phase de remémoration entre le cas source et le problème cible peut être guidé par les relations de subsomption dans l'ontologie. Dans l'exemple précédent, comme le site de cuisine ne contient pas de recettes qui satisfassent complètement le problème cible, Tom doit trouver une recette maximisant la satisfaction de chacune des contraintes du problème cible. Par exemple, comme l'ontologie du domaine culinaire définit la connaissance « L'abricot est un fruit à noyau » alors la contrainte du problème qui consiste à rechercher une recette contenant des bananes ou des abricots peut-être relâchée pour rechercher une recette contenant des bananes ou des fruits à noyaux. Par ailleurs, comme l'ontologie du domaine culinaire définit aussi la connaissance « La cerise est un fruit à noyau », alors la recette de clafoutis de Tom peut être retrouvée lors d'une recherche de recette contenant des fruits qui sont des bananes ou des fruits à noyau. Par conséquent, la structure de l'ontologie du domaine et notamment les relations de subsomption permettent de guider la phase de remémoration du système de RÀPC.

Dans une ontologie classique, la relation de subsomption entre une classe et sa superclasse dépend de propriétés nécessaires et suffisantes partagées par les instances de la sous-classe. Cependant, ce type de classification n'est pas flexible et ne permet pas de représenter l'aspect nuancé ou encore graduel de la représentation mentale des connaissances. Considérons par exemple la classe des alcools forts définie par la propriété nécessaire et suffisante d'avoir un taux d'alcool minimum de 18% [1]. La classe des malibus (21% d'alcool) sera considérée comme une sous-classe des alcools forts alors que ça ne sera pas le cas pour la classe des martinis (16% d'alcool). De plus, aucune distinction ne sera faite entre la classe des tequilas (35% d'alcool) et la classe des whiskys (généralement entre 40% d'alcool et 50% d'alcool) au regard de la superclasse des alcools forts. Héléne Rosch propose une solution permettant de représenter l'aspect graduel des classes en introduisant la notion de prototype [Rosch, 1973]. Le prototype d'une classe est une sous-classe qui représente au mieux cette classe. Plus une sous-classe est similaire au prototype de sa classe, plus elle sera typique de sa classe. La représentation de la typicalité des sous-classes d'une même superclasse permet d'ordonner les sous-classes par rapport à la superclasse. Plus une sous-classe est typique de sa superclasse plus cette sous-classe est similaire au prototype de la superclasse. Par exemple, la classe des whiskys peut être considérée comme plus typique de la classe des alcools forts que la classe des tequilas. Par ailleurs, les classes atypiques (c'est-à-dire, les « moins » typiques) sont les classes à la frontière de la superclasse. Par exemple, la classe des martinis (16% d'alcool) pourra être considérée comme une sous-classe de la superclasse des alcools forts mais être une classe atypique. Ainsi, les classes atypiques d'une superclasse permettent de rendre flexibles les frontières de leur superclasse.

Problématiques

Dans les systèmes de RÀPC, l'acquisition des UC des quatre conteneurs se fait généralement à l'aide d'un expert du domaine afin de garantir la validité et la qualité des connaissances. Un des avantages des systèmes de RÀPC est la réutilisation de cas passés qui ont précédemment été mémorisés, ce qui permet de réduire l'effort d'acquisition de l'expert diminuant le problème du goulot d'étranglement [Feigenbaum, 1984]. Cependant, l'acquisition des autres conteneurs de

1. `https://www.service-public.fr/professionnels-entreprises/vosdroits/F22379`

connaissances pour un système de RÀPC demande un long travail de la part de l'expert, et la mise à jour de ces connaissances est coûteuse.

Une solution possible pour réduire l'effort d'acquisition des connaissances dans les conteneurs de connaissances est d'utiliser des processus d'extraction de connaissances. L'extraction de connaissances peut se faire lors de la construction des conteneurs de connaissances par de la fouille de textes ([Ceausu et Desprès, 2005]), par exemple, ou encore par de la fouille de données sur les connaissances existantes pour extraire des connaissances d'adaptation ou du domaine ([Gaillard *et al.*, 2011]). [Ceausu et Desprès, 2005] proposent de faciliter la construction d'une ressource terminologique à partir de textes correspondant à des procès verbaux d'accidents en extrayant les termes du domaine et les relations entre termes. [Gaillard *et al.*, 2011] proposent d'extraire des connaissances d'adaptation dans le domaine culinaire en prenant en compte des incompatibilités d'ingrédients ou des ingrédients souvent associés dans des recettes. L'extraction peut aussi se faire de façon opportuniste lorsqu'une adaptation a échoué. Par exemple, [Cordier *et al.*, 2008,Badra *et al.*, 2009] proposent d'acquérir, auprès d'utilisateurs, les connaissances permettant de réparer une adaptation qui a échoué. En effet, les retours des utilisateurs du système de RÀPC sur les causes de l'échec du système peuvent être sauvegardées sous forme de connaissances d'adaptation. Cette solution permet de diminuer le temps d'acquisition de connaissances mais implique d'utiliser différents processus d'extraction pour les différents conteneurs de connaissances. De plus, cette solution impose la présence d'un expert indispensable pour valider les connaissances dont la qualité peut être variable, et le problème de la mise à jour des connaissances est toujours présent.

Une autre solution, choisie dans ce travail, est d'utiliser une communauté en ligne (*e-community* en anglais), aussi appelée communauté virtuelle, qui correspond à un groupe de personnes partageant un même domaine d'intérêt et qui interagissent entre elles par voie électronique par le biais de courriels, de forums, de sites collaboratifs, etc. Par exemple, les utilisateurs (producteurs et/ou lecteurs) du site de cuisine consulté par Tom forment une communauté en ligne et ces utilisateurs interagissent à propos de recettes de cuisine ou de connaissances d'adaptation culinaire. Utiliser une communauté en ligne pour acquérir les connaissances permet de remplacer l'expert ou restreindre notablement son travail.

Un bénéfice de l'acquisition de connaissances par une communauté en ligne est d'obtenir une quantité importante d'UC de façon peu coûteuse en un temps « restreint ». En outre, les interactions de la communauté en ligne permettent une mise à jour continue des connaissances, ce qui est moins coûteux qu'avec un expert ou via une méthode d'extraction de connaissances. Cependant, l'acquisition des connaissances par une communauté en ligne entraîne certains biais qu'il est nécessaire de contrôler. En effet, cette méthode d'acquisition est influencée par des facteurs humains tels que le niveau d'expertise et le sérieux des utilisateurs de la communauté en ligne, ou encore par des avis différents sur les connaissances produites :

- Le niveau d'expertise ou le sérieux des utilisateurs influencent directement la qualité des connaissances produites. Ainsi, l'opinion personnelle ou la croyance d'un utilisateur du système de RÀPC à propos des UC peuvent varier. Par exemple, la connaissance d'adaptation affirmant que la cerise peut être substituée par la banane dans le clafoutis choisi par Tom a été apportée par un utilisateur avec un plus faible niveau d'expertise que l'utilisateur ayant affirmé que la cerise peut être substituée par de l'abricot ; la croyance de Tom envers la première connaissance d'adaptation est donc plus faible que celle envers la deuxième connaissance d'adaptation. Le problème de la fiabilité des UC se pose alors lorsqu'un système de RÀPC exploite des UC provenant d'une communauté en ligne. En effet, en utilisant des UC provenant d'une communauté en ligne, un système de RÀPC utilise des UC plus ou moins fiables. Pour un utilisateur, la satisfaction des réponses qu'un système de RÀPC

retourne dépend des UC utilisées dans le moteur du système. L'utilisation d'une UC non fiable dans le moteur d'un système entraîne une faible qualité des réponses retournées par ce système. Ainsi, si Tom interroge un système de RÀPC pour obtenir « un dessert au lait et aux fruits qui sont des bananes ou des abricots », et que le système lui renvoie une recette adaptée du clafoutis contenant des bananes alors Tom ne sera pas satisfait par cette réponse.

– Acquérir plusieurs avis à propos d'une UC provenant d'une communauté en ligne permet de rendre compte de l'aspect graduel de la représentation mentale des utilisateurs. Ce phénomène peut se traduire par la représentation de nuances dans l'acquisition des relations de subsomption de l'ontologie du domaine. Représenter la gradualité d'une relation de subsomption peut consister à représenter la typicalité entre une sous-classe et sa superclasse. Par exemple, acquérir différents avis de la communauté en ligne sur les propositions « Tequila est un bon exemple d'alcool fort » et « Whisky est un bon exemple d'alcool fort », permet de distinguer entre elles la classe des tequilas et la classe des whiskys vis-à-vis de leur superclasse qui est la classe des alcools forts.

Approches proposées dans la thèse

Gestion de la fiabilité des connaissances

Une contrainte lorsqu'un système de RÀPC utilise des connaissances provenant d'une communauté en ligne est de gérer la fiabilité de ces connaissances. En effet, lorsqu'un utilisateur interroge le système de RÀPC, la fiabilité des connaissances utilisées par le moteur du système a un impact direct sur la qualité des réponses retournées par le système. Utiliser des connaissances non fiables dans le raisonnement fait par le moteur du système de RÀPC peut impliquer de mauvaises inférences et ainsi produire des réponses de mauvaise qualité.

Contrôler la fiabilité des UC apportées par la communauté en ligne peut se faire en collectant des connaissances, appelées métaconnaissances, sur ces mêmes UC. Certaines métaconnaissances peuvent être accessibles de par le fonctionnement même de la communauté en ligne. Dans un forum ou sur un site collaboratif, les utilisateurs doivent très souvent s'inscrire pour interagir : l'auteur d'une UC est ainsi connu. D'autres métaconnaissances peuvent provenir d'interactions des utilisateurs de la communauté en ligne, comme les évaluations relatives à la véracité d'une UC ou au degré d'expertise d'un utilisateur. Grâce aux métaconnaissances, les facteurs qui influencent la fiabilité d'une UC peuvent ainsi être contrôlés et mesurés afin de gérer la fiabilité de cette UC utilisée par le système de RÀPC.

Notre approche propose de construire un modèle de métaconnaissances nommé MKM (*Meta-Knowledge Model*) permettant de représenter la fiabilité des UC apportées par une communauté en ligne. MKM permet de représenter la fiabilité de chaque UC au regard de chaque utilisateur de la communauté en ligne. De plus, MKM est utilisé pour étendre un système de RÀPC exploitant des UC provenant d'une communauté en ligne afin de gérer la fiabilité des UC. L'extension d'un système de RÀPC avec MKM a deux buts principaux :

1. Filtrer les UC non fiables afin qu'elles ne soient pas utilisées par le système de RÀPC.
2. Classer les réponses retournées par le système de RÀPC grâce à la fiabilité des UC impliquées dans les réponses.

La fiabilité d'une UC pour un utilisateur de la communauté en ligne est interprétée comme la représentation de l'utilité et de l'exactitude de cette UC pour l'utilisateur. Dans MKM, la fiabilité d'une UC pour un utilisateur est dépendante de :

- métaconnaissances provenant d'interactions des utilisateurs de la communauté en ligne dans le système lorsqu'un utilisateur produit une UC, qu'il évalue une UC correspondant à sa croyance envers cette UC, ou qu'il évalue un autre utilisateur correspondant à sa confiance envers cet utilisateur.
- métaconnaissances calculées par le système qui sont la qualité d'une UC, la confiance d'un utilisateur envers un autre utilisateur et la réputation d'un utilisateur dans la communauté en ligne.

La contribution de la représentation de la fiabilité par MKM a fait l'objet de deux publications, une à la conférence internationale de RÀPC (ICCBR) en 2013 [Gaillard *et al.*, 2013a] et l'autre à la conférence nationale d'ingénierie des connaissances (IC) en 2013 [Gaillard *et al.*, 2013b].

La fiabilité permet de filtrer les UC non fiables et de n'utiliser que les UC les plus fiables dans le système de RÀPC. Finalement, les réponses retournées par le système de RÀPC sont classées par rapport à un score qui prend en compte la fiabilité des UC impliquées dans chacune des réponses pour l'utilisateur qui interroge le système de RÀPC.

La publication à ICCBR 2014 [Gaillard *et al.*, 2014b] et à la conférence européenne de gestion de connaissances (ECKM) en 2015 [Gaillard *et al.*, 2015b] ont présenté l'impact positif de la gestion de la fiabilité et de son utilisation pour filtrer des UC utilisées par le système de RÀPC et pour classer les réponses en fonction de la fiabilité des UC impliquées dans chacune des réponses retournées.

Gestion de l'aspect graduel des connaissances

Un avantage d'acquérir des connaissances grâce à une communauté en ligne est la possibilité de collecter différents avis à propos d'une même UC. La prise en compte des différents avis à propos d'une UC permet de représenter l'aspect graduel des représentations mentales. Les différents avis sont collectés sur les UC des connaissances du domaine organisées sous forme d'une ontologie, et plus particulièrement sur la relation de subsomption entre les classes de l'ontologie. Dans ce contexte, les sous-classes $\mathtt{B}_i$ d'une même classe `A` peuvent être distinguées au regard de `A` grâce à leur typicalité par rapport à `A`.

Dans un premier temps, notre approche propose d'ordonner les sous-classes $\mathtt{B}_i$ d'une même classe `A` grâce à la notion de typicalité. Cette représentation nous permet de structurer plus finement les sous-classes $\mathtt{B}_i$ de `A` en ajoutant des classes intermédiaires qui permettent de diviser les sous-classes $\mathtt{B}_i$ en vue de leur typicalité par rapport à `A`. Ce raffinement de l'ontologie permet d'améliorer la remémoration. La publication à ICCBR en 2015 [Gaillard *et al.*, 2015a] présente cette approche et évalue son impact positif sur les réponses retournées par le système de RÀPC.

Dans un deuxième temps, la représentation de la typicalité des sous-classes $\mathtt{B}_i$ de `A` nous permet, lors de la phase de remémoration, de sélectionner le(s) cas remémoré(s) le(s) plus typique(s) au regard du problème cible. La typicalité d'un cas remémoré au regard du problème cible dépend de la typicalité des sous-classes du problème cible et des sous-classes du cas remémoré avec les classes du problème cible où certaines contraintes ont été relaxées.

Plan du mémoire

Ce document est organisé de la façon suivante.
Le premier chapitre présente un état de l'art sur les notions utilisées dans la thèse. La section 1.1 présente les notions de communauté en ligne et de métaconnaissances au centre de cette thèse. La section 1.2 étudie les métaconnaissances utilisées dans le chapitre 3 de cette thèse, à savoir : la croyance, la qualité, la provenance, la confiance, la réputation et la fiabilité. La section 1.3 se

focalise sur la notion de confiance dans les systèmes informatiques. La section 1.4 présente les systèmes fondés sur la confiance qui utilisent les métaconnaissances présentées dans la section 1.2. En section 1.5, la catégorisation des connaissances dans une ontologie est ensuite étudiée en comparant différentes visions et en approfondissant la notion de typicalité utilisée dans le chapitre 5 de cette thèse. La section 1.6 présente un état de l'art sur le RàPC, au cœur de cette thèse, et les notions de métaconnaissances et de typicalité dans les systèmes de RàPC.

Le deuxième chapitre présente le système culinaire de RàPC eTaaable qui a été construit dans le cadre de cette thèse. eTaaable utilise le même moteur de RàPC que le système de RàPC Taaable présenté en section 2.1 . Les conteneurs de connaissances utilisés par eTaaable sont présentés en section 2.2. Dans eTaaable les conteneurs de connaissances sont édités de façon collaborative par une communauté en ligne sur le site Web aTaaable présenté dans la section 2.3. Le moteur commun de eTaaable et Taaable est un moteur générique de RàPC nommé Tuuurbine qui est présenté en section 2.4. Enfin, la section 2.5 présente la construction d'une base de tests réalisée pour évaluer les apports de cette thèse et pour être distribuée afin d'évaluer des travaux futurs.

Le troisième chapitre présente le premier apport principal de cette thèse. Il présente MKM, le modèle permettant de gérer la fiabilité des connaissances provenant d'une communauté en ligne et qui peut étendre un système de RàPC. Après avoir introduit les motivations, les principes de MKM et de l'extension d'un système de RàPC utilisant des connaissances provenant d'une communauté en ligne sont présentés en section 3.1. La section 3.2 introduit l'architecture générale de MKM et les différentes métaconnaissances permettant de calculer la fiabilité d'une UC, à savoir : la confiance *a priori* d'un utilisateur envers un autre utilisateur du système, la croyance d'un utilisateur envers une UC, la qualité d'une UC, la confiance calculée d'un utilisateur envers un autre utilisateur et la réputation d'un utilisateur dans la communauté en ligne. La section 3.2 est fondée sur l'étude bibliographique du chapitre 1. La section 3.3 présente l'utilisation de MKM pour aTaaable. La section 3.4 présente ensuite comment MKM est utilisé pour étendre un système de RàPC en ajoutant une fonction de filtre en amont du moteur du système permettant de ne prendre en compte que les UC les plus fiables dans le moteur et en ajoutant une fonction de classement en aval du système qui ordonne les réponses du système par fiabilité.

Le quatrième chapitre correspond à l'illustration des apports introduits en chapitre 3 dans le système eTaaable. La section 4.1 présente un exemple dans le système eTaaable étendu par MKM. L'évaluation de l'extension du système eTaaable avec MKM est présentée en section 4.2.

Le cinquième chapitre présente le deuxième apport principal de cette thèse. Ce chapitre se concentre sur la gestion de l'aspect graduel des connaissances pour d'une part raffiner l'ontologie du domaine grâce à la typicalité et d'autre part prendre en compte la typicalité dans le moteur du système de RàPC. Les motivations et principes de l'acquisition et de la représentation de la typicalité sont d'abord présentés. La section 5.1 décrit la méthode d'acquisition de la typicalité des sous-classes d'une superclasse. La section 5.2 présente plusieurs méthodes de répartition des sous-classes d'une superclasse de l'ontologie du domaine en plusieurs sous-ensembles grâce à la typicalité afin de raffiner l'ontologie et par conséquent affiner la remémoration. Enfin, la section 5.3 présente la sélection du ou des cas le(s) plus typique(s) durant la phase de remémoration du RàPC.

Le sixième chapitre correspond à l'illustration et à l'évaluation des apports introduits en chapitre 5 dans le système eTaaable. La section 6.1 présente un exemple des impacts du raffinement de l'ontologie du système eTaaable grâce à la typicalité. L'évaluation du raffinement de l'ontologie du système eTaaable grâce à la typicalité est présentée en section 6.2. La section 6.3 présente la sélection des cas les plus typiques durant la phase de remémoration du système eTaaable

avec un exemple. L'évaluation de l'intégration de la sélection des cas les plus typiques durant la phase de remémoration du système eTaaable est présentée en section 6.4.

Enfin, nous concluons le document en mettant en évidence les contributions et les perspectives de recherches.

Chapitre 1

Contexte et état de l'art

Sommaire

1.1 Définitions

1.1.1 Communauté en ligne

Une communauté en ligne, aussi appelée communauté virtuelle, sur l'Internet, est un groupe de personnes avec des idées, des buts ou des intérêts similaires qui interagissent en communiquant grâce à des outils Internet et qui dépassent les standards d'une communauté en faisant disparaître les limites de temps et d'espace [Lawrence, 1995,Rothaermel et Sugiyama, 2001]. [Rothaermel et Sugiyama, 2001] soulignent que les communautés en ligne permettent de satisfaire deux différents buts. Une communauté en ligne permet d'une part d'échanger des informations en combinant du contenu, par exemple une recette ou la description d'une maladie, et de la communication, par exemple un commentaire ou une publication dans un forum. Une communauté en ligne permet d'une autre part de satisfaire au moins un des besoins parmi les quatre suivants : échanger des informations autour d'un domaine d'intérêt, créer des relations virtuelles, faire des transactions ou se divertir. Cette thèse se focalise sur les communautés en ligne qui échangent des informations autour d'un domaine.

[Dreyer *et al.*, 2010] utilisent le terme de communauté en ligne pour définir « un groupe divers de personnes, délimité par un intérêt commun dans une industrie ou une organisation, qui se soucient de contribuer et de coopérer en ligne pour le bien être du groupe ». De plus, [Hagel et Armstrong, 1997] soulignent que les communautés en ligne émergent spontanément, bénéficiant de l'enthousiasme de ses membres. À partir de ces définitions, dans le cadre de cette thèse, nous définissons le terme communauté en ligne comme une agrégation de personnes qui ont un intérêt commun et veulent échanger des informations et des connaissances à propos de cet intérêt par le moyen d'une communication Internet.

Dans une communauté en ligne, l'échange des informations peut être régulé par une inscription obligatoire, cependant, ce n'est pas suffisant pour contrôler les caractéristiques des personnes de la communauté en ligne. Les caractéristiques des personnes de la communauté en ligne peuvent être définies selon plusieurs critères comme :

- le niveau d'expertise ;
- le niveau d'implication ;
- le niveau d'honnêteté.

La diversité des profils des utilisateurs dans une communauté en ligne impacte directement les informations provenant de cette communauté. Les informations apportées par des communautés en ligne sont hétérogènes, et peuvent être incertaines, incomplètes et/ou non fiables.

Dans la littérature, les caractéristiques des utilisateurs et les descriptions des informations peuvent être représentées par des métadonnées.

1.1.2 Donnée, information et connaissance

La distinction entre la notion de donnée, d'information et de connaissance n'est pas claire dans la littérature. Cette distinction doit cependant être faite afin de pouvoir les manipuler.

Dans le domaine de la gestion de données et de la gestion de connaissances plusieurs auteurs ont définis les trois termes afin de les distinguer. Si la définition de donnée est clairement définie pour les différents auteurs, la frontière entre information et connaissance est souvent floue. Nous allons cependant faire ressortir les distinctions qui sont faites par ces auteurs.

Pour [Davenport et Prusak, 1998, Bubenko et Orci, 1989, Wiederhold, 1984] les données sont des faits observables qui peuvent être volumineux, qui sont objectivement vérifiables et peuvent changer rapidement dans le temps.

Pour [Bubenko et Orci, 1989], une information est une donnée ou une collection de données interprétées tandis qu'une connaissance est plus complexe, impliquant un niveau d'abstraction plus élevé et une certaine généralisation [Ramamoorthy et Wah, 1989, Wiederhold, 1984]. [Wiederhold, 1984] ajoute qu'une connaissance est imprécise ou incertaine car elle est subjective et ne peut donc pas être objectivement vérifiable. De plus, contrairement aux données, pour [Wiederhold, 1984] les connaissances ne changent pas rapidement. [Bubenko et Orci, 1989, Newell, 1982] affirment que les connaissances sont liées à un processus d'interprétation ; elles sont liées à un but.

Dans le domaine de l'intelligence artificielle, [Aamodt et Nygård, 1995] ont étudié les trois termes afin de les définir.

- Les données sont définies comme les entrées syntaxiques d'un processus d'interprétation qui est la première étape d'un processus de décision.
- Les informations sont des données à qui ont donne du sens ; les informations sont les entrées et les sorties dans le processus de décision d'un système à base de connaissances.
- Les connaissances sont des informations apprises et utilisées comme des ressources dans un processus de décision ; les connaissances sont la sortie d'une processus d'apprentissage.

[Aamodt et Nygård, 1995] distinguent trois rôles différents des connaissances dans un système informatique.

- Interprétation de données : les connaissances permettent de créer des informations à partir des données.
- Élaboration d'informations : les connaissances permettent de créer de nouvelles informations à partir d'une collection d'informations.
- Apprentissage de connaissances : en créant de nouvelles informations, de nouvelles connaissances peuvent être apprises et utilisées dans de futurs processus de décision.

Nous utilisons les définitions de [Aamodt et Nygård, 1995] dans ce travail.

1.1.3 La notion de métaconnaissance

Une métadonnée est une donnée utilisée pour définir ou décrire une autre donnée quel que soit son support. Selon [Berners-Lee, 1997], une métadonnée décrivant une ressource électronique peut être interprétée par une machine à propos d'une ressource électronique. En 1995, un modèle pour décrire les métadonnées des ressources électroniques, le *Dublin Core Metadata*[2], a été créé. Il consiste en 15 propriétés comme le titre, le créateur ou le sujet d'un document. En 1999, une recommandation du *Word Wide Web Consortium*[3] (W3C) définit comment créer un réseau de métadonnées utilisant le langage RDF (*Resource Description Framework*) [McBride, 2004].

De la même façon que les métadonnées sont des données sur les données, les métaconnaissances sont des connaissances pour définir ou décrire d'autres connaissances. [Pitrat, 1990] qui a réalisé une étude détaillée sur le terme de métaconnaissance, a défini différents types de métaconnaissances qui peuvent être (1) des propriétés sur les connaissances, par exemple l'utilité ou la veracité des connaissances, (2) des connaissances sur les connaissances d'un sujet, par exemple quelles connaissances à un individu, et (3) des connaissances sur l'utilisation des connaissances, par exemple comment stocker ou créer des nouvelles connaissances. Un modèle de métaconnaissances concernant les propriétés des connaissances a été proposé par [Naudet *et al.*, 2010] pour

2. `http://dublincore.org/`
3. `https://www.w3.org/`

décrire des connaissances incomplètes et non fiables provenant d'une communauté en ligne. Dans ce modèle, chaque unité de connaissances est décrite par des métaconnaissances qui intègrent un ensemble d'aspects qui sont la confiance, la croyance, la provenance, le contexte et la qualité.

Le modèle de métaconnaissances présenté dans la thèse est fondé sur le modèle de [Naudet *et al.*, 2010] pour représenter la fiabilité d'une connaissance pour un utilisateur à partir de métaconnaissances telles que la croyance, la qualité, la confiance et la réputation.

1.2 Présentation des différentes métaconnaissances utilisées dans notre modèle de métaconnaissances

1.2.1 La croyance

Le dictionnaire d'Oxford définit la croyance comme « l'acceptation qu'une affirmation est vraie ou que quelque chose existe » [4].

Plusieurs auteurs ont étudié la notion de croyance, notamment [Kant, 1869] pour qui la croyance est un fait de notre esprit reposant sur des principes objectifs doués de raison et d'autres plus subjectifs propres à l'esprit de l'individu qui possède cette croyance. Selon [Hume, 2012], une croyance possède deux caractéristiques : c'est une idée forte liée à une impression sur le moment et cette idée forte engage l'individu qui la possède dans une action. Beaucoup de travaux ont été réalisés en mettant en parallèle la connaissance et la croyance, selon [Nietzsche, 1878] une croyance même forte n'est pas une connaissance tandis que pour [Kant, 1869], seules les croyances communicables et valables pour chacun nous font accéder à la connaissance.

[Rao *et al.*, 1995] ont introduit l'architecture BDI (*Belief, Desires, intentions*) qui est un cadre pour une logique où un agent a des croyances, des désirs et des intentions. L'architecture BDI repose sur la définition de [Hume, 2012], puisque les croyances d'un agent va l'engager dans l'action.

Dans la littérature, la croyance est représentée grâce à différents modèles qui utilisent la logique modale [Marek et Truszczyński, 1991, Halpern et Moses, 1985], la logique défaisable de [Nute, 2001], l'approche probabiliste bayésienne [Smets et Kennes, 1994] ou la logique subjective [Jøsang, 2001]. [Jøsang, 2001] utilise la logique subjective pour représenter une croyance incertaine ou incomplète. Une croyance est représentée par l'opinion d'un utilisateur u à propos d'une proposition x représentée par un quadruplet $\mathcal{W}_x^A = (b,d,u,a)$ où $b+d+u=1$ avec $b,d,u \in [0,1]$ qui représentent respectivement la croyance (*belief*), l'incrédulité (*disbelief*), l'incertitude (*uncertainty*) et avec $a \in [0,1]$ le taux de base représentant l'opinion *a priori* de x. Le taux de base a est utilisé lorsque u ne possède aucune évidence pour représenter b, d et u. Comme le souligne [Kant, 1869], les paramètres du quadruplet définissant l'opinion de u sur x sont calculés en fonction d'évidences, c'est-à-dire, des faits objectifs, mais aussi en fonction d'impressions subjectives de u liées à x. Dans le modèle de [Richardson *et al.*, 2003], la croyance personnelle d'un agent en une affirmation représente l'exactitude, la crédibilité et/ou la pertinence de l'affirmation.

Il est pertinent dans cette thèse de représenter la croyance personnelle d'un utilisateur à propos d'une connaissance. En effet, les croyances de l'utilisateur à propos des UC impliquées dans un processus de raisonnement influencent la satisfaction de l'utilisateur sur les réponses retournées par le processus. Notre définition de la croyance d'une UC repose sur celle de [Richardson *et al.*, 2003] où un utilisateur va croire en une UC si il pense qu'elle est exacte, c'est-à-dire, qu'elle

4. *"An acceptance that something exists or is true, especially one without proof"*, `http://www.oxforddictionaries.com/`

est conforme à la vérité, ou crédible par rapport à ses connaissances, c'est-à-dire, qu'elle peut être crue selon ses propres connaissances.

Dans notre travail, nous définissons la croyance comme suit :

Définition 1. *La croyance d'un utilisateur envers une unité de connaissances est le degré d'acceptation subjective que l'unité de connaissances soit exacte ou crédible.*

Un utilisateur possède aussi une croyance envers les UC qu'il a lui-même produites et cette croyance peut être partielle. En effet, un utilisateur peut produire une UC envers quoi il ne croit pas totalement, ou envers quoi sa croyance pourra évoluer avec l'apparition de nouveaux faits, comme la contradiction de cette UC par un expert.

En se fondant sur [Kant, 1869], les croyances des utilisateurs à propos d'une UC est un indicateur de la qualité de cette UC.

1.2.2 La qualité

La qualité d'une donnée a été définie par [Juran, 1988] comme l'aptitude de la donnée à être utilisée dans les tâches prévues (de prise de décision par exemple) et par [Crosby, 1980] comme étant conforme aux conditions requises. Dans de nombreux travaux, la qualité d'une donnée est définie comme un concept multidimensionnel où chaque dimension influence le niveau de qualité de la donnée. Les dimensions qui reviennent le plus souvent sont l'exactitude, la crédibilité, la rapidité de mise à jour, la cohérence, la fiabilité, l'utilité, la provenance et l'accessibilité [Wang et Strong, 1996, Wang, 1998, Shankar et Watts, 2003, Redman, 1997]. [Wang et Strong, 1996] listent 15 dimensions organisées en 4 catégories :

- la qualité intrinsèque : l'exactitude, la crédibilité, l'objectivité et la réputation ;
- la qualité contextuelle : la pertinence, la rapidité de mise à jour, la couverture et la quantité de données pertinentes ;
- la qualité de représentation : la facilité d'interprétation, la facilité de compréhension et la consistance ;
- la qualité d'accessibilité : l'accessibilité et la sécurité.

[Zaveri *et al.*, 2015] ajoutent une catégorie additionnelle qui correspond au concept de confiance et ajoutent de nouvelles dimensions : la croyance, la provenance et la réputation. La provenance, la confiance et la réputation sont étudiées dans les sections suivantes. La qualité des connaissances, caractérisée par les dimensions précédemment énoncées, ont un impact important sur le succès des prises de décisions de systèmes à base de connaissances [Shankaranarayan *et al.*, 2003, DeLone et McLean, 1992].

Il est donc important, notamment dans cette thèse, d'évaluer la qualité d'une UC dans une base de connaissances alimentée par des utilisateurs d'une communauté en ligne, dont l'expertise peut être variable. Dans notre travail, nous considérons uniquement la qualité intrinsèque d'une UC, et nous définissons la qualité comme suit :

Définition 2. *La qualité d'une unité de connaissances est la perception de la communauté sur l'exactitude et la crédibilité de cette unité de connaissances.*

1.2.3 La provenance

Dans la littérature, la provenance d'une information est considérée selon deux aspects différents. Dans le premier aspect, la provenance d'une information correspond au processus qui a créé ou transformé cette information, tandis que le second aspect correspond à l'origine de

l'information avant sa création dans le système [Buneman *et al.*, 2001, Pinheiro da Silva *et al.*, 2003, Ding *et al.*, 2004, Tan, 2004, Groth *et al.*, 2006]. Sur le Web, la provenance d'une connaissance est un indicateur de la qualité ou de la fiabilité de cette connaissance [Golbeck et Mannes, 2006, Hartig et Zhao, 2009].

Dans cette thèse, nous considérons seulement la première vision de la provenance d'une UC en se focalisant sur l'auteur de l'UC. Ainsi, dans le contexte d'une communauté en ligne, nous définissons la provenance comme suit :

Définition 3. *La provenance d'une unité de connaissances correspond à l'utilisateur qui a produit l'unité de connaissances.*

1.2.4 La confiance

Bien que [Luhmann, 1979] ait présenté la confiance comme un fait basique de la vie humaine, la confiance est un terme complexe à définir. De nombreux auteurs en sciences humaines ont tenté de définir la confiance, mais aucun d'entre eux n'est parvenu à une définition consensuelle.

Pour [Deutsch, 1962] ainsi que pour [Boon et Holmes, 1991], la confiance est une dimension individuelle où la confiance d'un individu envers une tierce personne est la conscience d'être en présence d'une situation où l'issue négative peut être beaucoup plus importante que l'issue positive, mais que malgré tout, l'individu pense que les capacités et les intentions de la tierce personne dont dépend l'issue positive sont suffisantes pour prendre ce risque.

Pour certains auteurs, la confiance s'inscrit dans une réalité sociale. Selon [Luhmann, 1979], la confiance permet à un individu de diminuer la complexité environnementale qui l'entoure ; faire confiance permet de s'adapter à son environnement en détournant certaines possibilités pouvant découler d'une situation. Pour [Barber, 1983], l'acte de faire confiance à un individu est une espérance sur les compétences techniques de l'individu mais aussi sur sa responsabilité et ses obligations morales. Quant à [Putnam *et al.*, 1994], il affirme que la confiance est un outil de la vie sociale qui permet de coordonner les actions.

Selon [Gambetta, 2000], la confiance est une espérance sur les actions de la personne à qui on accorde sa confiance qui a des conséquences sur nos propres actions. De plus, tout comme [Marsh, 1994], Gambetta définit la confiance comme dépendant du contexte.

Dans le domaine informatique, lorsque les utilisateurs d'un système fournissent des services ou des connaissances, la notion de confiance est centrale dans les processus de prises de décision. La confiance doit alors être représentée et mesurée. [Golbeck, 2005] se rapproche de la définition de Gambetta, pour définir la confiance comme un engagement à croire au bon déroulement des futures actions d'un utilisateur, par exemple, en *e-commerce*, croire qu'un utilisateur fournira un produit de qualité à un moindre prix dans un délai réduit. Pour [Gambetta, 2000], « La confiance est une probabilité subjective en quoi un utilisateur u espère qu'un autre utilisateur v effectue une action donnée de laquelle son bien-être dépend. » Selon [Grandison et Sloman, 2000], la confiance est dépendante d'un contexte spécifique et se définit par une croyance quantifiée quant aux capacités d'un utilisateur. [Jøsang *et al.*, 2007] ainsi que [McKnight et Chervany, 1996] mettent en relation les notions de volonté, de dépendance, de contexte et de risque puisqu'ils définissent la confiance comme la volonté de dépendre de quelqu'un ou de quelque chose dans une situation rassurante malgré la possibilité de conséquences négatives.

Certains auteurs comme [Sabater et Sierra, 2002, Artz et Gil, 2007, Jøsang *et al.*, 2007, Gaillard, 2011, Alchiekh Haydar, 2014] ont dégagé certaines généralités sur la confiance présentées dans la section suivante. Les définitions sur la confiance diffèrent, cependant trois éléments sont toujours présents dans les définitions :

- une personne faisant confiance (*trustor*) ;
- une cible ou une personne envers qui la confiance est attribuée (*trustee*) ;
- une situation.

De plus, certaines notions et propriétés ressortent lors de la définition de la confiance. Les notions et propriétés de la confiance sont présentées dans la section 1.3 tandis que la section 1.4 présente les systèmes informatiques fondés sur la confiance.

Dans cette thèse, nous nous intéressons aux connaissances produites par les utilisateurs. Ainsi, la définition de la confiance est directement en rapport avec les connaissances produites. Dans notre travail, nous définissons la confiance comme suit :

Définition 4. *La confiance d'un utilisateur u envers un autre utilisateur v dans une communauté en ligne portant sur un domaine spécifique correspond à la perception subjective de u et/ou des connaissances apportées par v.*

Nous n'intégrons pas la notion de contexte dans cette définition car dans notre travail, le contexte de la confiance correspond au domaine spécifique de notre application, c'est-à-dire la cuisine. Cependant, nous pourrons intégrer la prise en compte de différents contextes plus spécialisés dans de prochains travaux. Par exemple, dans le domaine de la cuisine, Marie peut avoir confiance envers Paul dans le contexte de la cuisine méditerranéenne (par exemple, pour la recette de la bouillabaisse) mais ne pas avoir confiance envers Paul dans le contexte de la cuisine traditionnelle française (par exemple, pour la recette du bœuf bourguignon).

1.2.5 La réputation

Pour [Jøsang *et al.*, 2007], dans les systèmes informatiques, la réputation est ce qui est dit ou cru à propos d'un utilisateur. La réputation est considérée comme une mesure globale ou collective de la fiabilité d'un utilisateur fondée sur les recommandations ou les évaluations des autres utilisateurs de la même communauté. Selon [Sztompka, 1999], la réputation d'un utilisateur u représente comment un autre utilisateur doit croire le contenu produit par l'utilisateur u. Pour [Zacharia et Maes, 2000], la réputation est multimodale dans le sens où un utilisateur peut avoir différentes réputations en fonction du contexte et la réputation d'un utilisateur est fondée sur des processus sociaux qui dépendent des interactions passées entre l'utilisateur jugé et les utilisateurs du système.

Dans les réseaux sociaux comme Facebook [5] ou Twitter [6], la réputation d'un utilisateur peut dépendre du nombre d'amis ou d'abonnés, du nombre de comptes que l'utilisateur a sur les différents réseaux sociaux, du suivi et des évaluations des contenus publiés, etc. [Agichtein *et al.*, 2008, Romero *et al.*, 2011, Bakshy *et al.*, 2011]. Pour [Gisselbrecht *et al.*, 2015], la « qualité » d'un utilisateur dans un réseau social, qui peut être vue comme la réputation de l'utilisateur, est relative à ses actions sur le réseau social. La qualité de l'utilisateur peut être caractérisée par le contenu des messages publiés par l'utilisateur, la popularité de l'utilisateur et/ou la popularité de l'utilisateur sur une thématique donnée.

Selon [Resnick et Zeckhauser, 2002], un système de confiance à base de réputation doit avoir *trois* propriétés :

- Les utilisateurs sont considérés sur une longue période : à chaque interaction il y a l'attente de futures interactions.
- Les évaluations à propos des interactions des utilisateurs dans le système doivent être collectées et distribuées.

5. `htttp://facebook.com/`
6. `htttp://twitter.com/`

– Les évaluations passées doivent guider les décisions sur les interactions courantes.

De nombreux systèmes modélisent la réputation des utilisateurs comme les systèmes pair à pair (P2P) [Kamvar *et al.*, 2003], les systèmes bayésiens [Jøsang et Ismail, 2002], les systèmes utilisant des communautés en ligne [Levien, 2009] ou des sites de commerce en ligne [Sabater et Sierra, 2002]. Les détails et principes du calcul de la réputation dans ces différents systèmes sont décrits en section 1.3.

Dans notre travail, nous définissons la réputation comme suit :

Définition 5. *La réputation d'un utilisateur est une mesure de la perception collective de la communauté envers cet utilisateur et des connaissances qu'il a créées.*

Nous verrons dans la section 1.3 que dans certains travaux, la réputation d'un utilisateur v est une mesure de la perception d'un seul utilisateur u envers v. Cependant, dans notre travail, la réputation de v est la mesure de la perception de toute la communauté à propos de v alors que la mesure de la perception de u à propos de v correspond à la confiance de u envers v.

1.2.6 Fiabilité

Cette section fait référence à la fiabilité qui est la traduction en anglais de *trustworthiness* et non de *reliability*[7]. Dans la littérature, la fiabilité peut être (1) entre un utilisateur et un autre utilisateur ou (2) entre un utilisateur et une information ou un objet. Nous nous concentrons ici sur le deuxième cas. La fiabilité entre un utilisateur et une information ou un objet correspond alors à la confiance d'un utilisateur envers cette information ou cet objet.

Plusieurs auteurs ont proposé des modèles pour construire ou prédire la fiabilité d'une information ou d'un objet pour un utilisateur. Le modèle construit par [Golbeck, 2005] prédit la fiabilité d'un film pour un utilisateur dans un système de recommandation. Dans [Gil et Ratnakar, 2002], une façon de représenter la fiabilité d'une affirmation extrait d'un document est proposée. Le détail de ces deux modèles est présenté dans la section 1.4.

Dans la littérature, la fiabilité est fortement liée à d'autres concepts. [Wang et Strong, 1996] comparent la qualité et la fiabilité d'une information. Pour ces auteurs, la qualité représente l'aptitude de l'information à être utilisée tandis que la fiabilité définit la probabilité perçue qu'une information préserve la confiance entre l'utilisateur qui juge l'information et l'utilisateur qui est à la l'origine de l'information. [Bertino et Lim, 2010, Dai *et al.*, 2008] montrent que la fiabilité d'une information dépend de la provenance de cette information. Pour [Moturu et Liu, 2011], la fiabilité d'un contenu partagé sur les réseaux sociaux peut être représentée par la réputation de l'auteur de ce contenu, les performances de ce contenu, si le contenu a été partagé ou évalué, et les qualités perçues de ce contenu, telles que sa taille, son style ou sa structure. Selon [Cobden, 2014], la fiabilité d'une information dépend de la qualité de l'information, de la provenance de l'information et des acteurs ayant participé à la création de l'information en prenant en compte des signaux observables, tels que leur apparence, leur comportement et leur réputation.

Dans la littérature, la fiabilité d'une information ou d'un objet pour un utilisateur peut être représentée pour différents types d'informations ou d'objets :

– une affirmation provenant d'un document ou d'un utilisateur. Par exemple, [Gil et Ratnakar, 2002] représentent la fiabilité de l'affirmation provenant d'un document, et [Richardson *et al.*, 2003] proposent de prédire la fiabilité d'une affirmation d'un utilisateur u par rapport aux utilisateurs envers qui il a confiance et qui ont évalué l'affirmation (cf. section 1.4 pour les détails de ces modèles).

7. En anglais, *reliability* fait référence à la fiabilité d'un système, d'un logiciel ou d'une machine en termes de proportion de pannes attendues.

- un article dont l'auteur n'est pas toujours connu et dont le contenu est souvent long (plus d'une phrase) et peut être modifié par l'auteur ou une autre personne (par exemple dans Wikipédia[8]), commenté ou évalué (par exemple, [Adler *et al.*, 2008, Qin et Cunningham, 2012, Pal et Counts, 2011]).
- un item qui est un objet du domaine d'application pouvant être un livre, un film, etc. Par exemple, le modèle construit par [Golbeck, 2005] prédit la fiabilité d'un film pour un utilisateur dans un système de recommandation et [Abdul-Rahman et Hailes, 2000a] proposent de représenter la fiabilité d'un item en fonction des recommandations d'utilisateurs dans le système.
- du contenu provenant des systèmes sociaux comme Facebook ou Twitter. Les particularités de ces contenus est qu'ils n'ont de sens qu'avec le lien de leur auteur, que le contenu peut être de mauvaise qualité et que le contenu n'a d'existence que dans un laps de temps réduit. Par exemple, [Castillo *et al.*, 2011] proposent un modèle pour représenter la crédibilité (similaire à la fiabilité) d'un contenu publié sur Twitter (un *tweet*).

L'étude de la littérature montre alors que la fiabilité d'une information ou d'un objet x pour un utilisateur u peut être liée à la provenance de x, à la qualité de x, à la confiance de u envers l'utilisateur v qui a fait une recommandation à propos de x ou qui a produit x, et/ou à la réputation de v. Dans le cas où v a produit x, la prise en compte de la confiance de u envers v ou de la réputation de v dans la représentation de la fiabilité de x pour u permet dans un même temps de prendre en compte la provenance de x, d'après la définition 3.

À partir de ces constatations et du contexte de la thèse, nous définissons la fiabilité comme suit :

Définition 6. *La fiabilité d'une unité de connaissances pour un utilisateur u est la représentation de l'utilité et de l'exactitude de cette unité de connaissances pour u fondées sur la qualité de l'unité de connaissances, la confiance de u envers l'utilisateur v qui a créé l'unité de connaissances et la réputation de v.*

1.3 La confiance dans les systèmes informatiques

1.3.1 Paramètres et propriétés de la confiance

La confiance a certaines propriétés qui sont consensuelles pour les auteurs que nous avons étudiés :

- La confiance n'est pas symétrique : « Jean a confiance en Pierre » n'entraîne pas que « Pierre a confiance en Jean ».
- La confiance n'est pas distributive : « Jean a confiance en (Pierre et Alice) », signifiant que Jean a confiance au duo, Pierre et Alice, n'entraîne pas que « Jean a confiance en Pierre seul » et « Jean a confiance en Alice seule ». Par exemple, Jean peut avoir confiance en Pierre et Alice pour la préparation du repas de noël car Pierre est un bon cuisinier pour les plats de viandes et Alice est une bonne pâtissière. En revanche, pour la préparation du repas de noël, Jean n'a pas confiance en Pierre seul car il ne sait pas faire les desserts et Jean n'a pas confiance en Alice seule car elle ne sait pas préparer les plats de viandes.

8. `htttp://wikipedia.org/`

- La confiance est contextuelle : « Jean a confiance en Pierre dans le domaine de la mécanique » n'entraîne pas que « Jean a confiance en Pierre dans le domaine culinaire ».
- La confiance n'est pas transitive : si « Jean a confiance en Pierre » et « Pierre a confiance en Alice » alors « Jean a confiance en Alice » n'est pas forcément vraie. Une discussion sur la transitivité est détaillée en section 1.3.3.

D'autres notions connexes à la confiance reviennent chez de nombreux auteurs :

- le risque : plus le fait d'accorder sa confiance est risquée, plus la confiance sera difficile à accorder, mais plus la confiance sera renforcée si elle est accordée [Luhmann, 1979]. [Marsh, 1994] définit le risque par un rapport entre les coûts et les bénéfices de la situation de confiance. [Boyle et Bonacich, 1970] ont étudié la notion de risque influençant la confiance en théorie des jeux dans le dilemme du prisonnier.
- la réciprocité : certains auteurs considèrent que si un utilisateur u s'est montré fiable pour un autre utilisateur v, alors u peut penser que v aura un comportement similaire. Ainsi, la confiance de v en u est augmentée [Boyd et Richerson, 2009, Marsh, 1994]. Cependant, un effet de réciprocité ne veut pas dire que la confiance est symétrique. Par exemple, si Pierre s'est montré fiable envers Jean, Pierre peut penser que Jean sera quelqu'un de fiable. Ainsi la confiance de Pierre envers Jean sera augmentée, mais ne sera pas forcément du même niveau que la confiance de Jean envers Pierre.
- la mémoire : la confiance peut être représentée en utilisant toutes les expériences passées [Resnick et Zeckhauser, 2002], en utilisant seulement les expériences les plus récentes [Jøsang et Ismail, 2002, Das et Islam, 2012] ou en favorisant les expériences les plus récentes grâce à un facteur d'oubli [Marsh, 1994].
- la prise de décision : la confiance d'un utilisateur envers un autre utilisateur est à la source de la prise de décision de coopérations, de compétitions ou de transactions, comme pour le système REGRET [Sabater et Sierra, 2002]. Marsh [Marsh, 1994] a établi un seuil de confiance pour établir une coopération entre un utilisateur u et un autre utilisateur v, car selon [Williams, 1988], une coopération réussie entre u et v, n'est possible que lorsque u a confiance en v.

1.3.2 Confiance globale vs. confiance locale

La confiance gérée par les systèmes informatiques peut être une confiance locale ou une confiance globale. Dans les deux cas, la confiance peut être gérée soit de façon centralisée, soit localement par chaque utilisateur ; la gestion de la confiance est alors distribuée. Lorsque la confiance est gérée de manière distribuée, le système doit intégrer un moyen de communication pour propager la confiance entre utilisateurs.

Confiance globale. La confiance globale est considérée comme étant équivalente à la réputation. La confiance globale exprime l'opinion de tous les utilisateurs du système envers un utilisateur u.

Si la confiance globale est gérée de manière centralisée, à chaque fois qu'un utilisateur u interagit avec un utilisateur v et que u a évalué le résultat de cette interaction, grâce à un score par exemple, cette évaluation est envoyée et stockée dans une unité centrale afin de mettre à jour l'ancien score de réputation de v dans le système. Ainsi, à tout moment un utilisateur peut interroger l'unité centrale afin de connaître le score de réputation d'un utilisateur en particulier.

Si la confiance globale est gérée de manière distribuée, la réputation d'un utilisateur v est obtenue en demandent à chaque utilisateur du système ses évaluations d'interaction avec v.

Confiance locale. La confiance locale représente la fiabilité d'un utilisateur v selon l'opinion d'un seul utilisateur u ou selon le réseau de confiance de u si u n'a pas d'opinion sur v. Le réseau de confiance de u correspond à l'ensemble des utilisateurs du système envers qui u a confiance.

Dans le cas où la confiance est gérée de manière centralisée, chaque évaluation du résultat d'une interaction de u avec v est stockée dans une unité centrale. À la réception d'une évaluation, l'unité centrale met à jour la confiance locale de u envers v. Dans le cas où la confiance est gérée de manière distribuée, l'évaluation du résultat de l'interaction de u avec v est stockée par u, ce qui permet de mettre à jour la confiance locale de u envers v. Si u désire interagir avec v et qu'il n'a jamais interagi avec v, alors la confiance locale doit être calculée. La confiance locale de u envers v doit alors être inférée par transitivité dans le réseau de confiance de u. La transitivité de la confiance est présentée dans la section 1.3.3.

[Haydar *et al.*, 2014] distinguent en plus la confiance collective qui représente la fiabilité d'un utilisateur v selon l'opinion d'un seul utilisateur u combinée à l'opinion des utilisateurs du réseau de confiance de u, même si u a une opinion sur v.

1.3.3 Transitivité de la confiance

x a confiance en y et y a confiance en z n'implique pas que x a confiance en z. En effet, Marsh [Marsh, 1994] et [Christianson et Harbison, 1996] prouvent que la confiance n'est pas transitive. Le premier point important soulevé par [Christianson et Harbison, 1996] est le problème du contexte. Par exemple, si Jean a confiance en Pierre pour réparer sa voiture et Pierre a confiance en Alice pour lui installer le système d'exploitation de son ordinateur, alors il n'est pas possible d'inférer que Jean a confiance (ou non) en Alice pour la réparation de sa voiture ou même pour l'installation du système d'exploitation de son ordinateur. De plus, pour [Christianson et Harbison, 1996], lorsque Jean fait confiance à Pierre sur ses compétences en tant que réparateur de voiture, cela ne veut pas dire qu'il lui fait confiance sur ses capacités de jugement des compétences d'une tierce personne quant à la réparation d'une voiture.

Cependant, certains auteurs utilisent la transitivité de la confiance pour propager la confiance locale dans leur système, par exemple [Kamvar *et al.*, 2003]. Dans les systèmes utilisant la transitivité de la confiance, lorsqu'un utilisateur u n'a jamais eu d'expérience directe avec un utilisateur cible v, u calcule sa confiance locale en v en demandant des recommandations aux utilisateurs de son réseau de confiance. Le réseau de confiance de u correspond au groupe d'utilisateurs envers lesquels il a confiance. Ce réseau peut être constitué des amis de u, c'est-à-dire, les utilisateurs avec qui u a déjà eu une interaction et dont le résultat l'a satisfait, des amis de ses amis, des amis des amis de ses amis, etc. L'amplitude du réseau de confiance dépend des systèmes.

Pour contourner le problème de la transitivité de la confiance d'un utilisateur u envers un utilisateur v, certains auteurs comme [Jøsang et Pope, 2005] introduisent la notion de contexte et deux types de confiance illustrés dans la figure 1.1 :

- la confiance de référence correspond à la confiance sur les capacités de jugement des compétences d'une tierce personne, c'est-à-dire, une confiance sur les recommandations. Pierre transmet sa confiance en Alice à Jean comme une recommandation. Dans la figure 1.1, Jean Possède une confiance de référence en Pierre, c'est-à-dire que Jean a confiance en les recommandations de Pierre dans un domaine spécifique.
- la confiance fonctionnelle qui correspond à la confiance sur les compétences d'un autre utilisateur dans un domaine spécifique. Dans la figure 1.1, la confiance fonctionnelle se trouve entre Pierre et Alice.

[Jøsang et Pope, 2005] précisent que la confiance de référence et la confiance fonctionnelle doivent être utilisées dans le même contexte, par exemple toutes deux dans le domaine de la mécanique.

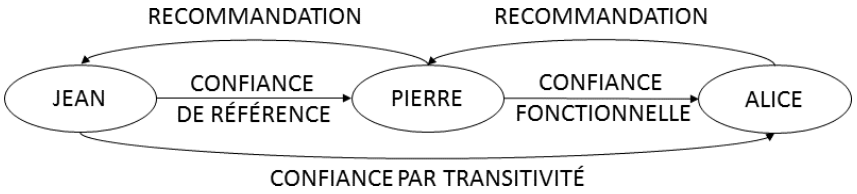


FIGURE 1.1 – Confiance par transitivité.

[Abdul-Rahman et Hailes, 1996] ajoutent que la confiance par transitivité de Jean envers Alice peut être caractérisée par un degré de confiance qui peut être différent de la confiance fonctionnelle de Pierre envers Alice, car elle est influencée par la confiance de référence de Jean envers Pierre.

Le calcul de la confiance par transitivité peut s'appuyer sur d'autres travaux en logique subjective [Jøsang *et al.*, 2006] étudiés dans la section 1.4 ou dans la théorie des sous-ensembles flous. Pour certains auteurs comme [Ramchurn *et al.*, 2004,Lesani et Bagheri, 2006], la confiance peut être vue comme une relation floue où un degré d'appartenance α est associé à un couple (u,v) signifiant que l'utilisateur u possède un degré de confiance α envers l'utilisateur v. Dans ce contexte, la confiance par transitivité peut être calculée à partir d'opérateurs introduits dans le cadre d'études sur la transitivité entre relations floues. Par exemple, [Sanchez, 1984] introduit l'opérateur max-min pour calculer la valeur de la relation floue R entre x et z à partir de la valeur de R entre x et y et de la valeur de R entre y et z : $R(x,z) \geqslant max(min(R(x,y),R(y,z)))$.

1.4 Systèmes fondés sur la confiance

Cette section présente des systèmes ou modèles reposant sur la confiance et qui peuvent manipuler les notions de croyance, de provenance, de réputation et/ou de fiabilité.

1.4.1 Représentation de la confiance dans les systèmes commerciaux

De nombreux services en ligne existent pour aider les utilisateurs à choisir de nouveaux produits. Nous retrouvons parmi les plus connus Ebay [9], Amazon [10] et Epinions [11].

Ebay. Ebay est un site de ventes aux enchères où des vendeurs particuliers vendent des articles divers à des acheteurs. Après chaque vente, l'acheteur et le vendeur se donnent mutuellement un score pour évaluer la qualité de l'échange. Ebay est un système de confiance centralisée fondé sur la réputation. Son fonctionnement est très simple puisque le score de réputation est calculé en soustrayant les scores négatifs aux scores positifs. Cependant, ce système introduit de nombreux biais :

– il y a un effet de réciprocité entre l'acheteur et le vendeur, car les notes octroyées sont visibles par les deux parties. En effet, de peur d'obtenir une note négative, très peu d'utilisateurs osent mettre de tels scores.

9. `http://ebay.com/`
10. `http://amazon.com/`
11. `http://epinions.com/`

- un utilisateur avec un score très mauvais peut se réinscrire sous un nouveau nom et ainsi effacer son passé non fiable.
- le nombre de votes n'est pas pris en compte. Un utilisateur qui a reçu 10 votes positifs et 1 vote négatif aura une moins bonne réputation qu'un utilisateur qui aura reçu 30 scores positifs et 20 scores négatifs.

Amazon. Amazon est un site de vente en ligne qui s'est d'abord spécialisé dans les livres pour s'élargir au matériel informatique, vêtements, accessoires de beauté, etc. Un membre peut noter un produit avec un système d'étoiles allant de 1 à 5. Les notes des produits sont utilisées pour faire des recommandations sur des produits similaires. Tous les utilisateurs, membres ou non, peuvent évaluer les commentaires sur les produits en exprimant si le commentaire a été utile ou non. Les commentaires sont alors classés par ordre décroissant d'utilité et pour chaque commentaire, il est indiqué le nombre d'utilisateurs qui ont évalué le commentaire et le nombre de fois qu'un utilisateur a trouvé le commentaire utile.

Des problèmes apparaissent du fait que des utilisateurs non-inscrits peuvent voter. En effet, une personne peut voter plusieurs fois pour un même commentaire et le faire monter dans le classement. Une solution trouvée par les créateurs du site a été de limiter un vote par adresse IP, pour un commentaire donné.

Epinions. Le site Epinions [12] est un site Web qui recueille des évaluations ou des commentaires sur tout type de produits allant de l'électroménager aux films. Un utilisateur inscrit peut noter un produit avec une échelle allant de 1 à 5 étoiles, où 1 étoile est la plus mauvaise note, et donner un avis plus détaillé à travers un commentaire. De plus, un utilisateur peut noter un commentaire laissé par un autre utilisateur grâce à une échelle de 4 modalités : pas utile, un peu utile, bien utile et très utile. Ainsi, les commentaires d'un produit sont classées par qualité, calculée à partir des évaluations reçues, lors de l'affichage d'un produit.

Un utilisateur u a aussi la possibilité d'ajouter un autre utilisateur dans son réseau de confiance constitué des utilisateurs qu'il considère les plus fiables. De plus, u peut ajouter des utilisateurs dans une liste noire. Pour un produit donné, les évaluations et commentaires des utilisateurs de la liste noire ne sont pas affichés, et les évaluations et commentaires des utilisateurs du réseau de confiance sont affichés en premier. Les données d'Epinions forment la base d'évaluation utilisée par le modèle MoleTrust [Massa et Avesani, 2004], présenté dans la section 1.4.3, qui représente la confiance dans un système de filtrage collaboratif.

1.4.2 Un modèle général de la confiance : le modèle de Marsh

Dans son modèle, [Marsh, 1994] se place dans le courant des systèmes multi-agents, mais ne traite de la confiance qu'entre deux agents. Dans son modèle, la confiance est un score compris dans l'intervalle $[-1,1[$. Marsh accepte la méfiance complète correspondant à -1 mais rejette la confiance complète ou confiance aveugle correspondant à 1. Pour Marsh, la confiance aveugle n'est pas incluse dans la notion de confiance mais correspond à la notion de *confidence* définie par [Luhmann, 1990]. Pour [Luhmann, 1990], le terme *confidence* est utilisé au lieu de la confiance lorsque la situation n'est pas risquée, qu'il n'y a pas la possibilité d'être déçu, et que donc aucun choix n'a à être fait. Marsh définit trois types de confiance :

12. Depuis 2014, le site Epinions ne permet plus les inscriptions, les connexions et les notations. Seule la visualisation des produits est possible.

- La confiance basique constitue la disposition générale d'un utilisateur [13] u à faire confiance, elle est notée $\texttt{confiance}(u) \in [-1,1[$.
- La confiance générale est la confiance d'un utilisateur u envers un utilisateur v sans prendre en compte une situation spécifique, elle est notée $\texttt{confiance}(u,v) \in [-1,1[$. Cette connaissance implique la connaissance de l'autre agent.
- La confiance de situation est la confiance d'un utilisateur u envers un utilisateur v dans une situation donnée α, elle est notée $\texttt{confiance}(u,v,\alpha) \in [-1,1[$. Pour la confiance de situation, Marsh utilise la définition de [Rempel et Holmes, 1986], où il affirme que la confiance est dirigée envers les individus et dans des situations spécifiques. Marsh propose une méthode pour calculer la confiance de situation de u envers v dans une situation donnée en prenant en compte l'importance de la situation, c'est-à-dire si l'utilisateur a beaucoup de chances d'obtenir des bénéfices, et l'utilité de la situation, c'est-à-dire, ce que la situation va lui apporter.

Dans son modèle, un agent peut coopérer avec un autre si sa confiance de situation est supérieure à un seuil de coopération. Marsh calcule le seuil de coopération en prenant en compte les risques perçus de la situation, les compétences perçues de l'autre agent dans la situation, un facteur d'oubli et la réciprocité.

Remarque : Le modèle de Marsh représente une référence pour les systèmes informatiques manipulant la notion de confiance. En effet, beaucoup de modèles dont ceux présentés en sections 1.4.3, 1.4.4, 1.4.5, 1.4.6 et 1.4.7 utilisent les notions introduits par Marsh.

1.4.3 Modèles pour les systèmes de filtrage collaboratif

Les systèmes de filtrage collaboratif sont des systèmes qui recommandent des objets du domaine d'application, par exemple des films ou des livres, aux utilisateurs du système. La recommandation d'un objet pour un utilisateur u peut se faire en sélectionnant les utilisateurs similaires à u et en utilisant les évaluations de ces utilisateurs similaires. Dans certains systèmes, la sélection des utilisateurs similaires peut reposer sur la confiance.

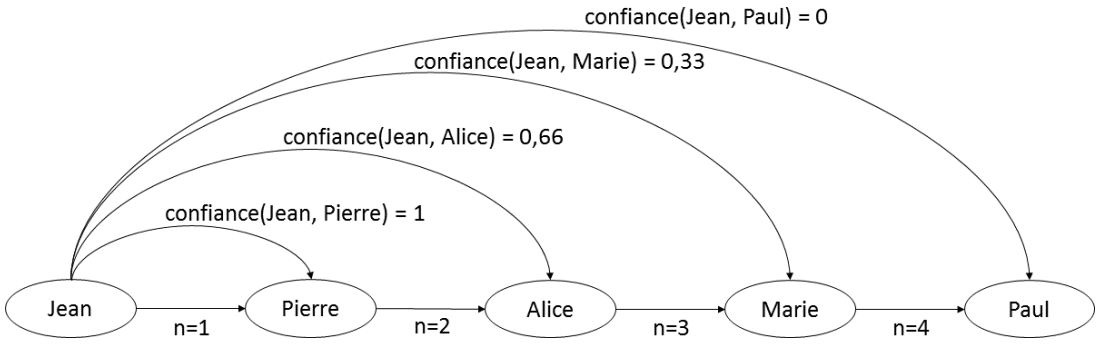


FIGURE 1.2 – Illustration du modèle MoleTrust avec $d = 3$.

13. Bien que le modèle de Marsh se place dans le courant des systèmes multi-agents, nous utilisons ici le terme « utilisateur » par souci d'homogénéisation avec le reste du document.

MoleTrust. Le modèle MoleTrust [Massa et Avesani, 2004] est un modèle fondé sur la confiance destiné aux systèmes de recommandation utilisant les données d'Epinions (cf. section 1.4.1). Nous rappelons que sur le site Epinions, un utilisateur u peut ajouter un utilisateur v a son réseau de confiance. Dans le modèle, la confiance de u envers v est définie par :

$$\texttt{confiance}(u,v) = \begin{cases} (d-n+1)/d & \text{si } d \geqslant n \\ 0 & \text{sinon.} \end{cases} \tag{1.1}$$

n est la longueur du chemin permettant de lier u et v, si u est directement lié à v, c'est-à-dire si u a ajouté v à son réseau de confiance, alors $n = 1$, si u et v sont liés par l'intermédiaire de w alors $n = 2$, etc. d est la longueur maximale autorisée du chemin entre u et v (c'est-à-dire, le nombre maximal d'arcs permettant de relier u à v, avant que la confiance entre les deux ne soit nulle). Par exemple, avec $d = 3$, si u est relié à v par un chemin de longueur 4 ou plus grand que 4, alors la confiance de u envers v est de 0. Si u et v sont reliés par plusieurs chemins différents, le plus court chemin est utilisé pour calculer la confiance que u a envers v.

La figure 1.2 présente un exemple de comment le score de confiance de Jean est calculé pour les utilisateurs Pierre, Alice, Marie et Paul étant donné les relations suivantes : Jean a ajouté Pierre à son réseau de confiance, Pierre a ajouté Alice à son réseau de confiance, Alice a ajouté Marie à son réseau de confiance et Marie a ajouté Paul à son réseau de confiance. Le score de confiance est calculé avec $d = 3$, ainsi la confiance de Jean envers Paul est nulle.

Remarque : le modèle ne prend pas en compte le contexte pour représenter la confiance. En effet, le site Epinions regroupe des produits de catégories diverses, par exemple des produits informatiques et des produits électroménagers. Un utilisateur u pourrait avoir confiance en l'utilisateur v pour les produits informatiques mais pas pour les produits électroménagers. Par ailleurs, un utilisateur peut seulement indiquer qu'il a une confiance absolue envers un autre utilisateur en le mettant dans son réseau de confiance, il ne peut pas indiquer un niveau de confiance graduel.

TidalTrust. Golbeck [Golbeck, 2005] propose le modèle TidalTrust testé sur le site FilmTrust [14] de recommandations de films regroupant un réseau social de 400 utilisateurs. Un utilisateur peut d'une part évaluer un film avec 8 valeurs discrètes comprises entre une demi-étoile et 4 étoiles, et d'autre part évaluer un autre utilisateur du réseau social avec 10 valeurs entières dans l'intervalle [1,10]. Golbeck considère qu'utiliser des valeurs discrètes est plus pertinent qu'utiliser un intervalle continu pour collecter des valeurs de confiance.

TidalTrust permet de calculer le score de recommandation d'un film m pour un utilisateur u qui n'a jamais évalué m. Si u n'a jamais évalué d'utilisateurs qui ont évalué m, TidalTrust permet de calculer la confiance de u envers les utilisateurs qui ont évalué m. Pour calculer le score de confiance de u envers un utilisateur v ayant évalué m, TidalTrust procède par plusieurs étapes :

- Le système cherche les évaluateurs de m avec qui u est directement connecté, c'est-à-dire les utilisateurs qu'il a directement évalués, et qu'il a évalués avec une note plus grande ou égale à un seuil fixé, 7 par exemple.
- S'il n'y a pas de connexions directes entre u et les évaluateurs de m, le système cherche les utilisateurs connectés avec la source par un chemin de longueur 2, c'est-dire les utilisateurs directement connectés avec les utilisateurs directement connectés à u. Le processus continue à chercher en augmentant de 1 la longueur du chemin à chaque itération, jusqu'à ce qu'un chemin soit trouvé. La recherche s'arrête lorsqu'au moins un utilisateur a été trouvé dans le réseau social.

14. http ://trust.mindswap.org/FilmTrust/

– Le score de confiance est calculé à partir de tous les utilisateurs compris dans le chemin trouvé.

La confiance entre un utilisateur u est un utilisateur w est représentée par :

$$\texttt{confiance}(u,w) = \frac{\sum_{v \in adj(u), \texttt{confiance}(u,v) \geqslant c_{max}} \texttt{confiance}(u,v).\texttt{confiance}(v,w)}{\sum_{v \in adj(u), \texttt{confiance}(u,v) \geqslant c_{max}} \texttt{confiance}(u,v)}$$

où $adj(u)$ représente les utilisateurs directement connectés à u et c_{max} est le seuil de confiance permettant de sélectionner les utilisateurs envers qui u a le plus confiance.

Pour l'utilisateur u, le score de recommandation $\texttt{recommandation}(u,m)$ pour l'utilisateur u à propos du film m est défini par :

$$\texttt{recommandation}(u,m) = \frac{\sum_{v \in S, \texttt{confiance}(u,v) \geqslant c_{max}} \texttt{confiance}(u,v).\texttt{recommandation}(v,m)}{\sum_{v \in S} \texttt{confiance}(u,v)}$$

où S est l'ensemble des utilisateurs retrouvés comme étant les utilisateurs les plus proches connectés à u ayant évalués m et c_{max} est le seuil de confiance permettant de sélectionner les utilisateurs envers qui u a le plus confiance.

Remarque : contrairement à MoleTrust, une confiance graduelle peut être exprimée et un seuil est appliqué pour sélectionner les utilisateurs du réseau de confiance. Comme pour MoleTrust, seul le chemin le plus court est utilisé. Nous verrons par la suite que certains modèles utilisent un ensemble de chemins de longueur variable.

Modèle d'Abdul-Rahmann. Dans le modèle de [Abdul-Rahman et Hailes, 2000b], un utilisateur u utilise les recommandations d'autres utilisateurs à propos d'un utilisateur v, par exemple un auteur de livre, qui a produit une information ou un objet, par exemple un livre. Les auteurs déterminent la fiabilité d'un utilisateur u envers un utilisateur v dans un contexte donné en fonction des expériences directes de u avec v et des recommandations des autres utilisateurs faites à u à propos de v.

Dans le modèle, la confiance est modélisée selon différentes propriétés :

– la confiance est dépendante du contexte.
– la confiance peut être représentée par des valeurs discrètes négatives et positives : 4 valeurs dans le modèle : très fiable - fiable - pas fiable - pas du tout fiable.
– la confiance est fondée sur des expériences.
– la confiance n'est pas transitive : pour représenter la fiabilité d'un utilisateur u envers un utilisateur v, seules les recommandations des utilisateurs ayant eu une expérience avec v sont prises en compte.
– la confiance est subjective : plusieurs utilisateurs peuvent avoir une perception différente de la fiabilité d'un utilisateur v.
– la confiance est dynamique : de nouvelles expériences ou recommandations vécues par l'utilisateur u peuvent faire augmenter ou décroître son niveau de confiance envers un autre utilisateur.

Chaque agent du système dispose de deux conteneurs d'expériences :

- le conteneur d'expériences directes stockant pour chaque utilisateur u du système et chaque contexte c un quadruplet $s = (s_{tf}, s_f, s_{pf}, s_{ptf})$ où s_{tf} est le nombre d'évaluations ayant reçu la valeur « très fiable », s_f est le nombre d'évaluations ayant reçu la valeur « fiable », s_{ptf} est le nombre d'évaluations ayant reçu la valeur « pas fiable » et s_{ptf} est le nombre d'évaluations ayant reçu la valeur « pas du tout fiable ».
- le conteneur d'expériences de recommandations stockant pour chaque utilisateur u du système et chaque contexte c quatre ensembles T_{sf}, T_f, T_{pf} et T_{ptf} contenant des valeurs de l'ensemble $G = \{-3; -2; -1;\ 0;\ 1;\ 2;\ 3\}$. Les valeurs de G représentent les écarts de perception entre la perception qu'un utilisateur v, qui a fait une recommandation à u, a d'un utilisateur w et la perception directe que u a de w. Par exemple, si u fait une recommandation à v à propos de w avec la valeur « fiable » et que u connait une expérience directe avec w qu'il évalue avec la valeur « pas fiable », alors la distance de perception entre la recommandation de v et l'expérience directe que u a réellement vécue aura la valeur 1.

Dans le modèle, si par exemple, un utilisateur u veut connaître la fiabilité d'un auteur w de livres de science fiction, u estime la fiabilité qu'il a en w en plusieurs étapes :

1. u obtient les recommandations des utilisateurs qui ont eu une expérience direct avec w.
2. En fonction des expériences de recommandations passées avec ces utilisateurs, u adapte la valeur de ces recommandations.
3. Un poids est attribué à la recommandation adaptée de chaque utilisateur v en fonction des distances de perceptions passées entre les recommandations passées de l'utilisateur v et l'expérience directe de u. Chaque recommandation adaptée de l'agent v se voit attribuer un poids contribuant à la valeur de fiabilité finale que u a envers w en fonction de la valeur de la recommandation adaptée de v et de la valeur de la confiance de recommandation qu'a u en v.
4. L'estimation de la fiabilité que u a envers w correspond à une des valeurs *très fiable*, *fiable*, *pas fiable* et *pas du tout fiable* qui correspond à la valeur de la recommandation ayant reçu le poids le plus fort.
5. La dernière étape consiste à mettre à jour les expériences de recommandations entre u et chaque utilisateur v qui a fait une recommandation ainsi que l'expérience directe entre u et w.

Remarque : la difficulté de ce modèle est l'utilisation de valeurs discrètes limitant les calculs possibles. De plus, comme la transitivité est rejetée, l'estimation de la fiabilité de u envers w n'est possible que si des utilisateurs ayant déjà fait des recommandations à u ont déjà eu une expérience directe avec w. En effet, si un utilisateur v fait une recommandation à u et que v n'a jamais fait de recommandations à u dans le passé, alors cette recommandation sera seulement prise en compte dans la dernière étape mettant à jour les expériences de recommandations. La recommandation sera prise en compte dans le calcul des prochaines estimations de fiabilité.

1.4.4 Modèles utilisant directement la réputation

Lorsqu'un utilisateur u veut interagir avec un utilisateur v, certains modèles représentent directement la valeur de la réputation de v qui est l'avis de la communauté sur cet utilisateur, même si l'utilisateur u a auparavant interagi avec v.

REGRET. Dans le système de commerce électronique REGRET [Sabater et Sierra, 2002], les utilisateurs peuvent coopérer, être en compétition ou faire des transactions en fonction de la réputation des utilisateurs dans le système. La réputation d'un utilisateur est caractérisée par 3 dimensions. La réputation d'un utilisateur v a une dimension individuelle, une dimension sociale et une dimension ontologique.

La dimension individuelle de la réputation de v prend seulement en compte l'expérience directe entre un utilisateur u et v. Par exemple, lorsque u fait une transaction avec v, un contrat est défini entre ces deux utilisateurs comprenant un critère de prix, un critère de qualité et une date de livraison du produit. Le résultat d'une expérience de transaction entre u et v est modélisé en fonction de la différence entre le contrat initial et les résultats de la transaction. Cette différence va permettre à u de classer v dans une catégorie, par exemple, la catégorie « fait payer trop cher ».

La dimension sociale de la réputation de v prend seulement en compte les informations provenant de la communauté. REGRET définit 3 types de dimensions sociales de la réputation de v :

- la réputation provenant des témoins de v, c'est-à-dire des utilisateurs ayant des informations à propos de v.
- la réputation des utilisateurs qui ont déjà interagi avec v et le type d'interactions qu'ont eu ces utilisateurs avec v. Cette dimension fait appel à des règles floues : par exemple, si un utilisateur a interagi avec v par le biais d'une transaction, alors sa réputation aura plus d'impact que s'il a interagi avec v par le biais d'une coopération.
- la réputation provenant du système qui dépend du rôle de v dans le système. Ce type de réputation est utilisé pour les nouveaux utilisateurs.

La dimension ontologique de la réputation de v permet de combiner une réputation prenant en compte plusieurs aspects afin de représenter une réputation complexe. Par exemple, la réputation d'être quelqu'un dans les temps et la réputation d'être quelqu'un pour qui la qualité importe. Chacun des aspects de la réputation est représenté par sa dimension individuelle et/ou sociale.

Remarque : la dimension individuelle modélisée dans REGRET est équivalente à la confiance locale dans la littérature. L'utilisation de la dimension individuelle combinée à la dimension collective permet d'appuyer ou de modérer le point de vue de l'agent qui juge grâce à la communauté. De plus, la dimension ontologique permet d'affiner la valeur de la réputation.

Bêta réputation. [Jøsang et Ismail, 2002] et [Mui *et al.*, 2002] ont réalisé des modèles où la réputation d'un agent est calculée grâce à la densité de probabilité de la loi bêta qui est une famille des lois de probabilités continues.

Dans le modèle de [Jøsang et Ismail, 2002], la réputation d'un utilisateur u est fondée sur l'accumulation d'expériences positives ou négatives que les autres utilisateurs ont eus avec u permettant de mettre à jour la réputation de u après chaque nouvelle expérience. De ce fait, la réputation est vue comme une variable aléatoire représentée grâce à la loi bêta sur l'intervalle [0,1] et déterminée grâce à deux paramètres : α et β.

[Jøsang et Ismail, 2002] définissent une fonction de réputation de u correspondant à la densité de probabilité de la loi bêta :

$$\mathcal{Q}(u) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1}$$

avec $p \in [0,1]$ et Γ la fonction Gamma [Artin, 2015]. α et β correspondent à :

$$\alpha = r + 1$$
$$\beta = s + 1$$

avec r, le nombre d'expériences positives que les autres utilisateurs ont eus avec u et s, le nombre d'expériences négatives que les autres utilisateurs ont eus avec u.

La probabilité qu'une prochaine expérience avec u sera positive est représentée par la valeur de l'espérance de la fonction de réputation et est exprimée par :

$$\begin{aligned} E(u) &= \frac{\alpha}{\alpha + \beta} \\ &= \frac{r+1}{r+s+2} \end{aligned}$$

À partir de la fonction de réputation, [Jøsang et Ismail, 2002] introduisent le score de réputation d'un utilisateur u compris dans l'intervalle [-1,1]. Cet intervalle leur paraît plus intuitif car la valeur neutre est représentée par 0. Le score de réputation est calculé comme suit :

$$\texttt{réputation}(u) = \frac{r-s}{r+s+2}$$

Le score de réputation de u est interprété comme une indication sur l'espérance des futures actions de u.

En plus de la réputation, [Jøsang et Ismail, 2002] représentent l'opinion d'un utilisateur u à propos d'un utilisateur v grâce à la logique subjective. Fondé sur les travaux de [Jøsang, 2001] qui représentent l'opinion d'un utilisateur u à propos d'une proposition x (cf. section 1.2.1), [Jøsang et Ismail, 2002] représentent l'opinion de u à propos des conseils de v par le triplet $\mathcal{W}_v^u = (b_v^u, d_v^u, u_v^u)$ où $b_v^u + d_v^u + u_v^u = 1$ avec $b_v^u, d_v^u, u_v^u \in [0,1]$ qui représentent respectivement la croyance (*belief*), l'incrédulité (*disbelief*) et l'incertitude (*uncertainty*) de u envers les conseils de v et sont calculés comme suit :

$$\begin{aligned} b_v^u &= \frac{r}{r+s+2} \\ d_v^u &= \frac{s}{r+s+2} \\ u_v^u &= \frac{2}{r+s+2} \end{aligned}$$

[Jøsang et Ismail, 2002] proposent de représenter l'opinion transitive de u à propos des conseils de w en utilisant l'opinion de u à propos des conseils de v, notée $\mathcal{W}_w^{u:v} = (b_w^{u:v}, d_w^{u:v}, u_w^{u:v})$ avec :

$$\begin{aligned} b_w^{u:v} &= b_v^u b_w^v, \\ d_w^{u:v} &= b_v^u d_w^v, \\ u_w^{u:v} &= d_v^u + u_v^u + b_v^u u_w^v \end{aligned}$$

Remarque : Contrairement à Ebay, le score de réputation, la croyance, l'incrédulité et l'incertitude du modèle de [Jøsang et Ismail, 2002] prennent en compte le nombre d'interactions. Ainsi, un utilisateur ayant eu beaucoup d'expériences au total dont beaucoup d'expériences négatives ne sera pas favorisé à l'avantage d'un utilisateur ayant eu peu d'expériences mais que (ou presque que) des expériences positives.

1.4.5 Modèles de flux

Certains systèmes calculent la confiance locale ou globale (réputation) par transitivité avec des itérations répétées ou dans de longues chaînes. Ces systèmes sont adaptés pour résister aux attaques de groupes. Dans les systèmes fondés sur la confiance, une attaque de groupe est une alliance d'un groupe d'utilisateurs malveillants ne fournissant pas ou mal les services qui leur sont demandés. L'alliance de ce groupe d'utilisateurs malveillants permet à leurs membres d'augmenter leur réputation dans le système, en s'attribuant mutuellement de bonnes évaluations, et permet de faire diminuer la réputation des autres utilisateurs du système, en leur attribuant de mauvaises évaluations.

EigenTrust. Le modèle EigenTrust [Kamvar *et al.*, 2003] est un des modèles de confiance destinés aux systèmes pair-à-pair (P2P) les plus cités dans la littérature. Les systèmes P2P sont des réseaux de nœuds, où chaque nœud est à la fois client et serveur. Les nœuds s'échangent des objets qui sont le plus souvent des fichiers. EigenTrust permet d'attribuer une réputation (ou un score de confiance globale) à chaque nœud, dans un système P2P d'échange de fichiers, fondé sur l'historique du résultat des échanges de fichiers qu'il a fourni. De plus, EigenTrust permet de protéger le système contre les attaques de groupes de nœuds malveillants ne fournissant pas les fichiers demandés.

Dans un premier temps, le système représente les scores de confiance locale entre un nœud i et un autre nœud j. Tout comme pour le modèle de confiance utilisé sur eBay [Resnick et Zeckhauser, 2002], l'interaction de i avec j peut être positive ou négative. Le nombre d'interactions positives qu'a eu i avec j est noté $sat(i,j)$ et le nombre d'interactions négatives qu'a eu i avec j est noté $unsat(i,j)$. La confiance locale de i en j correspond à la différence entre les interactions positives et les interactions négatives : $c(i,j) = sat(i,j) - unsat(i,j)$.

Cependant, ce calcul de la confiance locale n'est pas résistant à l'attaque d'un groupe de nœuds qui s'allient en se donnant des évaluations positives entre eux et des évaluations négatives aux autres nœuds. Les scores de confiance locaux sont alors normalisés afin de limiter l'impact de ce genre d'attaques. Le score de confiance locale normalisé noté $\texttt{confiance}_{i,j}$ est défini par :

$$\texttt{confiance}_{i,j} = \frac{\max(c(i,j),0)}{\sum_{k \in L_i} \max(c(i,k),0)} \in [0,1]$$

avec L_i, l'ensemble des nœuds qui ont eu une expérience directe avec i.

Le score de confiance de i envers j peut aussi être calculé à partir d'un nœud k que i connaît, c'est-à-dire, un nœud k que i a déjà évalué.

$$\texttt{confiance}_{i,j} = \sum_{k \in L_i, \texttt{confiance}_{k,j} \neq \texttt{?}} \texttt{confiance}_{i,k} \texttt{confiance}_{k,j}$$

où $\texttt{confiance}_{k,j} \neq \texttt{?}$ signifie que k a déjà interagi avec j.

Finalement, le vecteur colonne $\texttt{confiance}_i = \begin{bmatrix} \texttt{confiance}_{i1} \\ \texttt{confiance}_{i2} \\ \cdots \\ \texttt{confiance}_{im} \end{bmatrix}$ contenant les scores de confiance de i envers les m nœuds du système calculés à partir des connaissances de u, peut être représenté

grâce à la matrice $C = [\texttt{confiance}]_{im}$ contenant les scores de confiance locale normalisés :

$$\texttt{confiance}_i = C^T c_i$$

où C^T est la transposée de C et c_i est le vecteur ligne [$\texttt{confiance}_{i1}$,$\texttt{confiance}_{i2}$,...,$\texttt{confiance}_{im}$] des scores de confiance locale normalisé.

Afin d'avoir une vision encore plus globale et interroger les connaissances des connaissances de i, le vecteur colonne $\texttt{confiance}_i$ peut être défini par $\texttt{confiance}_i = (C^T)^2 c_i$. Le processus peut encore continuer avec $\texttt{confiance}_i = (C^T)^n c_i$. Finalement, plus n est grand, plus $\texttt{confiance}_i$ va converger vers le même vecteur colonne. Ainsi, après une longue chaîne d'itérations, chaque vecteur de confiance $\texttt{confiance}_x$ avec $x \in [1,m]$ converge vers le même vecteur colonne contenant la réputation de chaque nœud dans le système.

Remarque : les limites de ce modèle correspondent au fait que le score de confiance normalisé supprime les scores de satisfactions négatives en utilisant le maximum entre 0 et le score de confiance non normalisé, ce qui entraîne qu'un nouvel arrivant aura la même réputation qu'un mauvais nœud.

Google PageRank L'algorithme PageRank de Google [15] [Page et al., 1998], permet d'attribuer un score, le *PageRank*, aux pages Web. Ce score contribue à l'algorithme de classement des pages Web lors d'une recherche Google. L'algorithme PageRank repose sur les liens hypertextes entre pages Web, où un lien hypertexte est vu comme une relation de confiance. Le *PageRank* d'une page p mesure la probabilité qu'un internaute se trouvant sur une page donné, arrive sur cette page p. Cet algorithme repose sur l'hypothèse qu'un internaute, qui se retrouve sur une page, choisit aléatoirement de suivre un lien de cette page.

L'attribution du *PageRank* d'une page est récursive et repose sur trois principes :

- plus une page a un *PageRank* important plus elle contribue au *PageRank* des pages vers lesquelles elle pointe ;
- plus une page contient de liens, moins elle contribue au *PageRank* des pages vers lesquelles elle pointe ;
- plus une page reçoit de liens, plus son *PageRank* est important.

Le *PageRank* d'une page p, noté $R(p)$, est calculé somme suit :

$$R(p) = c \sum_{q \in pr\acute{e}d(p)} \frac{R(q)}{N_q}$$

où $pr\acute{e}d(p)$ correspond à l'ensemble des pages qui pointent vers p, N_q est le nombre de liens que contient la page q et c est un facteur de normalisation.

Remarque : l'algorithme Google PageRank est résistant aux attaques de groupes. En effet, même si une page p est beaucoup citée, mais que les pages qui la citent ne sont pas elles-mêmes citées par une page ayant un *PageRank* important, alors p n'aura jamais un *PageRank* important.

Advogato Advogato [16] est un site Web réunissant une communauté de développeurs de logiciels libres [Levien, 2009]. Chaque membre de la communauté peut attribuer un niveau de certification à un autre membre. Trois niveaux de certification sont possibles : apprenti, amateur

15. `http://google.com/`
16. `http://advogato.org/`

et maître. Ces trois niveaux de certification permettent de construire trois différents réseaux de confiance. Chacun des réseaux de confiance construit à partir d'un des trois niveaux de certification correspond à un graphe orienté où les nœuds sont les membres de la communauté et les arcs sont les liens de certification du niveau de certification correspondant.

L'algorithme permettant de modéliser la confiance dans Advogato permet de modéliser la confiance d'un nœud source s envers chacun des autres nœuds d'un graphe donné. L'algorithme prend en entrée ce nœud source s et le graphe de nœuds correspondant à un niveau de certification. L'algorithme retourne pour chaque nœud du système, une valeur `vrai` ou `faux` signifiant que s a ou n'a pas confiance en ce nœud pour le niveau de certification correspondant.

L'algorithme attribue récursivement une capacité à chaque nœud du graphe. Une capacité est un score entre 1 et m, le nombre de nœuds du graphe. Le nœud source s se voit attribuer une capacité de m. L'algorithme repose sur les mêmes principes que l'algorithme de Google PageRank :

- plus un nœud a une capacité importante, plus il contribue à la capacité de ses successeurs ;
- plus un nœud a de successeurs, moins il contribue à la capacité de ses successeurs ;
- plus un nœud a de prédécesseurs, plus sa capacité est importante.

La capacité d'un nœud i, notée $\mathit{capacité}(i)$, est calculée comme suit :

$$\mathit{capacité}(i) = \begin{cases} \displaystyle\sum_{j \in \mathit{pred}(i)} \frac{\mathit{capacité}(j)}{N_j} & \text{si } i \neq s \\ m & \text{sinon.} \end{cases}$$

où $\mathit{préd}(i)$ correspond à l'ensemble des nœuds prédécesseurs de i et où N_j est le nombre de successeurs de j.

Si l'algorithme de confiance a retourné que le nœud source s avait confiance en le nœud i pour le niveau apprenti et le niveau maître, alors s a confiance en i en tant que maître. En revanche, si l'algorithme de confiance a seulement retourné que s avait confiance en le nœud i pour le niveau apprenti, alors s a seulement confiance en i en tant qu'apprenti.

Remarque : tout comme l'algorithme de Google PageRank, l'algorithme qui modélise la confiance pour le site Advogato est résistant aux attaques malveillantes de groupes. De plus, même si la sortie de l'algorithme de confiance est binaire, la construction des trois réseaux de confiance grâce aux trois niveaux de certification permet de graduer la confiance.

AppleSeed AppleSeed [Ziegler et Lausen, 2004] est un modèle qui utilise une méthode de propagation de l'énergie pour calculer la confiance entre utilisateurs. Le modèle s'applique à un système dans lequel les utilisateurs se notent entre eux. Les nœuds sont les utilisateurs et un arc entre un nœud i et un nœud j existe si i a noté j dans le système. La valeur sur l'arc entre i et j est un poids correspondant à la note que i a attribué à j normalisée entre 0 et 1. Le modèle permet de représenter la confiance entre un nœud source s et un nœud cible. Une somme d'énergie est injectée dans le nœud source s et se répand à travers les nœuds successifs de ce nœud source avec une perte d'énergie. La confiance entre un nœud i et un nœud j est équivalente à l'énergie envoyée par i et reçue par j :

$$e_{i \to j} = d.in(i).\frac{W(i,j)}{\displaystyle\sum_{(i,k) \in E} W(i,k)}$$

d correspond à un facteur de diffusion compris entre 0 et 1 où $d = 1$ signifie qu'il n'y a pas de perte d'énergie et $d = 0$ signifie que toute l'énergie de i est dissipée avant d'arriver à j ; in(i) correspond à l'énergie reçue par le nœud i ; $W(i,j)$ correspond à la valeur de l'arc reliant i à j et $\sum_{(i,k)\in E} W(i,k)$ correspond à la somme des valeurs de tous les arcs partant de i.

Remarque : comme pour l'algorithme de PageRank, dans le modèle AppleSeed, plus un utilisateur accorde sa confiance, moins l'attribution de sa confiance aura d'impact. Ainsi, l'acte de confiance d'un utilisateur de nature méfiante sera privilégié par rapport à l'acte de confiance d'un utilisateur de nature confiant. Cette façon de faire est discutable, car par exemple un utilisateur méfiant peut être un utilisateur malhonnête et un utilisateur confiant peut être un utilisateur honnête (cf. l'effet de réciprocité [Marsh, 1994]). Cependant, les valeurs sur les arcs permettent de privilégier certains utilisateurs de son réseau de confiance par rapport aux autres.

1.4.6 Modèles représentant la fiabilité d'une information

TRELLIS. TRELLIS [Gil et Ratnakar, 2002] est un système collaboratif permettant aux utilisateurs d'ajouter des annotations sur une analyse de documents en ligne. Les documents du système sont ajoutés par les utilisateurs du système et chaque document est lié à la description de sa source : l'auteur, l'éditeur, etc. Un utilisateur peut annoter une affirmation issue d'une source en créant une ou plusieurs affirmations qui lui permet d'introduire des hypothèses, des observations ou des conclusions sur l'affirmation annotée ou sa source. De plus, un utilisateur peut associer à une affirmation et à sa source, un score de fiabilité et de crédibilité sur une échelle de 1 à 6. Le score de fiabilité est établi en fonction des précédentes expériences de l'utilisateur avec la source tandis que le score de crédibilité est établi en fonction de l'analyse des autres informations disponibles à l'utilisateur lors de l'annotation.

Chaque annotation d'une affirmation a liée à une source s permet de calculer un score de qualité au couple (s,a). Ce score prend en compte le score de fiabilité et le score de crédibilité associé à a et s, si a a été utilisée ou non dans la conclusion de l'annotation, et si a a été critiquée. Finalement, la moyenne des scores de qualité de tous les couples (s,a_i) pour la même source s représente la confiance que les utilisateurs du système peuvent avoir en la source s.

Remarque : TRELLIS modélise la confiance des utilisateurs envers une source d'information, par l'intermédiaire de la qualité des affirmations issues de cette même source. Les scores de fiabilité et de crédibilité d'une affirmation utilisés pour calculer la qualité de l'affirmation d'une source ne sont pas calculés mais donnés par l'utilisateur qui annote l'affirmation et sont justifiés par des affirmations. Cependant, la justification par des annotations est discutable, car ces affirmations peuvent ne pas être fiables.

Modèle de Richardson. [Richardson *et al.*, 2003] proposent un modèle appliqué dans un réseau social pour prédire la croyance qu'un utilisateur a envers une affirmation. Dans le réseau social, un utilisateur u est connecté à un utilisateur v si u a déjà évalué v par un score de confiance compris entre 0 et 1. Les utilisateurs du réseau social peuvent déclarer des affirmations qui peuvent être évaluées par les autres utilisateurs avec des scores de croyance compris entre 0 et 1. Pour estimer la croyance d'un utilisateur u à propos d'une affirmation a, tous les chemins reliant u à un utilisateur ayant déjà attribué un score de croyance à a sont retrouvés. Un score de croyance est associé à chaque chemin grâce à une fonction de concaténation sur tous les scores de confiance du chemin et le score de croyance à la fin du chemin. Le score de croyance pour

l'utilisateur u à propos de l'affirmation a est calculé grâce à une fonction d'agrégation (minimum, maximum ou moyenne) appliquée aux scores de croyance associés à chaque chemin trouvé.

Dans l'exemple de la figure 1.3, u attribue un score de confiance de 1,0 à v, u attribue un score de confiance de 0,9 à w, v attribue un score de confiance de 0,5 à w et w attribue un score de confiance de 0,7 à x, v attribue un score de croyance de 0,7 à l'affirmation s et x attribue un score de croyance de 0,8 à s.

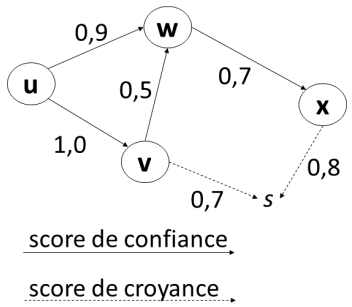


FIGURE 1.3 – Exemple de réseau social pour modéliser le modèle de Richardson et al.

3 chemins sont retrouvés et leur score de croyance associé est calculé par multiplication :

1. $u \to v \to s$: $1{,}0 \times 0{,}7 = 0{,}7$
2. $u \to w \to x \to s$: $0{,}9 \times 0{,}7 \times 0{,}8 = 0{,}504$
3. $u \to v \to w \to x \to s$: $1{,}0 \times 0{,}5 \times 0{,}7 \times 0{,}8 = 0{,}28$

Si le maximum est utilisé comme fonction d'agrégation, alors le score de croyance prédit pour u à propos de s est de 0,7.

Remarque : ce modèle permet de comparer et discuter différentes méthodes mathématiques de calcul de la confiance et de la croyance. L'agrégation par la moyenne, le minimum et le maximum est notamment discutée. Ce modèle peut ainsi servir de justification à d'autres modèles.

Wikipédia. [Zeng *et al.*, 2006] ont proposé un système pour l'encyclopédie en ligne de Wikipédia [17]. Ce système repose sur les réseaux dynamiques bayésiens pour représenter la fiabilité d'un article. Un nœud du réseau correspond à une version i d'un article. Ce système satisfait la propriété de Markov selon laquelle l'état de l'article i dépend de l'état de l'article $i-1$. La fiabilité de la nouvelle version d'un article, comprise entre 0 et 1, dépend alors de la version précédente, de l'auteur de la nouvelle version, de la quantité de texte ajouté et de la quantité de texte supprimé. Les auteurs ajoutent aussi la notion d'incertain à la fiabilité en utilisant la Bêta distribution.

1.4.7 Modèles fondés sur la croyance

Dans certains modèles, la confiance d'un agent envers un autre agent est formée autour d'un ensemble de croyances.

17. `www.wikipedia.org/`

Modèle de Castelfranchi et Falcone. Le modèle de [Castelfranchi et Falcone, 1998] est destiné aux systèmes multi-agents. Le modèle représente la confiance d'un agent cognitif x envers un autre agent y. Un agent cognitif est défini comme un système biologique autonome doué de croyances, de buts et de désirs. L'agent y peut être un système autonome mécanique, biologique ou logiciel. La confiance est caractérisée par un ensemble d'attitudes mentales caractérisant l'état d'esprit de x qui délègue l'atteinte d'un but g par l'intermédiaire des actions de y. Cet ensemble d'attitudes mentales est caractérisé par différentes croyances de l'agent x attribuant sa confiance à y pour atteindre le but g :

- la croyance envers les compétences de y : x croit que y peut atteindre g ;
- la croyance envers les dispositions de y : x croit que y atteindra g ;
- la croyance de dépendance en y : x croit qu'il peut dépendre de y pour atteindre g ;
- la croyance envers l'atteinte de g : x croit que g peut être atteint ;
- la croyance envers la volonté de y : x croit que y a la volonté d'atteindre g ;
- la croyance de persistance de y : x croit que y est stable, que y ne changera pas d'avis lors de l'atteinte de g ;
- la croyance envers l'estime de soi de y : x croit que y a confiance en lui-même pour atteindre g.

Les 3 dernières croyances participent à la formation de la confiance seulement si l'agent y est lui aussi un agent cognitif.

Remarque : ce modèle est un modèle théorique présentant la confiance entre deux agents. Il ne présente pas les modulations de la confiance lorsqu'un agent tiers intervient dans la relation de confiance. De plus, ces croyances sont des croyances qui n'ont pas d'existence en dehors de l'esprit de l'agent qui accorde sa confiance.

Modèle de Knap. Dans leur modèle destiné aux réseaux sociaux, [Knap et Mlynková, 2011] représentent la confiance dans un domaine spécifique en quantifiant des regroupements de croyance. La confiance d'un utilisateur u envers un utilisateur v dans le contexte c dépend de 9 regroupements de croyance :

- affinité : similarité de v par rapport à u ;
- compétence : niveau de compétence de v dans le contexte c ;
- expérience : la qualité des anciennes expériences que u a eu avec v ;
- expertise : le niveau d'expertise de v dans le contexte c ;
- pratique : nombre d'interventions de v dans le contexte c ;
- réputation : jugement de v par les autres utilisateurs du réseau social ;
- honnêteté : habitudes sur le respect des engagements de v ;
- fiabilité : v à l'habitude de tenir ses engagements ;
- bonne volonté : implication/motivation de v.

Remarque : ce modèle est un modèle prenant en compte un ensemble important de croyances pour représenter la confiance. Cependant, certaines croyances sont très difficiles à modéliser et à quantifier avec des utilisateurs réels, telle que la croyance en la bonne volonté d'un utilisateur. De plus, certaines croyances sont interdépendantes, comme l'honnêteté et la fiabilité.

1.4.8 Conclusion sur les modèles gérant la confiance

Nous avons présenté un certain nombre de modèles et de systèmes fondés sur la confiance. La confiance est souvent dépendante de la croyance, de la qualité, de la réputation et/ou de la fiabilité. Pour la suite de cette thèse, nous nous inspirons tout particulièrement de :

- la section 1.4.3 qui présente des systèmes de filtrage collaboratif fondés sur la confiance. En effet, les systèmes de filtrage collaboratif et les systèmes de RÀPC proposent tous deux à un utilisateur, un ensemble de réponses répondant à ses critères ou à sa requête. Nous pouvons alors nous inspirer de ces systèmes pour calculer la confiance entre utilisateurs de la communauté en ligne produisant les UC utilisées par un système de RÀPC.
- la section 1.4.4 qui présente des systèmes qui calculent la réputation des utilisateurs de leur système car nous voulons représenter la perception collective d'un utilisateur de la communauté en ligne sur la qualité des UC qu'il peut générer.
- la section 1.4.6 qui présente des systèmes qui calculent la fiabilité d'une information pour un utilisateur car de la même façon que ces systèmes représentent la fiabilité des informations, nous voulons représenter la fiabilité d'une unité de connaissances utilisée dans un système de RÀPC.

1.5 Catégorisation des connaissances

1.5.1 Les différentes visions de la catégorisation des connaissances

[Smith et Medin, 1981] présentent trois types de modèles de classification des classes d'un domaine de connaissances : le point de vue classique, le point de vue probabiliste et le point de vue de l'exemplaire. Le point de vue classique provenant d'Aristote affirme que toutes les instances d'une classe partagent les propriétés communes qui sont nécessaires et suffisantes pour définir la classe. Les deux autres points de vue sont plus souples et permettent de représenter des classes « naturelles » qui ne peuvent pas forcément être classifiées selon des critères précis.

1.5.2 Le point de vue probabiliste et le point de vue de l'exemplaire

Utilisant la notion de ressemblance de famille dans les classes, [Beckner, 1959] définit la notion de classes polythétiques, qui contrastent avec la notion de classes monothétiques où les membres d'une classe partagent tous les propriétés nécessaires et suffisantes à la classe. Les membres d'une classe polythétique partagent un large ensemble de propriétés communes mais non nécessaires, et les propriétés sont partagées par une majorité des membres de la classe. Par exemple, pour la classe `Oiseau`, la propriété « peut voler » est une propriété partagée par beaucoup de membres de la classe `Oiseau`, mais n'est pas partagée par tous les membres ; une autruche, qui est un oiseau, n'a pas cette propriété. En revanche, la propriété « peut voler » est caractéristique ou saillante pour la classe `Oiseau`. De cette manière, un membre de la classe `Oiseau` est typique, c'est-à-dire, est un « bon » exemple de `Oiseau`, s'il possède cette propriété saillante. Cette sorte de membre est nommée *prototype* de `Oiseau`. Dans le point de vue probabiliste, une classe a un prototype unique.

Le point de vue de l'exemplaire [Medin et Schaffer, 1978,Bareiss *et al.*, 1990] est plus flexible car une classe accepte des points de vue multiples qui sont des exemples réels de la classe. Par exemple, les classes `RougeGorge` et `Hirondelle` peuvent être considérées comme étant les sous-classes les plus représentatives de la classe `Oiseau`.

1.5.3 La typicalité

La notion de typicalité est liée ici à l'idée que les membres de la classe ne sont pas tous de « bons » exemples. Pour les psychologues Rosch et Mervis [Rosch et Mervis, 1975b], la typicalité d'une instance de la classe `A` dépend de sa similarité avec les autres instances de `A` et de sa dissimilarité avec les instances des autres classes. Pour eux, une baleine est considérée comme étant atypique de la classe `Mammifère`, non pas seulement parce qu'une baleine n'est pas très similaire aux autres mammifères, mais aussi parce qu'une baleine partage des propriétés avec les instances de la classe `Poisson`. [Barsalou et Sewell, 1985], pour leur part, affirment que la typicalité d'une instance d'une classe dépend de sa similarité avec les autres instances de la classe, et à quelle fréquence les gens ont associé cette instance comme un membre de la classe. Selon [Cruse, 1990] les éléments typiques d'une catégorie partagent des propriétés communes saillantes et peu de propriétés inhabituelles. Pour certains auteurs le contexte a une influence sur le jugement de la typicalité d'une instance par rapport à sa classe. [Roth et Shoben, 1983] montrent qu'en fonction du contexte linguistique, les instances les plus typiques d'une classe sont différentes. Dans leurs expériences, lorsqu'il a été demandé de classer des instances de la classe `Animal` dans le contexte de la traite, la vache et la chèvre ont été classées comme les animaux les plus typiques alors que dans le contexte des excursions à dos d'animaux, le cheval et la mule ont été classés comme les animaux les plus typiques. [Barsalou et Sewell, 1984], quant à eux, ont démontrés dans d'autres expériences que la nationalité ou la profession des sujets jugeant les différentes classes avaient une influence. Dans leurs expériences, les américains jugent que le merle et l'aigle sont les instances les plus typiques de la classe `Animal`, alors que pour les chinois ce sont le cygne et la paon. Pour ce qui est de la profession, par exemple, un garagiste aura un point de vue différent sur les instances de la classe `Voiture` qu'un « non expert ».

Un degré de typicalité peut-être associé à un couple (a,`C`) où a est une instance de la classe `C`. L'instance la plus typique de la classe est le prototype de la classe.

Le point de vue probabiliste affirme qu'une classe est représentée par un seul prototype abstrait. Les propriétés partagées par les instances de la classe sont plus ou moins saillantes. Chaque propriété est associée à un poids qui correspond à la probabilité qu'une instance donnée de la classe partage cette propriété. Le degré de typicalité d'une instance pour une classe dépend de sa similarité avec le prototype représenté par un vecteur de propriétés. Dans les travaux présentés par [Hampton, 1979], une classe est décrite par un ensemble de propriétés et un poids est attribué à chaque propriété. Ce poids compris entre 0 et 1 représente la saillance de la propriété pour la classe. Par exemple, pour la classe `Oiseau`, le vecteur de propriétés du prototype peut être (`peut_voler`, `peut_chanter`, `a_des_ailes`, `est_ovipare`) et chaque propriété est associée à un poids entre 0 et 1, par exemple (0,9; 0,5; 1,0; 1,0). L'instance d'une classe est décrite par un ensemble de propriétés associées à un degré d'appartenance compris entre -1 et 1. La typicalité d'une instance a par rapport à sa classe `C` est calculée par :

$$S(a,C) = \sum_{i=1}^{n} w_{P_i} v(P_i,a) \tag{1.2}$$

où P_i est une propriété de la classe `C`, w_{P_i} est la saillance de la propriété P_i et $v(P_i,a)$ est le degré d'appartenance de P_i pour a. $S(a,C)$ est compris entre $-n$ et n où n est le nombre de propriétés différentes. Dans l'exemple, la typicalité d'une instance a par rapport à `Oiseau` est compris entre -4 et 4. Si a est un canard, alors son vecteur de propriété pourrait être $(0{,}2; -0{,}3; 1{,}0; 1{,}0)$ et donc $S(a,Oiseau) = w_{\texttt{peut\_voler}} \times 0{,}2 + w_{\texttt{peut\_chanter}} \times -0{,}3 + w_{\texttt{a\_des\_ailes}} \times 1{,}0 + w_{\texttt{est\_ovipare}} \times 1{,}0 = 2{,}03$.

Les travaux de [Yeung et Leung, 2006] sont similaires à ceux de Hampton. La seule différence est que Yeung et Leung représentent le degré d'appartenance entre 0 et 1 et non plus ente -1 et 1, et que la typicalité est normalisée entre 0 et 1 en ajoutant une division par la somme des poids des propriétés.

[Smith *et al.*, 1988] proposent une théorie similaire mais le prototype n'est pas représenté par des propriétés mais par des attributs associés à des poids entre 0 et 1. Chaque attribut d'un prototype est aussi associé à une liste de valeurs possibles, où on attribue un score à chaque valeur. Ce score correspond à un nombre de votes donnés par des utilisateurs. Par exemple, pour la classe `Oiseau`, l'attribut `moyen_de_locomotion` a un poids de 0,8 et la valeur `vole` a un score de 26, `marche` a un score de 3 et `nage` a un score de 1 sur 30 votes d'utilisateurs. De la même façon que pour le prototype, les valeurs des attributs de chaque instance reçoivent un score. La similarité entre le prototype P et une instance a dépend du nombre de votes communs entre P et a et du nombre de votes distincts entre P et a.

D'autres théories ont étudié la notion de typicalité, notamment la théorie des sous-ensembles flous où une instance appartient à une classe avec un degré d'appartenance compris entre 0 et 1. Pour [Zadeh, 1982], un prototype n'est pas un seul objet mais une classe floue. Pour [Dubois *et al.*, 1991], les classes floues sont décrites par des attributs où les valeurs sont typiques ou non. Pour deux classes `A` et `B`, où `B` est une sous-classe de `A`, trois types de spécialisation de `A` en `B` sont proposées : typique, normale et atypique. Le type de spécialisation dépend de la comparaison de la valeur typique ou non typique des attributs de `A` et `B`.

1.6 Le raisonnement à partir de cas (RÀPC)

1.6.1 Principes du RÀPC

Le RÀPC est un paradigme de résolution de problèmes par analogie qui utilise des expériences passées, les cas sources, pour résoudre un nouveau problème, le problème cible [Schank et Abelson, 1977].

Dans un système de RÀPC, un cas source est classiquement une paire (`srce`, `sol(srce)`) contenue dans une base de cas où `srce` est un problème et `sol(srce)` est une solution connue associée au problème. La résolution d'un problème cible, noté `cible`, consiste à réutiliser la solution `sol(srce)` d'un cas source pour trouver la solution du problème cible, noté `sol(cible)`, et construire un nouveau cas (`cible`, `sol(cible)`). La résolution du problème cible se fait classiquement en deux phases principales. La première phase est la remémoration, qui consiste à trouver le ou les(s) problème(s) source(s), chacun noté `srce`, le ou les plus similaire(s) au problème cible `cible`. La deuxième phase est l'adaptation qui consiste à adapter la solution `sol(srce)` de chaque `srce` pour trouver `sol(cible)`, la solution de `cible`.

La base de connaissances d'un système de RÀPC est habituellement composée de quatre conteneurs de connaissances [Richter, 1995] :

- la base de cas d'où proviennent les cas sources ;
- les connaissances du domaine qui sont les connaissances générales du domaine spécifique et qui sont utilisées durant toutes les phases du raisonnement ;
- les connaissances de similarité permettant de calculer la similarité entre cas sources et problème cible durant la phase de remémoration ;
- les connaissances d'adaptation permettant d'adapter la solution d'un cas source pour obtenir la solution du problème cible.

En RàPC, un cas source peut être représenté selon divers formalismes [Bergmann *et al.*, 2005]. Parmi les plus connus, les représentations par des couples attributs-valeurs [Aha *et al.*, 1991], les représentations orientées objet [Bergmann, 2002], les représentations textuelles [Lenz et Burkhard, 1996, Weber *et al.*, 2005] et les représentations logiques [Plaza, 1995, Cordier *et al.*, 2014]. Dans le cas des représentations logiques, formalisme utilisé par le système de RàPC Taaable [Cordier *et al.*, 2014], les cas de la base de cas ainsi que les connaissances d'adaptation sont décrites grâce aux connaissances du domaine.

Certains systèmes de RàPC utilisent une approche particulière pour la phase de remémoration : la remémoration guidée par l'adaptation introduit par Smyth [Smyth, 1996]. L'étape de remémoration repose sur les connaissances d'adaptation pour calculer la similarité entre les cas sources et le cas cible. Ainsi, pour un cas source donné, moins l'« effort d'adaptation » est important, plus le cas source est similaire au problème cible.

En plus de la phase de remémoration et d'adaptation, d'autres phases facultatives peuvent être utilisées dans un système de RàPC. Certains auteurs tels que [Mille, 1998, Aha et Gupta, 2002], définissent la phase d'élaboration qui précède la phase de remémoration. Cette phase permet de construire un problème cible, dont le format est adapté au système de RàPC, grâce aux connaissances du domaine. De plus, dans certains systèmes de RàPC, plusieurs cas sources sont retrouvés et ainsi plusieurs solutions peuvent être proposées. Une phase de sélection peut être intégrée après la phase de remémoration ou après la phase d'adaptation [Weber-Lee *et al.*, 1996].

Par ailleurs, Aamodt et Plaza [Aamodt et Plaza, 1994] citent d'autres phases comme la phase de révision (*reuse*) et la phase de mémorisation qui sont essentielles dans certains systèmes de RàPC, comme par exemple dans le système CHEF [Hammond, 1990]. La phase de révision correspond aux phases de tests et de réparation. Lorsqu'une solution a été créée après la phase d'adaptation, la phase de test permet de vérifier que la solution proposée est satisfaisante pour l'utilisateur. Si la phase de test signale un échec, la phase de réparation permet de modifier la solution et par la même occasion, de fournir des explications de l'échec qui pourront être stockées avec les connaissances du domaine. La phase de révision est réitérée tant qu'une solution satisfaisante n'est pas trouvée. Si une solution satisfaisante est trouvée, la solution peut être retenue et le nouveau cas (`cible`,`sol(cible)`) est stocké dans la base de cas : ceci correspond à la phase de mémorisation.

1.6.2 Problème d'utilité et qualité des réponses d'un système de RàPC

Le problème de l'utilité [Smyth, 1996] est un problème fondamental en RàPC. Il est défini par la volonté de satisfaire des critères opposés : la qualité des réponses retournées par le système et l'efficacité du système.

La qualité des réponses retournées par un système de RàPC est directement liée aux conteneurs de connaissances, et notamment à la base de cas. La qualité de la base de cas est dépendante de sa taille mais aussi du taux de recouvrement des cas de la base de cas [Smyth et Keane, 1995]. Le taux de recouvrement des cas de la base de cas est calculé à partir du nombre de cas cible que chaque cas peut résoudre.

L'efficacité du système de RàPC correspond au temps moyen de résolution du problème cible. Deux facteurs sont mis en concurrence pour diminuer le temps de résolution : (1) le temps de remémoration pour une taille de base de cas donnée et (2) le temps d'adaptation pour la même taille de base de cas [Smyth, 1996]. Lors de la construction d'un système de RàPC, le choix doit être fait pour privilégier l'un ou l'autre des critères de qualité ou d'efficacité.

Une autre caractéristique du RÀPC impactant la qualité des réponses est que le RÀPC est un raisonnement hypothétique qui génère des réponses qui peuvent être incorrectes. En effet, à la différence d'un raisonnement déductif qui ne peut pas générer de conclusions incorrectes si ses prémisses sont correctes, le RÀPC peut générer des réponses incorrectes même si les connaissances utilisées sont correctes.

1.6.3 Acquisition des connaissances en RÀPC

En RÀPC, l'acquisition des connaissances à travers les différentes phases de raisonnement du RÀPC est un enjeu majeur. Durant la phase de révision, les explications concernant l'échec de la construction de la solution du problème cible peuvent être sauvegardées dans la base de connaissances. Durant la phase de mémorisation, si la solution du cas cible créée est satisfaisante, la paire (`cible`,`sol(cible)`) est retenue dans la base de cas et le cas cible pourra ensuite servir de cas source lors de la résolution d'un nouveau problème. La phase principale d'acquisition de connaissances se déroule durant la création de la base de connaissances du système de RÀPC. Classiquement cette phase d'acquisition est réalisée grâce un expert du domaine permettant d'assurer la validité des connaissances acquises. Cependant d'autres alternatives d'acquisition existent pour remplacer ou suppléer l'expert, et c'est un des enjeux de cette thèse d'examiner certaines de ces alternatives. Certains systèmes utilisent des techniques de fouilles de textes [Ceausu et Desprès, 2005] ou d'extraction de connaissances provenant du web [Gaillard *et al.*, 2011]. Une autre possibilité est d'utiliser une communauté en ligne permettant l'édition collaborative des connaissances [Gaillard *et al.*, 2013a].

1.6.4 Les systèmes de RÀPC utilisant des métaconnaissances

Les métaconnaissances sont déjà utilisées en RÀPC, principalement pour la maintenance de la base de cas et pour les systèmes de recommandations fondés sur des méthodes de RÀPC. [Quijano-Sánchez *et al.*, 2012] proposent un système de recommandations de films destiné à un groupe d'utilisateurs en fonction des films vus par le passé par les membres du groupe mais aussi en fonction des films vus par d'autres groupes. Le système calcule la similarité entre groupes qui dépend de la similarité des membres, et de la confiance entre groupes fondée sur des critères tels que l'âge, le genre, etc. [Saaya *et al.*, 2011] intègrent la confiance en plus de la provenance dans un modèle de recherche collaborative sur le Web. Le modèle combine les méthodes du RÀPC et des réseaux pair-à-pair. Durant la recherche d'un nœud, les résultats précédemment sélectionnés par des nœuds dans le même réseau de confiance avec une requête similaire sont promus. Dans ce travail, un cas est un couple (requête, page web) et la provenance d'un cas est un indicateur de qualité du résultat. Ce système fonctionne comme un système de recommandations où les utilisateurs se voient proposer des solutions personnalisées. Néanmoins, les indicateurs de qualité portent seulement sur les cas de la base de cas qui ont déjà été retournés lors d'un épisode de recherche.

Les métaconnaissances sont aussi utilisées pour la maintenance de la base de cas. Pour maintenir les performances d'un système de RÀPC en stabilisant la taille de base de cas, [Smyth et Keane, 1995] ont proposé de modéliser les compétences des cas selon leur taux de recouvrement et leur richesse. Habituellement, les retours utilisateurs dans le système permettent de juger la qualité d'un nouveau cas et de réparer une adaptation qui a échoué [Dividino *et al.*, 2009]. En revanche, certains auteurs pensent que les retours utilisateurs sont insuffisants en raison d'observations de retours manquant, et en raison du fait que les métaconnaissances sont plus pertinentes pour améliorer les réponses du système. [Leake et Whitehead, 2007] montrent les limites des sys-

tèmes de RÀPC dans lesquels les retours permettent de guider la maintenance de la base de cas et impacter les futurs réponses du système. Par exemple, la réparation est propagée à travers les cas générés provenant d'un cas initial et la qualité d'un cas est calculée grâce à la longueur de la chaîne d'adaptation. En revanche, la provenance n'est pas intégrée dans le processus de raisonnement et les cas initiaux sont considérés comme étant de qualité équivalente.

[Richards, 2009] a étudié les supports de décision à partir d'une base de connaissances constituée par des experts mais aussi par des connaissances collectées à partir d'outils collaboratifs. Elle propose une approche d'ingénierie collaborative de connaissances mixant les méthodes de RÀPC et les méthodes de systèmes à base de règles pour permettre aux utilisateurs de gérer de façon collaborative la base de connaissances dans le temps. Cette approche permet aussi de représenter la fiabilité des connaissances de la base de connaissances.

D'après notre revue de la littérature, seule la confiance est utilisée dans le moteur des systèmes de RÀPC afin de personnaliser les réponses retournées par le système de RÀPC. La provenance, la qualité et la fiabilité sont seulement utilisées pour maintenir la base de cas et fournir des explications sur les réponses du système.

1.6.5 Les systèmes de RÀPC utilisant la notion de typicalité

La typicalité est une notion qui a déjà été utilisée dans les systèmes de RÀPC. [Weber-Lee *et al.*, 1996] utilisent la typicalité pour sélectionner le meilleur cas dans l'ensemble des cas les plus similaires remémorés. La mesure de la valeur la plus typique (*Most Typical Value*) introduite par [Friedman *et al.*, 1995] est utilisée pour déterminer le meilleur cas.

Tandis que [Weber-Lee *et al.*, 1996] utilisent une mesure de la typicalité dans le but de remémorer le meilleur cas, Protos [Bareiss *et al.*, 1990], un système d'apprentissage à partir d'exemples, utilise la typicalité des connaissances comme une partie des connaissances du domaine. Dans Protos, un exemple est un cas. Les connaissances du domaine et les cas sont représentés dans une seule structure : une structure de catégories. La structure de catégories est un réseau sémantique dans lequel les nœuds sont des propriétés ou des exemples qui sont les cas retenus, et les arcs sont des explications acquises par des experts lorsqu'une classification a échoué. Protos classifie un nouveau problème décrit par des propriétés en cherchant le cas le plus similaire en deux étapes majeures. La première étape consiste à remémorer la catégorie la plus similaire, la dernière, consiste à choisir le meilleur cas en fonction des arcs décrivant les prototypes et les différences entre cas.

Certains systèmes de RÀPC dans le domaine médical utilisent des cas prototypiques pour organiser la base de cas et guider l'étape de remémoration, par exemple, [Schmidt et Gierl, 1996] pour le diagnostic de syndromes dysmorphiques et [Bellazzi *et al.*, 1998] pour le diagnostic et le traitement du diabète. Dans les systèmes de RÀPC du domaine médical, les cas sont souvent représentés par une longue liste de symptômes. Par exemple, dans [Schmidt et Gierl, 1996], un cas représentant un patient atteint de syndromes dysmorphiques est décrit par 40 à 130 symptômes. Cependant, un diagnostic peut être représenté par un ensemble de symptômes typiques, par exemple, les symptômes les plus fréquents ou les plus fortement ressentis. Ainsi, ces systèmes permettent la construction de cas prototypiques décrits par des symptômes typiques. Ces cas prototypiques sont construits par des experts ou par des processus semi-automatiques sur la base de cas. La construction de cas prototypiques permet de réduire le nombre de symptômes décrits par rapport aux cas non prototypiques. L'étape de remémoration déclenchée par la description d'un nouveau patient se fait alors seulement sur les cas prototypiques, réduisant l'espace de recherche. Pour calculer la similarité entre un nouveau patient et un cas prototypique, [Schmidt et Gierl, 1996] utilisent deux mesures : la mesure de [Rosch et Mervis, 1975a] où la similarité dépend du

nombre de symptômes du nouveau patient correspondant aux symptômes d'un cas prototypique et la mesure de [Tversky, 1977] qui prend, en plus, en compte le nombre de symptômes du cas prototypique qui ne sont pas des symptômes du nouveau patient.

Chapitre 2

eTaaable : un système de raisonnement à partir de cas utilisant des connaissances provenant d'une communauté en ligne

Sommaire

Le domaine applicatif de cette thèse est le système de RàPC eTaaable du domaine culinaire qui utilise des connaissances provenant d'une communauté en ligne. Les origines de eTaaable proviennent du système de RàPC Taaable du domaine culinaire : eTaaable est une version de Taaable qui au lieu d'utiliser des connaissances provenant d'experts du domaine, utilise des connaissances provenant d'une communauté en ligne.

2.1 Le système Taaable

Le système Taaable est un système de RàPC du domaine culinaire qui recherche et adapte des recettes de cuisine pour satisfaire une requête utilisateur. Plusieurs versions de Taaable existent. La première version de Taaable a été créée en 2008 dans le but de participer au CCC [18] (*Computer Cooking Contest*). Le CCC est une compétition qui a lieu chaque année depuis 2008 et qui est organisée dans le cadre des congrès ICCBR (*International Conference on Case-Based Reasoning*). Depuis, Taaable participe au CCC chaque année et a déjà remporté plusieurs prix.

Le cadre de Taaable a été utilisé comme domaine applicatif de différentes thèses : la thèse d'Amélie Cordier [Cordier, 2008] ainsi que la thèse de Fadi Badra [Badra, 2009] pour l'acquisition de connaissances d'adaptation, la thèse de Julien Cojan [Cojan, 2011] pour l'adaptation des quantités des ingrédients dans les recettes et la thèse de Valmi Dufour-Lussier [Dufour-Lussier, 2014] pour l'adaptation des textes de préparation des recettes. Les apports de ces différentes thèses ont permis d'étendre Taaable au fil des années.

Taaable est composé d'une interface utilisateur, de quatre conteneurs de connaissances et d'un moteur. Les quatre conteneurs de connaissances de Taaable sont : la base de cas, les connaissances du domaine représentées dans une ontologie du domaine, les connaissances d'adaptation et les connaissances de similarité. Le moteur de Taaable a évolué en trois versions différentes.

Dans la première version du moteur de Taaable, les connaissances des conteneurs de connaissances étaient directement chargées dans la mémoire du moteur de Taaable à partir de différents fichiers encodés dans des formats différents [Badra *et al.*, 2008]. Cette version posait des problèmes de mise à jour et d'interopérabilité des connaissances encodées dans les différents fichiers.

La deuxième version du moteur de Taaable a permis d'améliorer le problème d'interopérabilité en utilisant un unique fichier RDF [19] (*Resource Description Framework*) exporté à partir d'un wiki sémantique nommé WikiTaaable [20] [Cordier *et al.*, 2009]. L'annexe B donne les détails du langage RDF. Dans Taaable, le fichier RDF exporté était chargé dans la mémoire du moteur de RàPC. Cependant, avec cette solution, le problème de mise à jour persistait. En effet, à chaque modification réalisée dans WikiTaaable, un nouvel export devait être effectué pour mettre à jour la mémoire du moteur de Taaable.

Pour remédier à ce problème, la troisième et dernière version du moteur de Taaable a été mise en place. Cette version correspond à une instance du moteur générique de RàPC nommé Tuuurbine [Gaillard *et al.*, 2014a]. Tuuurbine raisonne sur des triplets RDF stockés dans un *triplestore* interrogé par des requêtes SPARQL [21]. Tout comme RDF, le langage SPARQL est un standard du W3C permettant d'interroger des triplets RDF tandis qu'un *triplestore* est une base de données destinée à gérer des triplets RDF. Afin que Taaable puisse utiliser une instance de Tuuurbine comme moteur, WikiTaaable a été connecté à un *triplestore*. Ainsi, à chaque modification du wiki, les triplets RDF du *triplestore* sont mis à jour et le moteur de Taaable utilise les connaissances mises à jour. Les détails de Tuuurbine sont donnés en section 2.4.

18. `http://computercookingcontest.net/`
19. `http://www.w3.org/RDF`
20. `http://wikitaaable.loria.fr/`
21. `http://www.w3.org/2009/sparql`

2.2 Le système eTaaable

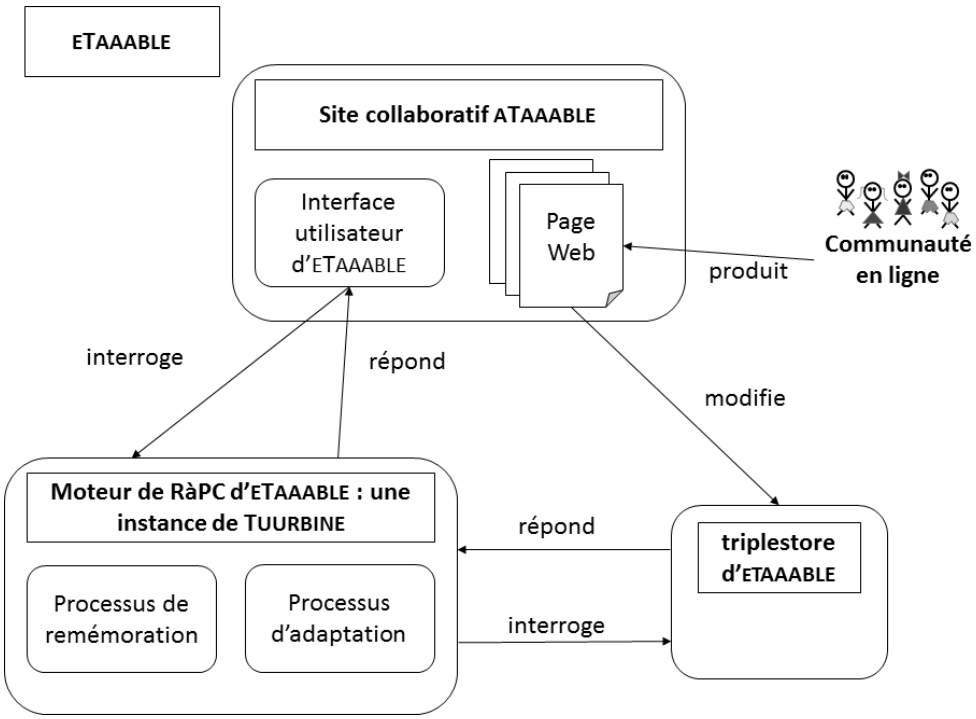


Figure 2.1 – Architecture d'eTaaable.

Le système de RàPC eTaaable est un système de RàPC du domaine culinaire qui recherche et adapte des recettes de cuisine comme Taaable. Cependant, eTaaable utilise des connaissances provenant d'une communauté en ligne. eTaaable utilise une interface utilisateur, une base de connaissances et un moteur qui, comme Taaable, est une instance de Tuuurbine. Comme pour Taaable, la base de connaissances de eTaaable notée BC est composée de quatre conteneurs de connaissances qui sont la base de cas, les connaissances du domaine représentées dans une ontologie du domaine, les connaissances d'adaptation et les connaissances de similarité. Les conteneurs de connaissances sont décrits dans les paragraphes suivants. En revanche, à la différence de Taaable, les conteneurs de connaissances sont produits par une communauté en ligne via le site Web collaboratif nommé aTaaable. Sur aTaaable, les utilisateurs éditent des pages Web, ce qui permet de générer, de façon cachée, des triplets RDF qui sont ensuite stockés dans un *triplestore*. La figure 2.2 présente l'architecture de eTaaable.

Nous présentons les UC de la base de connaissances de eTaaable grâce à la logique de descriptions $\mathcal{ALC}$ présentée en annexe A.

2.2.1 Les connaissances du domaine.

Les connaissances du domaine sont représentées dans une ontologie du domaine, notée `CD`, composée d'un ensemble de classes organisées selon la relation de subsomption notée $\sqsubseteq$ et classées dans plusieurs hiérarchies : aliment, type de plat, localisation et moment où un plat se déguste. Étant donné deux classes `A` et `B` de l'ontologie, `B` est subsumée par `A`, noté `B` $\sqsubseteq$ `A`, si l'ensemble des instances de `B` est inclus dans l'ensemble des instances de `A`. Par exemple,

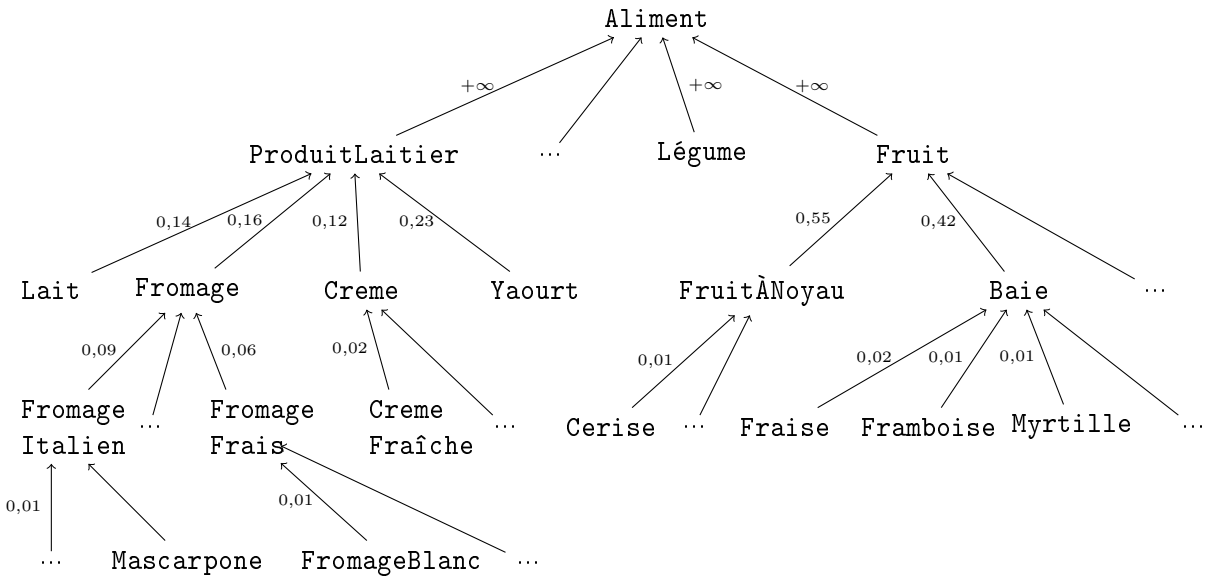


Figure 2.2 – Partie de la hiérarchie des aliments de l'ontologie du domaine utilisée par le système eTaaable. $\texttt{B} \xrightarrow{x} \texttt{A}$ signifie que $\texttt{B} \sqsubseteq \texttt{A}$ et $\texttt{coût}(\texttt{B},\texttt{A}) = x$.

$\texttt{Abricot} \sqsubseteq \texttt{FruitÀNoyau}$, signifie que tous les abricots sont des fruits à noyau. Dans CD, seules les UC produites par la communauté en ligne sont représentées. Les connaissances inférées de BC ne sont pas représentées. Ainsi, le diagramme de Hasse de chaque hiérarchie de l'ontologie du domaine est représenté grâce aux connaissances de CD.

Soit $(\mathcal{H}, \sqsubseteq)$, une hiérarchie de l'ontologie du domaine utilisée par eTaaable et soit $G = (\mathcal{H},r)$ le diagramme de Hasse de $(\mathcal{H}, \sqsubseteq)$. Si $\texttt{A} \sqsubseteq \texttt{B} \in \texttt{CD}$ et que $\texttt{A} \sqsubseteq \texttt{C} \notin \texttt{CD}$ et $\texttt{C} \sqsubseteq \texttt{B} \notin \texttt{CD}$ pour $\texttt{C} \neq \texttt{A}$ et $\texttt{C} \neq \texttt{B}$, alors $\texttt{A} \sqsubseteq \texttt{B}$ est une relation représentée dans le diagramme de Hasse. Les diagrammes de Hasse sont introduits dans l'annexe C.

Par exemple, $(\mathcal{H}_{Al}, \sqsubseteq)$ est la hiérarchie des aliments de l'ontologie du domaine utilisée par eTaaable et $G = (\mathcal{H}_{Al},r)$ est le diagramme de Hasse de $(\mathcal{H}_{Al}, \sqsubseteq)$. La figure 2.2 montre une partie de $G = (\mathcal{H}_{Al},r)$ de la hiérarchie des aliments où une flèche signifie « est subsumée par ». De plus, les valeurs sur les flèches qui ont été ajoutées correspondent aux connaissances de remémoration qui seront détaillées dans la section 2.4. Comme, $\texttt{Fraise} \sqsubseteq \texttt{Baie}$ est représentée dans $G = (\mathcal{H}_{Al},r)$, alors $\texttt{Fraise} \sqsubseteq \texttt{Baie} \in \texttt{CD}$.

2.2.2 La base de cas

La base de cas est un ensemble de recettes. Une classe de recettes est représentée par une conjonction de classes de la forme suivante :

- $\exists\texttt{typePlat.Tp}$, où Tp est une classe atomique de la hiérarchie des types de plat.
- $\exists\texttt{ing.Al}$ où Al est une classe atomique de la hiérarchie des aliments.
- $\exists\texttt{origine.Lo}$ où Lo est une classe atomique de la hiérarchie des localisations.
- $\exists\texttt{peutÊtreMangéComme.Mr}$ où Mr est une classe atomique de la hiérarchie des moments de repas.

Une classe des recettes est décrite par au moins un type de plat (par exemple, `Tarte`) et au moins un ingrédient (par exemple, `Fraise`), et peut être décrite par ses origines (par exemple, `France`) et le moment du repas où le plat produit se déguste (par exemple, `Dessert`).

Par exemple,

$$\mathtt{R}_1 = \mathtt{Recette} \sqcap \exists \mathtt{typePlat.Tarte} \sqcap \exists \mathtt{ing.PâteSablée} \sqcap \exists \mathtt{ing.Framboise} \sqcap \exists \mathtt{ing.Mascarpone}$$

représente la classe de recettes qui sont des tartes ayant pour ingrédient de la pâte sablée, des framboises et du mascarpone.

Une recette est une instance d'une sous-classe de `Recette`. Par exemple, si la recette « Tarte framboise mascarpone » est une recette de tarte ayant pour ingrédient de la pâte sablée, des framboises et du mascarpone, alors :

$$\mathtt{R}_1(\text{TARTE\_FRAMBOISE\_MASCARPONE})$$

On dit que TARTE_FRAMBOISE_MASCARPONE est indexée par `Tarte`, `PâteSablée`, `Framboise` et `Mascarpone`, qui sont des classes de l'ontologie du domaine. On note l'index de TARTE_FRAMBOISE_MASCARPONE comme suit :

$$idx(\text{TARTE\_FRAMBOISE\_MASCARPONE}) = \mathtt{Recette} \sqcap \exists \mathtt{typePlat.Tarte} \sqcap \exists \mathtt{ing.PâteSablée} \sqcap \exists \mathtt{ing.Framboise} \sqcap \exists \mathtt{ing.Mascarpone}$$

2.2.3 Les règles de substitution

Dans Taaable, certaines connaissances d'adaptation correspondent à des règles de substitution. Une règle de substitution σ est définie par un contexte d'utilisation qui peut être une recette ou un type de plat, par des ingrédients substitués et par des ingrédients substituants. L'ensemble des ingrédients substitués ou l'ensemble des ingrédients substituants peut être vide (mais pas les deux à la fois) ; ce qui signifie qu'une règle de substitution peut dire que des ingrédients peuvent être ajoutés ou supprimés et pas seulement remplacés. Une règle de substitution σ est définie syntaxiquement par :

$$\sigma = \mathtt{Contexte} \sqcap \mathtt{IngrédientsSubstitués} \rightsquigarrow \mathtt{Contexte} \sqcap \mathtt{IngrédientsSubstituants}$$

où $\rightsquigarrow$ signifie « est substitué par ». `Contexte` est une sous-classe de `Recette` ou une classe de la forme $\exists$`typePlat.Tp`, où `Tp` est une classe atomique de la hiérarchie des types de plat. `IngrédientsSubstitués` et `IngrédientsSubstituants` sont des conjonctions de classes de la forme $\exists$`ing.Al` où `Al` est une classe atomique de la hiérarchie des aliments.

Par exemple,

$$\sigma_{ex} = \exists \mathtt{typePlat.Tarte} \sqcap \exists \mathtt{ing.Myrtille} \sqcap \exists \mathtt{ing.Mascarpone} \rightsquigarrow \exists \mathtt{typePlat.Tarte} \sqcap \exists \mathtt{ing.Fraise} \sqcap \exists \mathtt{ing.FromageBlanc}$$

est la règle de substitution disant que les myrtilles et le mascarpone peuvent être remplacés par des fraises et du fromage blanc dans les tartes.

2.2.4 Les requêtes

Dans Taaable, le cas cible est une requête composée d'un ensemble de contraintes utilisateur. Formellement, une requête, notée $\mathtt{Q}$, est une recette qui en fonction des contraintes est composée d'une conjonction de classes de la même forme que pour les recettes.

Par exemple,

$$\mathtt{Q}_{ex} = \mathtt{Recette} \sqcap \exists \mathtt{typePlat.Tarte} \sqcap \exists \mathtt{ing.Fraise} \sqcap \exists \mathtt{ing.FromageBlanc} \sqcap \neg \exists \mathtt{ing.Vanille} \tag{2.1}$$

est la requête d'une recette de tarte avec des fraises, du fromage blanc mais pas de vanille.

Remarque : Lorsqu'une requête fait intervenir une négation, un problème se pose. En toute rigueur, avec la requête $\mathtt{Q}_{ex}$, nous avons :

$$\not\models_{BC} \mathtt{Q}_{ex}(\text{TARTE\_FRAMBOISE\_MASCARPONE})$$

En effet, rien ne dit qu'il n'y a pas de vanille dans la recette « Tarte framboise mascarpone » car $\neg\exists\mathtt{ing.Vanille}$ n'apparaît pas dans la définition TARTE_FRAMBOISE_MASCARPONE. Pour contourner ce problème, on se place sous l'hypothèse du mondes clos. Nous faisons l'hypothèse que tout ce que nous ignorons sur les recettes est considéré comme faux, c'est-à-dire que toutes les connaissances qui ne sont pas dans la base de connaissances sont rejetées. Par conséquent, $\not\models_{BC} \exists\mathtt{ing.Vanille}$(TARTE_FRAMBOISE_MASCARPONE) entraîne, selon l'hypothèse du monde clos, $\neg\exists\mathtt{ing.Vanille}$(TARTE_FRAMBOISE_MASCARPONE). D'une façon générale, on utilise la règle d'inférence :

$$\frac{\not\models_{BC} f}{\neg f}$$

où f est une formule de la forme $\exists\mathtt{p.A}$ (où $\mathtt{p}$ est une propriété et $\mathtt{A}$ est une classe atomique [22]).

2.2.5 Simplification d'écriture

Pour alléger les notations des recettes, des requêtes et des règles de substitution, nous procédons comme suit :

(a) Soit la fonction $\mathtt{p}(\mathtt{A})$ qui a une classe atomique $\mathtt{A}$ associe une propriété $\mathtt{p}$. Comme chaque classe atomique appartient à une hiérarchie et une seule, $\mathtt{p}(\mathtt{A})$ est associé à cette hiérarchie :
- la propriété `ing` est associée à la hiérarchie des aliments ;
- la propriété `typePlat` est associée à la hiérarchie des types de plats ;
- la propriété `origine` est associée à la hiérarchie des localisations ;
- la propriété `peutÊtreMangéComme` est associée à la hiérarchie des moments où se mange un plat.

Par exemple, si $\mathtt{A}$ est un aliment, il sera associé à la hiérarchie des aliments et $\mathtt{p}(\mathtt{A}) = \mathtt{ing}$. Pour `Pomme`, $\mathtt{p}(\mathtt{Pomme}) = \mathtt{ing}$ car `Pomme` appartient à la hiérarchie des aliments.

La fonction $\mathtt{p}$ permet de simplifier l'écriture des recettes, des requêtes et des règles de substitution.

22. On ne veut pas faire l'hypothèse du monde clos sur n'importe quel type de formule f. Par exemple, si $\models_{BC} \mathtt{A}(a)$ et $\not\models_{BC} \mathtt{A}(a)$, l'hypothèse du monde clos entraînerait une contradiction : $(\mathtt{A} \sqcap \neq \mathtt{A})(a)$.

(b) On substitue $\exists \mathtt{p}(\mathtt{A}_i).\mathtt{A}_i$ par $\mathtt{A}_i$ dans les définitions des classes de recettes, des requêtes et des règles de substitution. Par exemple, pour la requête $\mathtt{Q}_{ex}$:

$$\mathtt{Q}_{ex} = \mathtt{Recette} \sqcap \exists \mathtt{typePlat.Tarte} \sqcap \exists \mathtt{ing.Fraise} \sqcap \exists \mathtt{ing.FromageBlanc} \sqcap \neg \exists \mathtt{ing.Vanille}$$

$$\mathtt{Q}_{ex} = \mathtt{Recette} \sqcap \exists \mathtt{p}(\mathtt{Tarte}).\mathtt{Tarte} \sqcap \exists \mathtt{p}(\mathtt{Fraise}).\mathtt{Fraise} \sqcap \exists \mathtt{p}(\mathtt{FromageBlanc}).\mathtt{FromageBlanc} \sqcap \neg \exists \mathtt{p}(\mathtt{Vanille})\mathtt{Vanille}$$

En substituant $\exists \mathtt{p}(\mathtt{A}_i).\mathtt{A}_i$ par $\mathtt{A}_i$:

$$\mathtt{Q}_{ex} = \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{Fraise} \sqcap \mathtt{FromageBlanc} \sqcap \neg \mathtt{Vanille} \tag{2.2}$$

De la même façon, la classe $\mathtt{R}_1$ qui est une sous-classe de `Recette` :

$$\mathtt{R}_1 = \mathtt{Recette} \sqcap \exists \mathtt{typePlat.Tarte} \sqcap \exists \mathtt{ing.PâteSablée} \sqcap \exists \mathtt{ing.Framboise} \sqcap \exists \mathtt{ing.Mascarpone}$$

et l'index de TARTE_FRAMBOISE_MASCARPONE :

$$idx(\text{TARTE\_FRAMBOISE\_MASCARPONE}) = \mathtt{Recette} \sqcap \exists \mathtt{typePlat.Tarte} \sqcap \exists \mathtt{ing.PâteSablée} \sqcap \exists \mathtt{ing.Framboise} \sqcap \exists \mathtt{ing.Mascarpone}$$

sont simplifiés par :

$$\mathtt{R}_1 = \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{PâteSablée} \sqcap \mathtt{Framboise} \sqcap \mathtt{Mascarpone} \tag{2.3}$$

et :

$$idx(\text{TARTE\_FRAMBOISE\_MASCARPONE}) = \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{PâteSablée} \sqcap \mathtt{Framboise} \sqcap \mathtt{Mascarpone} \tag{2.4}$$

La règle de substitution :

$$\sigma_{ex} = \exists \mathtt{typePlat.Tarte} \sqcap \exists \mathtt{ing.CrèmeFraîche} \rightsquigarrow \exists \mathtt{typePlat.Tarte} \sqcap \exists \mathtt{ing.FromageBlanc}$$

est simplifiée par :

$$\sigma_{ex} = \mathtt{Tarte} \sqcap \mathtt{CrèmeFraîche} \rightsquigarrow \mathtt{Tarte} \sqcap \mathtt{FromageBlanc} \tag{2.5}$$

2.3 La base de connaissances de eTaaable et le site Web collaboratif aTaaable

Dans eTaaable, les UC de la base de connaissances sont construites à partir d'informations éditées sur un site collaboratif, nommé aTaaable (`http://ataaable.loria.fr`). Le site aTaaable réunit une communauté en ligne qui édite des informations du domaine culinaire. À chaque fois qu'un utilisateur édite une page sur aTaaable, des UC sont créées permettant de stocker des triplets RDF dans le *triplestore* d'eTaaable. Chaque paragraphe de cette section présente les différentes pages pouvant être éditées sur aTaaable, et pour chaque page les triplets RDF résultant de cette édition.

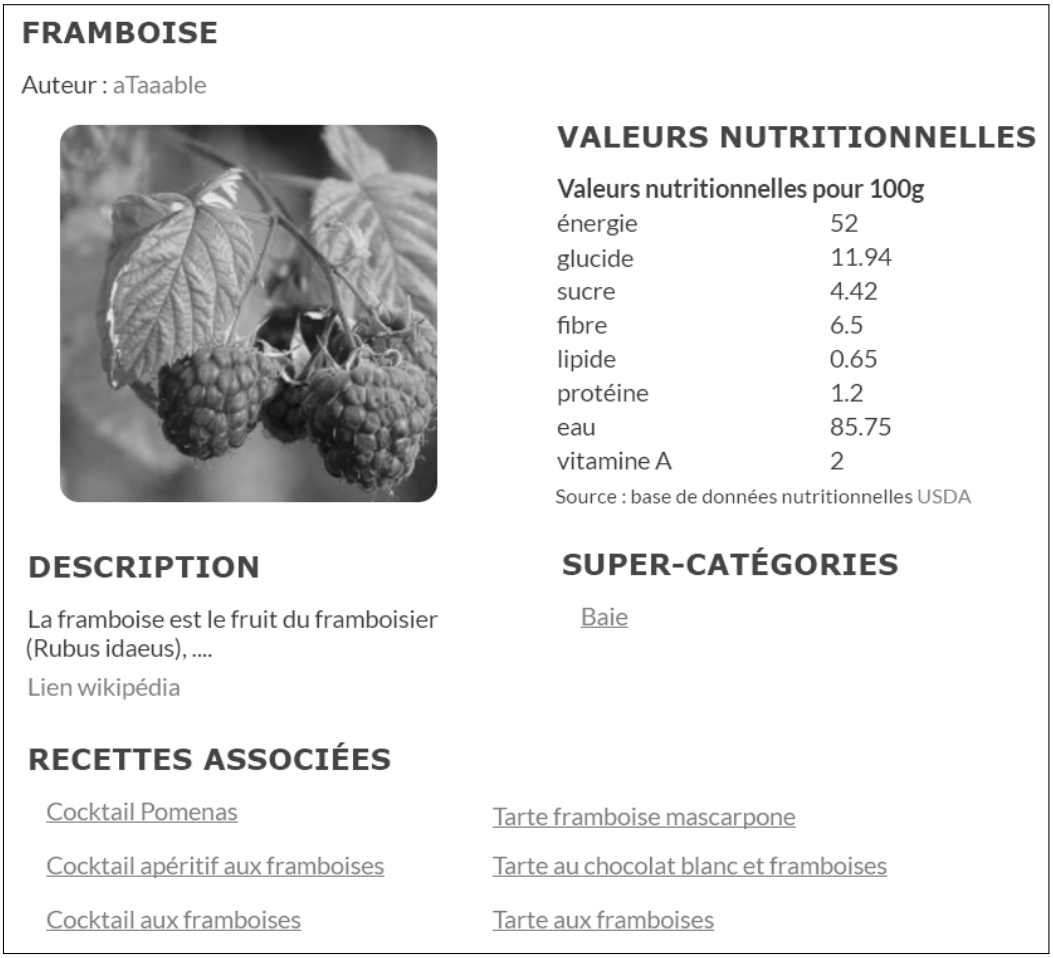


Figure 2.3 – Page de l'aliment `Framboise` sur aTaaable.

Édition des pages. Sur aTaaable, seuls les utilisateurs inscrits peuvent éditer. L'édition d'une page se fait par le biais de formulaires. À chaque fois qu'une page est éditée et validée sur aTaaable, certaines des informations de la page permettent de créer des UC qui sont stockées sous forme de triplets RDF dans le *triplestore* d'eTaaable.

2.3.1 Édition des pages représentant les classes de l'ontologie.

Sur aTaaable, les utilisateurs peuvent éditer une page représentant un aliment, un type de plat, une origine ou un moment où un plat se mange, en indiquant diverses informations. Ces informations correspondent à une description textuelle, un lien vers la page Wikipédia qui décrit la page, une ou plusieurs photo(s). Chacune des informations est remplie par l'utilisateur grâce à différents formulaires. S'il s'agit d'une page d'aliment les données nutritionnelles de l'aliment sont affichées aux utilisateurs. Ces données nutritionnelles proviennent d'une base de données libre de droits de *the United States Department of Agriculture* (USDA, `http://www.ars.usda.gov/`). Certains travaux de Taaable utilisent ces données nutritionnelles, par exemple [Cojan, 2011] pour l'adaptation des quantités des ingrédients dans les recettes. Dans cette thèse, nous n'utilisons pas ces données dans le raisonnement ; elles sont seulement données à titre indicatif aux utilisateurs de la communauté en ligne. La figure 2.3 montre la page de l'aliment `Framboise`

contenant les informations remplies par l'auteur de la page. De plus, l'utilisateur doit indiquer à quelle catégorie appartient l'élément représenté par la page. Dans la figure 2.3, l'utilisateur a fourni l'information que la framboise était une catégorie de baie.

À chaque fois qu'un utilisateur créée une page représentant un aliment, un type de plat, une origine ou un moment où un plat se mange, une UC est créée dans le conteneur des connaissances du domaine. Dans l'exemple de la figure 2.3, l'UC suivante est créée : $\texttt{Framboise} \sqsubseteq \texttt{Baie} \in \texttt{CD}$ signifiant que la classe `Framboise` est subsumée par la classe `Baie`. Cette UC est stockée sous forme d'un triplet RDF dans le *triplestore* de eTaaable :

$$\langle \texttt{Framboise rdfs:subClassOf Baie} \rangle$$

où `rdfs:subClassOf` est la propriété signifiant « est sous classe de ».

Les pages de recettes et de règles de substitutions sur aTaaable sont décrites à l'aide des pages représentant un aliment, un type de plat, une origine ou un moment où un plat se mange.

2.3.2 Édition d'une recette

Sur aTaaable, les utilisateurs peuvent éditer une page représentant une recette décrite par un titre, une préparation, les ingrédients que la recette contient, le type de plat produit par la recette et d'autres informations additionnelles facultatives. La préparation de la recette correspond à une description textuelle. Les ingrédients contenus dans la recette sont représentées par des lignes d'ingrédients où chaque ligne est une zone textuelle contenant le nom d'un ingrédient utilisé par la recette et éventuellement la quantité et l'unité de cet ingrédient dans la recette. Lorsqu'un utilisateur crée une recette, il doit entrer le texte de chacune des lignes d'ingrédients et lorsqu'il valide la recette, un processus d'annotation est déclenché. Ce processus d'annotation permet d'extraire pour chaque ligne, l'ingrédient qui correspond à une page d'aliment, si cette page existe, ainsi que la quantité de l'ingrédient et son unité, si ces informations existent. L'utilisateur doit préciser le type de plat de la recette parmi les noms de pages correspondant à un type de plat. Les informations additionnelles sont le temps de préparation, le temps de cuisson, le nombre de personnes, le coût de la recette, le niveau de difficulté de la recette, l'origine de la recette et le moment du repas durant lequel le plat produit par la recette se déguste. Chacun des ingrédients de la recette, l'origine de la recette, le type de plat de la recette et le moment du repas durant lequel la recette se déguste pointent vers les pages de aTaaable correspondantes. La figure 2.4 montre la page de la recette nommée « Tarte framboise mascarpone ». Cette recette est une tarte, a pour origine France, peut-être mangée comme dessert et contient de la pâte sablée, des framboises, etc.

À chaque fois qu'un utilisateur créée une page représentant une recette, deux UC sont créées. Dans l'exemple de la recette de la figure 2.4, les UC suivantes sont créées :

$$\begin{aligned} \texttt{R}_1 = \texttt{Recette} \sqcap \texttt{Tarte} \sqcap \texttt{PâteSablée} \sqcap \texttt{Framboise} \sqcap \texttt{Mascarpone} \\ \sqcap \texttt{Sucre} \sqcap \texttt{SucreVanillé} \sqcap \texttt{Maïzena} \sqcap \texttt{Œuf} \end{aligned}$$

et

$$\texttt{R}_1(\text{TARTE\_FRAMBOISE\_MASCARPONE})$$

Ces UC sont stockées sous forme de triplets RDF dans le triplestore de eTaaable dont :

$$\langle \texttt{TarteFramboiseMascarpone rdf:type Recette} \rangle$$
$$\langle \texttt{TarteFramboiseMascarpone typePlat Tarte} \rangle,$$
$$\text{et } \langle \texttt{TarteFramboiseMascarpone ingredient Framboise} \rangle$$

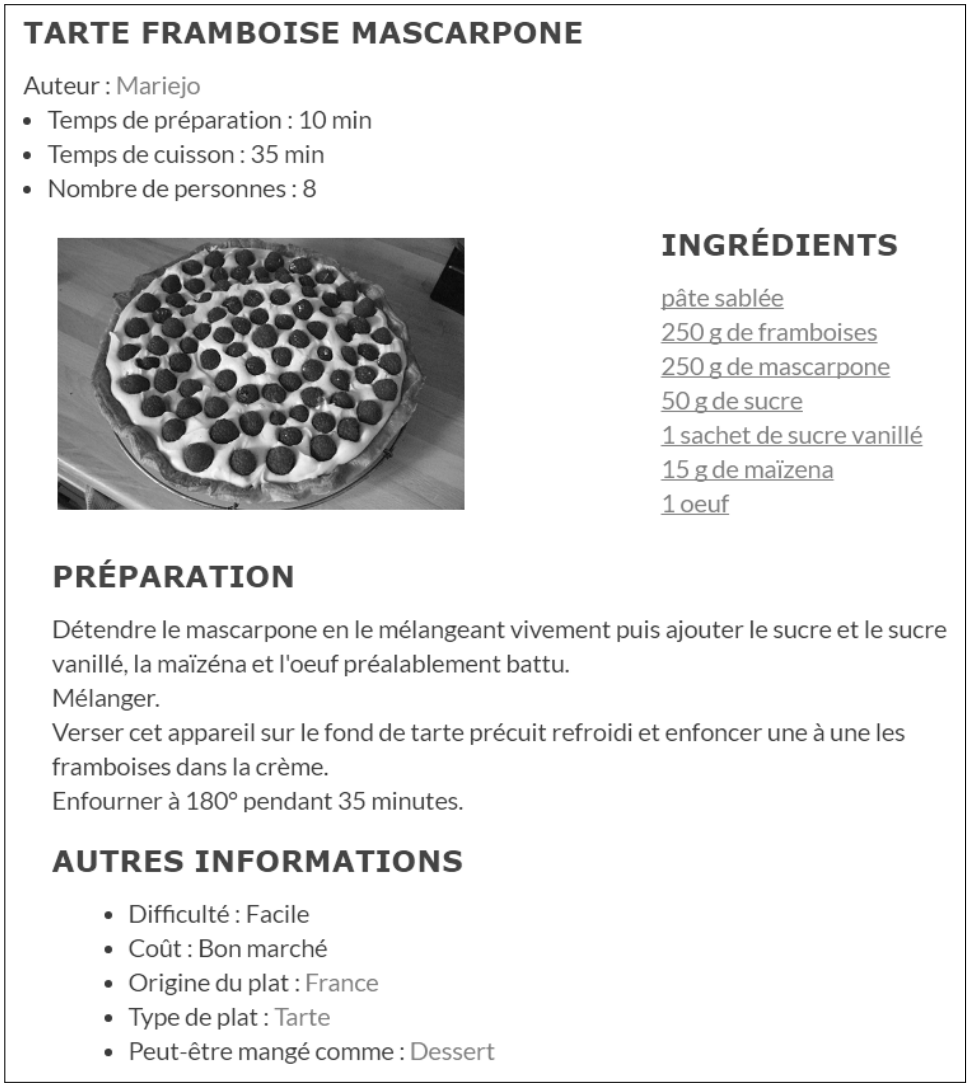


Figure 2.4 – Page de la recette « Tarte framboise mascarpone » sur aTaaable.

où `rdf:type` est la propriété signifiant « est instance de », `typePlat` est la propriété signifiant « a pour type » et où `ingredient` est la propriété signifiant « a pour ingrédient ».

2.3.3 Édition d'une règle de substitution

Sur aTaaable, une règle de substitution est décrite dans une page divisée en trois parties :
- les ingrédients substitués : par exemple, la myrtille et le mascarpone ;
- les ingrédients substituants : par exemple, la fraise et le fromage blanc ;
- le contexte, qui peut être une recette spécifique, comme « Tarte framboise mascarpone », ou un type de plat, comme Tarte, pour lequel la règle s'applique.

Lorsque l'utilisateur édite la page d'une règle de substitution, chaque ingrédient substitué et chaque ingrédient substituant doit être une page existante représentant un aliment. De la même façon, le contexte doit être une page existante représentant une recette ou un type de plat. La figure 2.5 montre la page de la règle de substitution intitulée « Substitution Tarte 1 », qui dit

Figure 2.5 – Page de la règle de substitution, sur aTaaable, qui dit que les myrtilles et le mascarpone peuvent être remplacés par des fraises et du fromage blanc dans les tartes.

que les myrtilles et le mascarpone peuvent être remplacés par des fraises et du fromage blanc dans les tartes.

Dans l'exemple de la figure 2.5, l'UC suivante est créée :

$$\sigma_{SubstitutionTarte1} = \texttt{Tarte} \sqcap \texttt{Myrtille} \sqcap \texttt{Mascarpone} \rightsquigarrow \texttt{Tarte} \sqcap \texttt{Fraise} \sqcap \texttt{FromageBlanc}$$

Cette UC est stockée dans le triplestore de eTaaable sous les formes des triplets RDF suivants :

$$\begin{aligned} &\langle \texttt{SubstitutionTarte1 contexte Tarte} \rangle \\ &\langle \texttt{SubstitutionTarte1 ingredientSubstitues Myrtille} \rangle, \\ &\langle \texttt{SubstitutionTarte1 ingredientSubstitues Mascarpone} \rangle, \\ &\langle \texttt{SubstitutionTarte1 ingredientSubstituants Fraise} \rangle, \\ \text{et } &\langle \texttt{SubstitutionTarte1 ingredientSubstituants FromageBlanc} \rangle \end{aligned}$$

où `contexte` est la propriété signifiant « a pour contexte », où `ingredientSubstitues` est la propriété signifiant « a pour ingrédient substitué » et où `ingredientSubstituants` est la propriété signifiant « a pour ingrédient substituant ».

2.3.4 Les utilisateurs de la communauté en ligne.

Chaque utilisateur de la communauté en ligne est décrit sur une page divisée en deux parties :

- Le profil de l'utilisateur décrit les informations personnelles de l'utilisateur qui sont son prénom, son sexe, ses origines, son pays de résidence, et ses habitudes culinaires, c'est-à-dire si l'utilisateur cuisine pour sa famille, pour ses amis, pour son travail, pour lui-même ou jamais. De plus, un profil utilisateur peut contenir une photo que l'utilisateur a téléchargée. La figure 2.6 montre le profil de l'utilisateur Emmanuel.
- Le tableau de bord de l'utilisateur liste les pages qu'il a créées sur aTaaable. La figure 2.7 montre le tableau de bord de l'utilisateur Emmanuel.

2.3.5 Interface utilisateur d'eTaaable

Le site aTaaable contient aussi l'interface utilisateur d'eTaaable permettant d'interroger le moteur d'eTaaable par une requête utilisateur et d'afficher les réponses retournées par le moteur. L'utilisateur liste ses contraintes séparées par des virgules. Il y a deux types de contraintes.

Figure 2.6 – Profil de l'utilisateur Emmanuel.

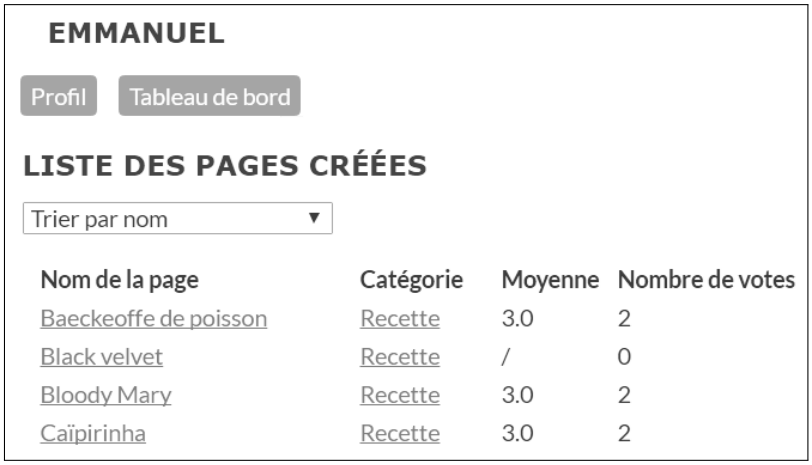


Figure 2.7 – Tableau de bord de l'utilisateur Emmanuel.

Si l'utilisateur désire que la recette contienne un ingrédient donné, soit d'un type de plat donné, soit d'une origine ou que le plat produit se mange à un moment donné du repas alors l'utilisateur doit écrire le nom de l'ingrédient, du type de plat, etc. Si l'utilisateur ne désire pas que la recette contienne un ingrédient donné, soit d'un type de plat donné, etc., alors l'utilisateur doit écrire le signe « − » suivi du nom de l'ingrédient, du type de plat, etc. Dans l'exemple de la figure 2.8, l'utilisateur veut une recette de tarte contenant des fraises, du fromage blanc et pas de vanille. Lorsque l'utilisateur valide sa requête, le moteur de eTaaable, qui est une instance de Tuuurbine, est déclenché et retourne ensuite les réponses calculées. Les 5 premières réponses sont affichées à l'utilisateur. Par exemple, la dernière réponse correspond à la recette « Tarte framboise mascarpone » qui est adaptée en remplaçant les framboises et le mascarpone par des fraises et du fromage blanc.

Figure 2.8 – Interface utilisateur de eTaaable.

2.4 Le moteur générique de RÀPC Tuuurbine

Cette section présente Tuuurbine, un moteur générique de RÀPC qui est le moteur utilisé par eTaaable. Tuuurbine est composé de deux processus de raisonnement : le processus de remémoration et le processus d'adaptation. Le fonctionnement de Tuuurbine est détaillé dans [Gaillard *et al.*, 2014a].

2.4.1 Le processus de remémoration

Le processus de remémoration consiste à trouver les cas sources les plus similaires au problème cible. Si une correspondance exacte existe entre un cas source de la base de cas et le problème cible, alors la solution sera proposée comme solution du problème cible. Ainsi, si $\mathtt{Q}$, la classe représentant la requête, subsume la classe $\mathtt{R}$ représentant une recette, alors les instances de $\mathtt{R}$ seront proposées comme solution à la requête $\mathtt{Q}$. Si aucune correspondance exacte n'existe, les contraintes de la requête sont relâchées en utilisant une fonction de généralisation notée Γ. La fonction Γ est composée d'étapes de généralisation qui transforment la requête $\mathtt{Q}$, avec un coût minimal, jusqu'à ce qu'au moins un cas source soit remémoré.

Une étape de généralisation est notée $\gamma = \mathtt{B} \rightsquigarrow \mathtt{A}$, où $\mathtt{A}$ et $\mathtt{B}$ sont des classes telles que $\models_{BC} \mathtt{A} \sqsubseteq \mathtt{B}$. Chaque étape de généralisation est associée à un coût noté $\mathtt{coût}(\mathtt{B} \rightsquigarrow \mathtt{A})$. La généralisation Γ de $\mathtt{Q}$ est une composition d'étapes de généralisations, $\gamma_1, \gamma_2, \ldots \gamma_n$: $\Gamma = \gamma_n \circ \ldots \circ \gamma_2 \circ \gamma_1$, avec $\mathtt{coût}(\Gamma) = \sum_{i=1}^{n} \mathtt{coût}(\gamma_i)$. Les détails du calcul du coût de généralisation sont détaillés dans [Cordier *et al.*, 2014].

Les fonctions de généralisation Γ_i sont construites jusqu'à trouver le coût minimal tel qu'au moins un cas source satisfasse $\Gamma(\mathtt{Q})$.

Soit `CD`, les connaissances de l'ontologie du domaine dont une partie est présentée dans la figure 2.2 où les scores sur les flèches entre une classe `B` et une classe `A` représentent le coût de généralisation de `B` en `A`. Soit la requête $\mathtt{Q}_{ex}$ de l'équation (2.2) que nous rappelons :

$$\mathtt{Q}_{ex} = \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{Fraise} \sqcap \mathtt{FromageBlanc} \sqcap \neg\mathtt{Vanille}$$

Soit la base de cas contenant une seule recette : la recette TARTE_FRAMBOISE_MASCARPONE dont l'index est représenté dans l'équation (2.4) que nous rappelons :

$$idx(\text{TARTE\_FRAMBOISE\_MASCARPONE}) = \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{PâteSablée} \sqcap \mathtt{Framboise} \sqcap \mathtt{Mascarpone}$$

Cette recette est une instance de la classe de recettes $\mathtt{R}_1$ représentée dans l'équation (2.3) que nous rappelons :

$$\mathtt{R}_1 = \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{PâteSablée} \sqcap \mathtt{Framboise} \sqcap \mathtt{Mascarpone}$$

La première fonction de généralisation permettant de remémorer une recette de la base de cas est

$$\Gamma = \mathtt{Fraise} \rightsquigarrow \mathtt{Baie} \circ \mathtt{FromageBlanc} \rightsquigarrow \mathtt{Fromage}$$

signifiant que `Fraise` est généralisée en `Baie` et `FromageBlanc` est généralisée en `Fromage`. Le coût de généralisation est $0{,}01 + (0{,}01 + 0{,}06) = 0{,}08$ car :

$$\begin{aligned} \mathtt{coût}(\mathtt{Fraise} \rightsquigarrow \mathtt{Baie}) &= 0{,}01, \\ \mathtt{coût}(\mathtt{FromageBlanc} \rightsquigarrow \mathtt{FromageFrais}) &= 0{,}01 \text{ et} \\ \mathtt{coût}(\mathtt{FromageFrais} \rightsquigarrow \mathtt{Fromage}) &= 0{,}06, \end{aligned}$$

qui signifie que le coût de généralisation de `Fraise` en `Baie` est de 0,01, le coût de généralisation de `FromageBlanc` en `FromageFrais` est de 0,01 et le coût de généralisation de `FromageFrais` en `Fromage` est de 0,06. La requête généralisée correspond alors à :

$$\Gamma(\mathtt{Q}_{ex}) = \ \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{Baie} \sqcap \mathtt{Fromage}.$$

Selon la figure 2.2, $\mathtt{Framboise} \sqsubseteq \mathtt{Baie}$ et $\mathtt{Mascarpone} \sqsubseteq \mathtt{Fromage}$. Ainsi $\mathtt{R}_1 \sqsubseteq \Gamma(\mathtt{Q}_{ex})$, et donc l'instance de $\mathtt{R}_1$, TARTE_FRAMBOISE_MASCARPONE est remémorée. Lorsque Tuuurbine a remémoré les cas sources les plus similaires, d'autres cas sources moins similaires peuvent aussi être retrouvés en continuant le processus de généralisation qui cherchera les prochaines généralisations, dans l'ordre des coûts croissants.

2.4.2 Le processus d'adaptation

Tuuurbine implémente deux processus d'adaptation. Le premier, nommé processus de généralisation-spécialisation, consiste à spécialiser la requête généralisée produite par la phase de remémoration. Selon $\Gamma(\mathtt{Q}_{ex})$, $\mathtt{Q}_{ex}$, $idx(\text{TARTE\_FRAMBOISE\_MASCARPONE})$, et `CD`, `Myrtille` est remplacée par `Fraise` et `Mascarpone` est remplacé par `FromageBlanc` dans l'index de TARTE_FRAMBOISE_MASCARPONE car `Baie` de $\Gamma(\mathtt{Q}_{ex})$ subsume les deux classes `Fraise` et

Myrtille et Fromage de $I(\mathtt{Q}_{ex})$ subsume les deux classes FromageBlanc et Mascarpone. Ainsi, par spécialisation de la requête généralisée, une réponse $\mathtt{Rép}_1$ pour $\mathtt{Q}_{ex}$ est :

$$\mathtt{Rép}_1 = \mathtt{Recette} \sqcap \mathtt{Plat} \sqcap \mathtt{PâteSablée} \sqcap \mathtt{Fraise} \sqcap \mathtt{FromageBlanc}$$

Un coût d'adaptation est associé à une adaptation issue du processus de généralisation-spécialisation. Ce coût est égal au coût de généralisation.

Dans l'exemple, comme $\mathtt{coût}(\mathtt{Fraise} \rightsquigarrow \mathtt{Baie} \circ \mathtt{FromageBlanc} \rightsquigarrow \mathtt{Fromage}) = 0{,}08$ alors le coût d'adaptation de TARTE_FRAMBOISE_MASCARPONE est de 0,08.

Le deuxième processus d'adaptation consiste à utiliser des règles de substitutions. Si la partie gauche d'une règle d'une substitution σ (contexte et ingrédients substitués) subsume la classe de la recette remémorée, alors la règle de substitutions est appliquée ; les ingrédients de la recette correspondant aux ingrédients substitués de la règle de substitution sont remplacés par les ingrédients substituants de la règle de substitution. Soit TARTE_MYRTILLE_MASCARPONE une recette ajoutée à la base de cas dont l'index est défini par :

$$idx(\text{TARTE\_MYRTILLE\_MASCARPONE}) = \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{PâteBrisée} \sqcap \mathtt{Myrtille} \sqcap \mathtt{Mascarpone} \sqcap \mathtt{Sucre} \sqcap \mathtt{Œuf}$$

et qui est une instance de la classe de recettes $\mathtt{R}_2$ définie par :

$$\mathtt{R}_2 = \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{PâteBrisée} \sqcap \mathtt{Myrtille} \sqcap \mathtt{Mascarpone} \sqcap \mathtt{Sucre} \sqcap \mathtt{Œuf}$$

Soit σ_{ex} défini par l'équation (2.5) que nous rappelons :

$$\sigma_{ex} \equiv \mathtt{Tarte} \sqcap \mathtt{Myrtille} \sqcap \mathtt{Mascarpone} \rightsquigarrow \mathtt{Tarte} \sqcap \mathtt{Fraise} \sqcap \mathtt{FromageBlanc}$$

Si TARTE_MYRTILLE_MASCARPONE est une recette remémorée par $\mathtt{Q}_{ex}$, σ_{ex} peut être appliquée à TARTE_MYRTILLE_MASCARPONE car :

$$\mathtt{R}_2 \sqsubseteq \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{Myrtille} \sqcap \mathtt{Mascarpone}$$

Une réponse $\mathtt{Rép}_2$ pour $\mathtt{Q}_{ex}$ est alors :

$$\mathtt{Rép}_2 = \mathtt{Recette} \sqcap \mathtt{Plat} \sqcap \mathtt{PâteBrisée} \sqcap \mathtt{Fraise} \sqcap \mathtt{FromageBlanc} \sqcap \mathtt{Sucre} \sqcap \mathtt{Œuf}$$

Un coût d'adaptation est associé à une adaptation issue d'une règle de substitution. Ce coût est égal au coût de généralisation additionné au coût de la règle de substitution qui a été fixé. Dans l'exemple, si $\mathtt{coût}(\sigma_{ex}) = 0{,}1$, et comme $\mathtt{coût}(\mathtt{Fraise} \rightsquigarrow \mathtt{Baie} \circ \mathtt{FromageBlanc} \rightsquigarrow \mathtt{Fromage}) = 0{,}08$ alors le coût d'adaptation de TARTE_MYRTILLE_MASCARPONE est de 0,18.

2.5 Construction d'une base de tests

Une méthodologie d'évaluation a été construite afin d'évaluer les différentes hypothèses présentées dans les prochains chapitres de cette thèse. L'évaluation de ces différentes hypothèses consiste à comparer différentes versions du système eTaaable.

Notre méthodologie d'évaluation s'appuie sur les méthodologies d'évaluations utilisées dans le domaine de la recherche d'informations (RI). Les systèmes de RI retournent un ensemble de documents en réponse à une requête utilisateur. Dans les années 60, afin d'évaluer des systèmes d'indexation en RI, les tests de Cranfield [Gleverdon et Cleverdon, 1962, Rees, 1965, Cleverdon *et al.*, 1966, Cleverdon, 1997] ont proposé une méthodologie encore aujourd'hui très répandue. Cette méthodologie repose sur la construction d'une collection de tests composée d'un ensemble fixe de documents et d'un ensemble de requêtes. Un score de pertinence pouvant prendre 5 valeurs est associé entre chaque requête et chaque document. La collection de Cranfield (*The Cranfield collection*) qui a été distribuée, contient 1398 documents qui sont des résumés d'articles en aérodynamique, un ensemble de 225 requêtes et un ensemble exhaustif des scores de pertinence pour tous les couples (requête,document).

Par la suite, d'autres collections de tests plus larges ont été construites. Par exemple, les tests Rcv1 permettant d'évaluer des systèmes de classifications de textes qui sont une amélioration des tests Reuters−21578 et Reuters−22173 [Rose *et al.*, 2002, Lewis *et al.*, 2004]. Les tests Rcv1 classifient 800000 articles de presse dans des thèmes variés tels que les nouvelles technologies, les sciences ou la santé grâce à 400000 termes référencés, où chaque couple (requête,document) est représenté.

Les collections de test les plus répandues aujourd'hui sont issues des conférences TREC (*Text Retrieval Conference*). Les conférences TREC sont divisées en sous-tâches appelées *tracks*. Par exemple, le *Web track* consiste à évaluer des approches de recherches à grande échelle sur le Web. L'ensemble de documents utilisé par le *Web track* est ClueWeb12 qui contient environ 800 million de documents. Les documents dans chaque sous-tâche sont interrogés par 25 à 50 requêtes ou thèmes. Avec des millions de documents, l'évaluation des couples (requête,document) ne peut pas être exhaustive. Ainsi, la méthodologie des conférences TREC utilise la technique de mise en commun (*pooling*). Pour une requête, seuls les n premiers documents retournés par chaque participant de la conférence sont évalués. Dans la troisième conférence TREC, il a été démontré que $n = 100$ était satisfaisant pour évaluer les performances de systèmes de RI [Harmon, 1996].

Nous nous appuyons sur cette méthodologie d'évaluation s'appuyant sur des collections de tests afin de valider les hypothèses posées dans les prochains chapitres. À cette fin, nous avons constitué une base de tests et une méthodologie pour la compléter. Notre base de tests est composée :

- d'un ensemble de plans de tests où chaque plan de tests est relié à un ensemble de requêtes. Chaque requête de nos plans de tests actuels porte sur une recette d'un type de plat donné contenant un ou plusieurs ingrédients désirés. Seuls deux types de plats différents sont représentés dans les requêtes de nos plans de tests : les tartes et les cocktails. Cette restriction permet de se focaliser sur une partie réduite de la base de cas afin de réduire le nombre de réponses différentes obtenues pour chaque requête et donc le nombre d'évaluations à acquérir. Ainsi, si le type de plat correspond aux tartes, seules des recettes de tartes seront remémorées et adaptées. Les ingrédients dans les requêtes sont choisis aléatoirement ou non en fonction des hypothèses à tester.
- d'un ensemble d'adaptations consistant en des couples (cas, substitutions). Les cas sont des recettes de aTaaable et les substitutions sont des substitutions d'ingrédients, où le nom des ingrédients correspondent aux noms des pages d'aliments de aTaaable.
- d'un ensemble d'évaluations sous forme de scores pour chaque adaptation. Chaque score d'évaluation est donné par un évaluateur qui est un utilisateur de l'e-communauté de aTaaable.

2.5.1 Description de la base de test

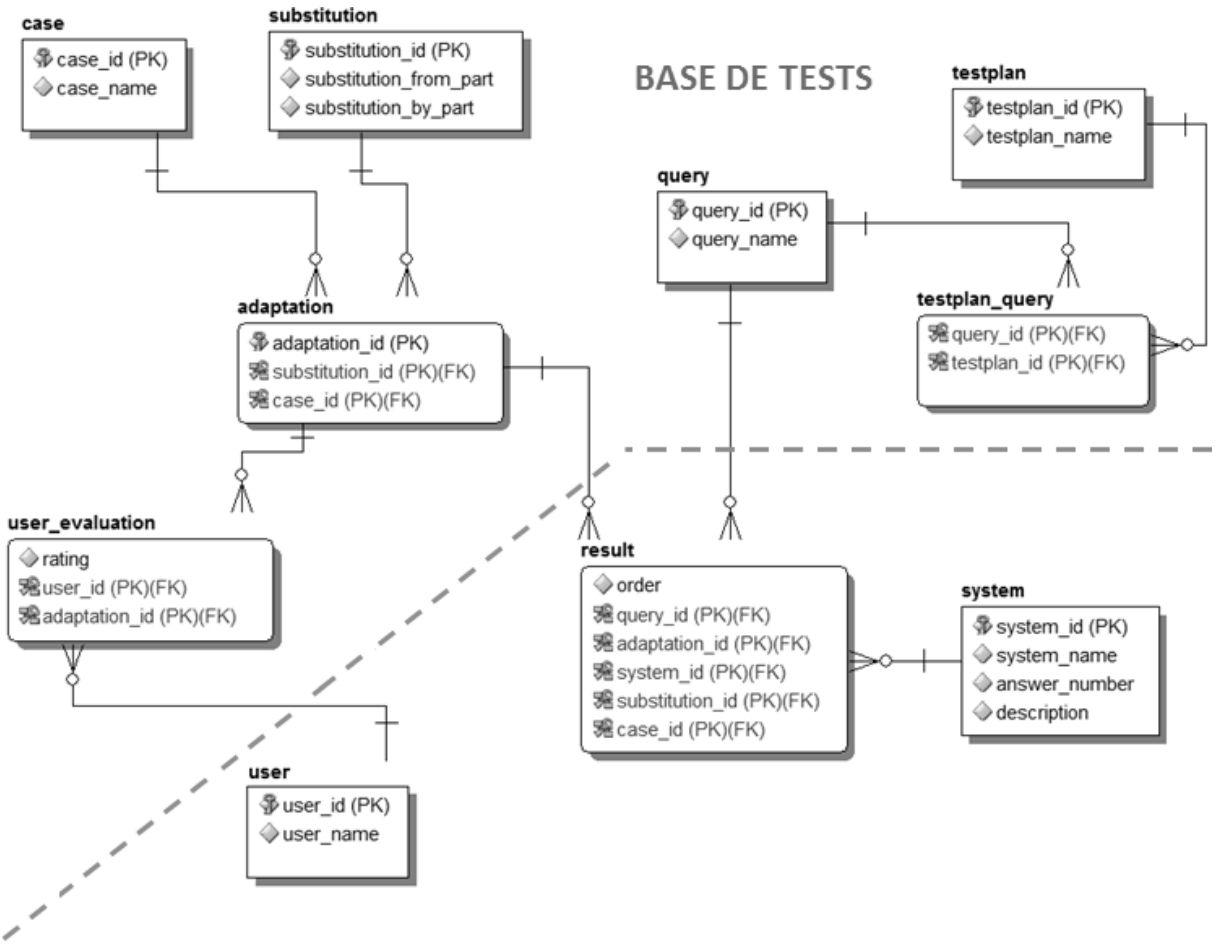


FIGURE 2.9 – Schéma UML de la base de données de la base de tests, des évaluateurs et réponses des systèmes.

La figure 2.9 représente le modèle UML de la base de données qui stocke la base de tests, l'identité des évaluateurs ayant participés aux évaluations pour acquérir les scores d'évaluation des adaptations de la base de tests et les réponses des systèmes à comparer. Les testeurs sont représentés par leur identifiant (`user_id`) et leur nom. La partie représentant la base de tests est décrite comme suit :

- Un plan de tests représenté par son identifiant (`testplan_id`) est associé à un ensemble de requêtes représentées par leur identifiant (`query_id`). Les versions de eTaaable à comparer pour valider une hypothèse sont toutes interrogées par les requêtes du même plan de tests.
- Une adaptation est décrite par son identifiant (`adaptation_id`), un identifiant de cas (`case_id`) et un identifiant de substitutions (`substitution_id`).
- Une substitution est décrite par son identifiant (`substitution_id`), un ensemble d'ingrédients substitués encodés sous forme d'une chaîne (`substitution_from_part`) et un ensemble d'ingrédients substituants encodés sous forme d'une chaîne (`substitution_to_part`).

– Un score d'évaluation (`rating`) est relié à l'identifiant d'une adaptation et à l'identifiant de l'utilisateur (`user_id`) qui a attribué le score.

L'ensemble des adaptations proviennent des réponses des différents systèmes comparés qui sont différentes versions d'eTaaable[23]. La réponse d'un système est représentée par son identifiant (`result_id`) et est liée à l'identifiant de la requête dont elle est issue (`query_id`), à l'identifiant du système qui l'a retournée (`system_id`), à l'ordre dans lequel elle a été retournée (`order`) et à l'identifiant de l'adaptation (`adaptation_id`) qu'elle a générée. Un système est représenté par son identifiant (`system_id`), son nom, le nombre de réponses qu'il doit retourner et sa description. Comme pour la méthodologie des conférences TREC, nous utilisons la technique de mise en commun (*pooling*). En effet, seules les adaptations générées par les systèmes à comparer sont évaluées, car il ne serait pas possible d'évaluer toutes les réponses possibles.

2.5.2 Méthodologie de la construction de la base de test

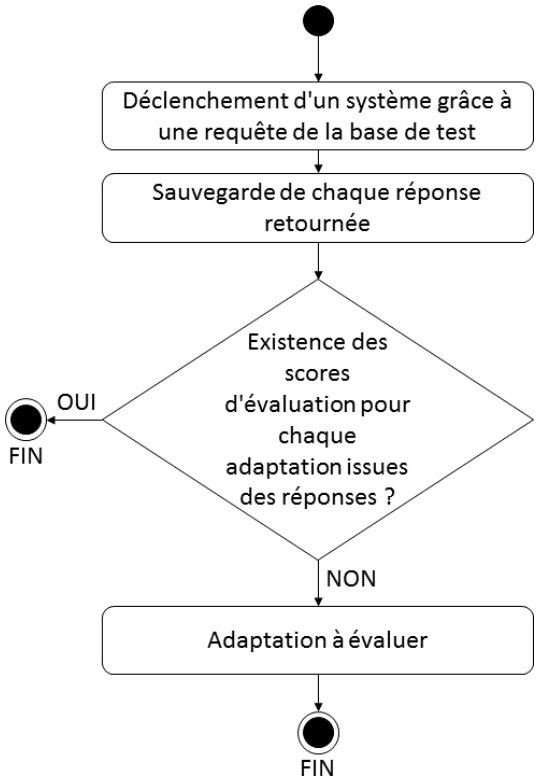


Figure 2.10 – Calcul des réponses contenant les adaptations à évaluer.

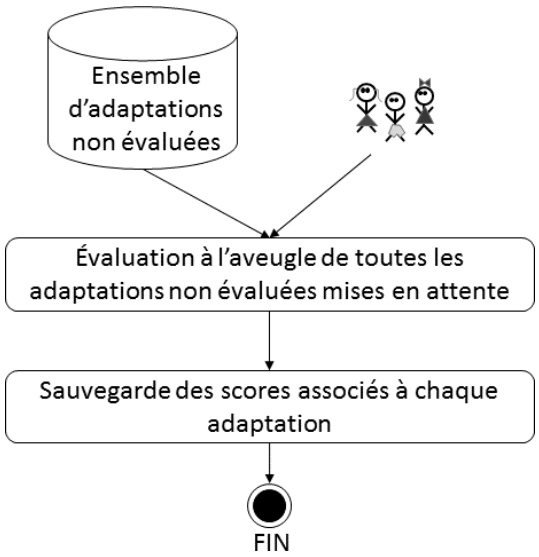


Figure 2.11 – Évaluation des adaptations.

Pour chaque plan de tests, l'ensemble des requêtes est construit et la base de tests doit être complétée avec des adaptations. D'après la figure 2.10, ces adaptations proviennent des réponses des différents systèmes à comparer durant cette thèse qu'on veut interroger avec les requêtes du plan de tests. Pour chaque système à comparer, on fixe à l'avance combien de réponses doit retourner le système et à quel plan de tests existant le système est associé. Les systèmes sont ensuite interrogés par les requêtes du plan de test. Les réponses des systèmes sont stockées dans la base de tests. Si l'adaptation composée d'un cas et d'une ou plusieurs substitutions n'existe

23. Lorsque la base de tests sera distribuée, des réponses provenant d'autres systèmes seront stockées.

pas encore dans la base de tests, cette adaptation est créée dans la base de tests, sinon aucune nouvelle adaptation n'est créée. Cette méthode d'acquisition des adaptations du plan de test est donc incrémentale. Plus le nombre de différents systèmes interrogés augmente, moins le nombre de nouvelles adaptations générées est important.

Dès lors que les réponses d'au moins un système ont été stockées, les adaptations peuvent être évaluées. D'après la figure 2.11, les adaptations sont évaluées à l'aveugle par des évaluateurs qui sont des utilisateurs de aTaaable. Les évaluateurs ne connaissent pas l'origine de l'adaptation, c'est-à-dire de quel système l'adaptation provient, d'autant plus qu'une adaptation a pu être générée par plusieurs systèmes. Chaque adaptation doit être évaluée par trois différents évaluateurs.

Comme le nombre d'adaptations nouvelles générées diminue au fil des systèmes interrogés, le nombre d'évaluations à acquérir diminue lui aussi. Le but à long terme est de pouvoir partager cette base de tests en dehors du cadre de cette thèse pour pouvoir évaluer les réponses de nouveaux systèmes construits dans le futur.

L'évaluation des adaptations se fait par les évaluateurs grâce à une interface présentée dans la figure 2.12 et disponible sur aTaaable. Pour faciliter les évaluations, l'évaluation des adaptations concernant une même recette se fait sur une interface unique. Si dans les réponses des systèmes à comparer, une même recette a été adaptée de 3 façons différentes, alors l'évaluateur aura à faire 3 évaluations sur cette même recette. Dans l'exemple illustré en figure 2.12, la recette « Quiche poireaux saumon » a été adaptée au moins une fois par chacune des substitutions suivantes : (1) remplacer le poireau par de l'oignon, (2) remplacer le poireau par de la rhubarbe et (3) remplacer le saumon par des lardons.

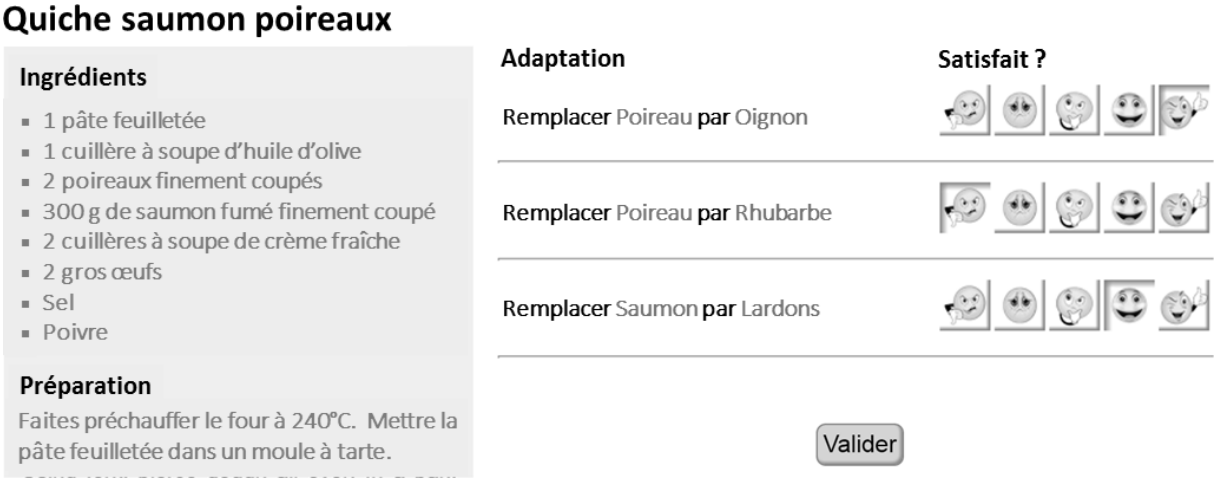


Figure 2.12 – Interface d'évaluation. L'utilisateur doit évaluer à quel point il est satisfait de certaines adaptations apportées à la recette affichée. Dans cette copie d'écran, les adaptations concernant la recette de « Quiche poireaux saumon ». L'utilisateur est très satisfait par la substitution `Poireau` $\rightsquigarrow$ `Oignon`, pas du tout satisfait par la substitution `Poireau` $\rightsquigarrow$ `Rhubarbe` et satisfait par la substitution `Saumon` $\rightsquigarrow$ `Lardons`.

Beaucoup de possibilités existent pour acquérir le retour de satisfaction des utilisateurs. Une d'elle est l'échelle de Likert [Likert, 1932], qui est composée d'un ensemble de modalités, permettant à l'utilisateur de donner son degré de satisfaction. L'échelle la plus fréquente est composée de 5 modalités, ne donnant ni trop, ni trop peu d'options aux utilisateurs pour s'exprimer. Parmi

ces 5 modalités, 2 sont positives (très satisfait, satisfait), 1 est neutre (ni satisfait, ni insatisfait) et 2 sont négatives (insatisfait et très insatisfait). Un avantage de cette échelle est que les différentes modalités peuvent être converties en des scores numériques. Par conséquent, nous avons choisi cette échelle à 5 modalités pour acquérir les scores de satisfaction des utilisateurs à propos d'une adaptation. Cependant, au lieu de choisir une échelle de mesure sémantique, c'est-à-dire une échelle dont les valeurs sont représentées par du texte, nous avons choisi une échelle visuelle où chaque valeur correspond à un émoticône. Tout comme pour l'échelle sémantique, on associe une valeur numérique à chaque émoticône :

- -2, l'émoticône « pas du tout content » () indiquant que l'utilisateur est très insatisfait ;
- -1, l'émoticône « pas content » () indiquant que l'utilisateur est insatisfait ;
- 0, l'émoticône « neutre » () indiquant que l'utilisateur n'est ni satisfait, ni insatisfait ;
- 1, l'émoticône « content » () indiquant que l'utilisateur est satisfait ;
- 2, l'émoticône « très content » () indiquant que l'utilisateur est très satisfait.

Dans l'exemple illustré en figure 2.12, l'évaluateur est très satisfait par la première substitution appliquée à la recette, très insatisfait par la deuxième substitution appliquée à la recette et satisfait par la troisième substitution appliquée à la recette.

Différents types de personnes ont participé à l'évaluation, en majorité : des étudiants, du personnel du LORIA ou des volontaire recrutés sur un forum de sciences cognitives. Tous les évaluateurs doivent être préalablement inscrits sur le site aTaaable. Les évaluateurs reçoivent des instructions écrites sur le déroulement de l'expérimentation sur l'interface de test de aTaaable. Chaque adaptation est évaluée par au moins 3 évaluateurs. 27 évaluateurs ont participé aux évaluations, 1025 adaptations ont été évaluées et 3506 notes ont été données par les différents évaluateurs.

2.5.3 Analyse des résultats des évaluations

La constitution de la base de cas a pour but final de pouvoir comparer différents systèmes par rapport aux scores de satisfaction obtenus par les réponses de ces systèmes.

La comparaison des systèmes fondée sur les résultats des évaluations est inspirée des systèmes de recommandation. Les systèmes de recommandation utilisent les techniques de filtrage collaboratif pour prédire des scores d'utilisateurs à propos d'items, par exemple des films, correspondant à une requête utilisateur et proposent les meilleurs items à l'utilisateur.

Les systèmes de recommandations et les systèmes de RàPC peuvent être considérés comme similaires car ils sont déclenchés par une requête utilisateur et ils retournent un ensemble de réponses qui peuvent satisfaire ou non l'utilisateur. Pour cette raison, et comme il l'a été fait dans [Quijano-Sánchez *et al.*, 2011], il est possible de comparer les satisfactions des réponses provenant de deux systèmes de RàPC de la même façon que sont comparées les satisfactions des recommandations de deux systèmes de recommandation.

Comparer des systèmes de recommandation. Soit A et B deux systèmes de recommandation. Tester si le système A est meilleur que le système B consiste à tester que les réponses provenant de A sont plus satisfaisantes que les réponses provenant de B [Ricci *et al.*, 2010]. Si une différence entre les deux systèmes est constatée, c'est-à-dire si A est meilleur que B, cette différence doit être significative pour conclure que A est meilleur que B. Une mesure standard pour mesurer si une différence est significative est la p-valeur dont le but est d'évaluer l'hypothèse nulle. L'hypothèse nulle suppose qu'une différence constatée par des expérimentations est obtenue par hasard. Dans le but d'obtenir des résultats significatifs, l'hypothèse nulle doit être rejetée. Si p-valeur $< 0{,}05$, alors l'hypothèse nulle est rejetée et les résultats sont significatifs.

Cela signifie que le fait que la probabilité que les résultats sont dus au hasard est inférieure à 5%.

Nous utilisons cette méthodologie pour comparer les systèmes à tester dans le cadre de cette thèse. Afin de tester si une différence entre deux systèmes à comparer est significative, un test statistique doit être choisi.

Choix du test statistique. Le test statistique approprié est choisi en fonction des différentes variables à étudier, c'est-à-dire en fonction de ce qui est contrôlé et ce qui est mesuré, et du mode de répartition des échantillons, c'est-à-dire la répartition des évaluateurs ayant participé à l'évaluation. Le choix du système est une variable indépendante, aussi appelée variable contrôlée, c'est-à-dire une variable qui impacte les résultats. Le score de satisfaction d'un utilisateur à une réponse d'un système est une variable dépendante, aussi appelée variable mesurée, qui dépend de la variable indépendante. Le score de satisfaction d'un utilisateur à une réponse d'un système est une variable ordinale non paramétrique qui n'a pas de distribution gaussienne et dont les valeurs possibles sont -2, -1, 0, 1 et 2. Dans notre méthodologie d'évaluation, chaque évaluateur peut tester les adaptations provenant de différents systèmes. En d'autres termes, un évaluateur n'est pas restreint à évaluer les adaptations des réponses d'un seul système. Par conséquent, les échantillons sont appariés.

Le test statistique requis pour évaluer la validité des résultats doit prendre en compte des variables non paramétriques et des échantillons appariés. Le test des rangs signés de Wilcoxon (*Wilcoxon signed-rank test*) [Wilcoxon et Wilcox, 1964] permet de tester que la médiane des scores de satisfaction obtenus par un système A est significativement plus élevée que la médiane des scores de satisfaction obtenus par un système B. Nous avons alors choisi le test des rangs signés de Wilcoxon pour valider nos hypothèses.

Chapitre 3

Gestion de la fiabilité des connaissances provenant d'une communauté en ligne grâce à un modèle de métaconnaissances

Sommaire

En RàPC, l'acquisition de la base de connaissances se fait classiquement par un expert afin de garantir la qualité des connaissances. Bien que l'effort d'acquisition des cas sources soit réduit grâce à la mémorisation de problèmes résolus par le système de RàPC, l'acquisition des connaissances du domaine, d'adaptations et de similarité est coûteuse. Solliciter une communauté en ligne pour alimenter les conteneurs de connaissances permet d'acquérir plus facilement

une importante quantité de connaissances mises à jour en temps réel grâce à la collaboration d'utilisateurs, réduisant ainsi le coût d'acquisition.

Cependant, l'acquisition par une communauté en ligne est influencée par certains facteurs humains tels que le niveau d'expertise des utilisateurs ou encore leur sérieux. Par conséquent, la fiabilité d'une UC introduite par un utilisateur du système peut être variable et le système de RÀPC va utiliser des connaissances n'ayant pas toutes la même fiabilité. Utiliser des connaissances non fiables dans un système de RÀPC peut entraîner une diminution de la qualité des réponses retournées par le système, en plus des réponses incorrectes pouvant être générées par le raisonnement hypothétique et non déductif du RÀPC (cf. la section 1.6.2 du chapitre 1). En utilisant des connaissances non fiables durant la phase de remémoration, les cas les plus satisfaisants peuvent ne pas être remémorés à l'avantage de cas moins satisfaisants. De plus, durant la phase d'adaptation, une « mauvaise » adaptation qui est défini comme une adaptation non satisfaisante pour les utilisateurs du système peut être appliquée à la solution du cas remémoré. Ainsi, il est nécessaire de gérer la fiabilité des connaissances lorsqu'on veut utiliser des connaissances provenant d'une communauté en ligne dans un système de RÀPC, afin que le système n'utilise que les connaissances les plus fiables, ce qui permet d'assurer une certaine qualité des réponses.

La section 3.1 de ce chapitre va présenter les principes généraux de MKM et de l'extension d'un système de RÀPC utilisant des connaissances provenant d'une communauté en ligne avec MKM. La section 3.2 présente le fonctionnement du modèle MKM dont l'objectif est de représenter la fiabilité des UC. La section 3.3 présente un exemple de calcul de fiabilité des UC d'aTaaable. La section 3.4 présente la prise en compte de la fiabilité en amont et en aval du système de RÀPC.

3.1 Principes de MKM (*Meta-Knowledge Model*) et extension d'un système de RÀPC avec MKM

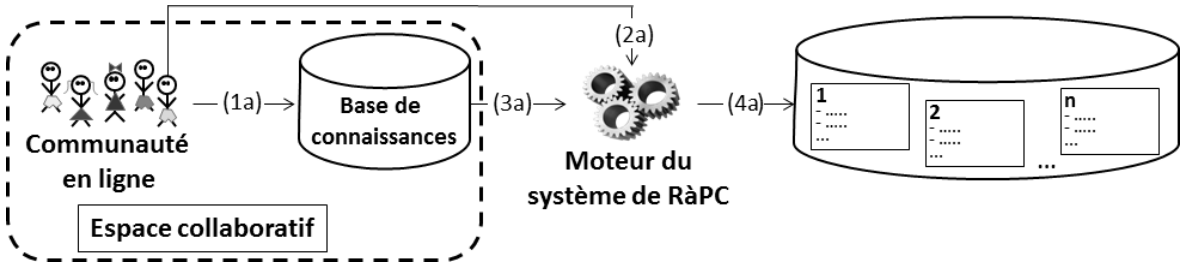
Dans le cadre de notre thèse, le modèle MKM permet d'étendre un système de RÀPC pour exploiter des connaissances provenant d'une communauté en ligne en gérant la fiabilité des UC pour les utilisateurs du système.

Cependant, MKM est modèle générique pour étendre des systèmes à base de connaissances produites par une communauté en ligne indépendamment de l'utilisation qui va être faite des connaissances ; le modèle MKM représente la fiabilité des UC produites par les utilisateurs de la communauté en ligne. MKM se traduit par l'ajout :

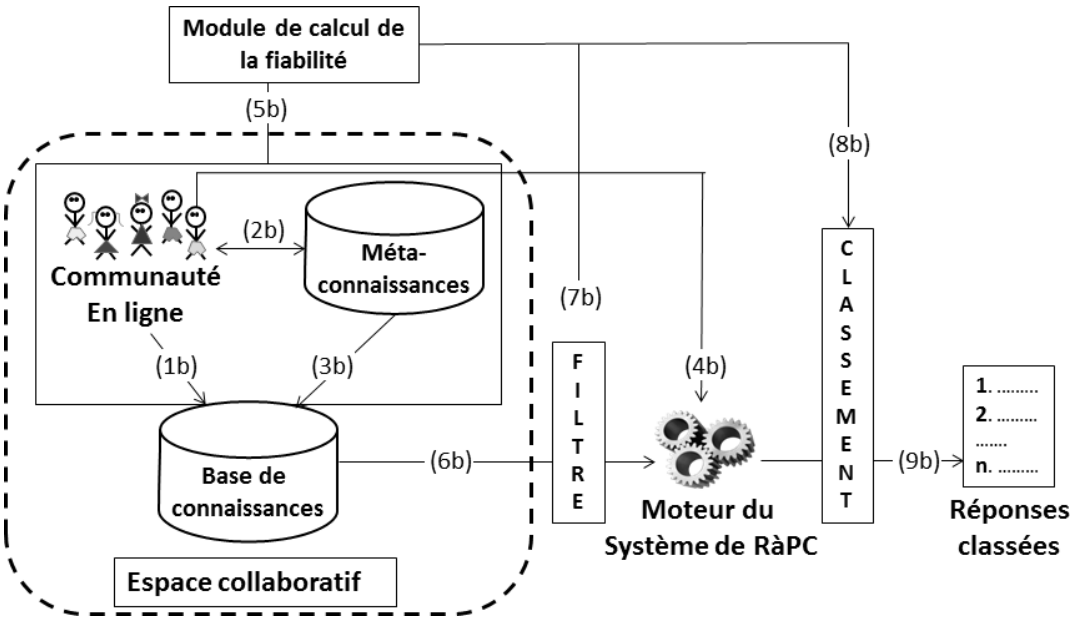
- d'un conteneur de métaconnaissances destiné à stocker les interactions des utilisateurs de la communauté en ligne dans le système. Ces interactions concernent la production d'une UC par un utilisateur, l'évaluation de la croyance qu'a un utilisateur envers une UC ou la confiance qu'a un utilisateur envers un autre utilisateur ;
- d'un module de calcul de la fiabilité d'une UC propre à chaque utilisateur de la communauté en ligne.

Étendre un système de RÀPC utilisant des connaissances provenant d'une communauté en ligne avec MKM permet de gérer la fiabilité des connaissances pour garantir la qualité des réponses retournées par le système de RÀPC. La figure 3.1 montre les points communs et les différences d'architecture entre un système de RÀPC utilisant des connaissances provenant d'une communauté en ligne sans MKM, noté RÀPC_{std} (« std » pour standard) dont les principes sont présentés en section 1.6 du chapitre 1 et le même système de RÀPC étendu avec MKM, noté RÀPC_{MKM}. Pour les points communs, les deux systèmes sont déclenchés par une requête

utilisateur (2a et 4b), utilisent le même moteur et la même base de connaissances composée des UC produites par une même communauté en ligne (1a et 1b) interagissant sur un même espace collaboratif. Dans les deux systèmes, le moteur retourne les réponses ordonnées par similarité avec les cas remémorés dont proviennent les réponses et la requête utilisateur.



(a) Architecture d'un système de RàPC classique qui utilise des connaissances provenant d'une communauté en ligne.



(b) Architecture d'un système de RàPC qui utilise des connaissances provenant d'une communauté en ligne étendu avec MKM.

FIGURE 3.1 – Comparaison de deux versions d'un système de RàPC, exploitant des UC provenant d'une communauté en ligne : une version sans l'extension MKM (a) et une version avec l'extension MKM (b).

Concernant les différences, alors que RàPC_{std} utilise une base de connaissances composée de quatre conteneurs d'UC (la base de cas, les connaissances du domaine, les connaissances d'adaptation et les connaissances de similarité), RàPC_{MKM} utilise un conteneur de métaconnaissances supplémentaire alimenté grâce aux interactions des utilisateurs dans le système qui correspondent à la production d'une UC par un utilisateur (1b), l'évaluation d'une UC (3b), ou l'évaluation d'un autre utilisateur (2b). De plus, lorsqu'un utilisateur interroge RàPC_{std} (2a),

toutes les UC de la base de connaissances sont disponibles pour être utilisées par le moteur de RàPC. À l'inverse, lorsqu'un utilisateur interroge RàPC$_{\text{MKM}}$ (4b), le module de calcul de la fiabilité de MKM calcule la fiabilité de chaque UC de la base de connaissances grâce au conteneur de métaconnaissances (5b). Le calcul de la fiabilité permet de filtrer les UC non fiables (7b) à l'entrée du moteur de RàPC$_{\text{MKM}}$. Les UC filtrées ne sont alors pas utilisées par le système de RàPC comme si elles étaient supprimées des conteneurs de connaissances.

De plus, d'après la figure 3.1, RàPC$_{\text{std}}$ retourne les réponses à l'utilisateur dans le même ordre que le moteur les retourne (4a). Avec RàPC$_{\text{MKM}}$, le calcul de la fiabilité permet aussi de classer par ordre de fiabilité décroissante (9b) les réponses retournées par le moteur de RàPC.

3.2 Fonctionnement de MKM

3.2.1 Architecture générale de MKM

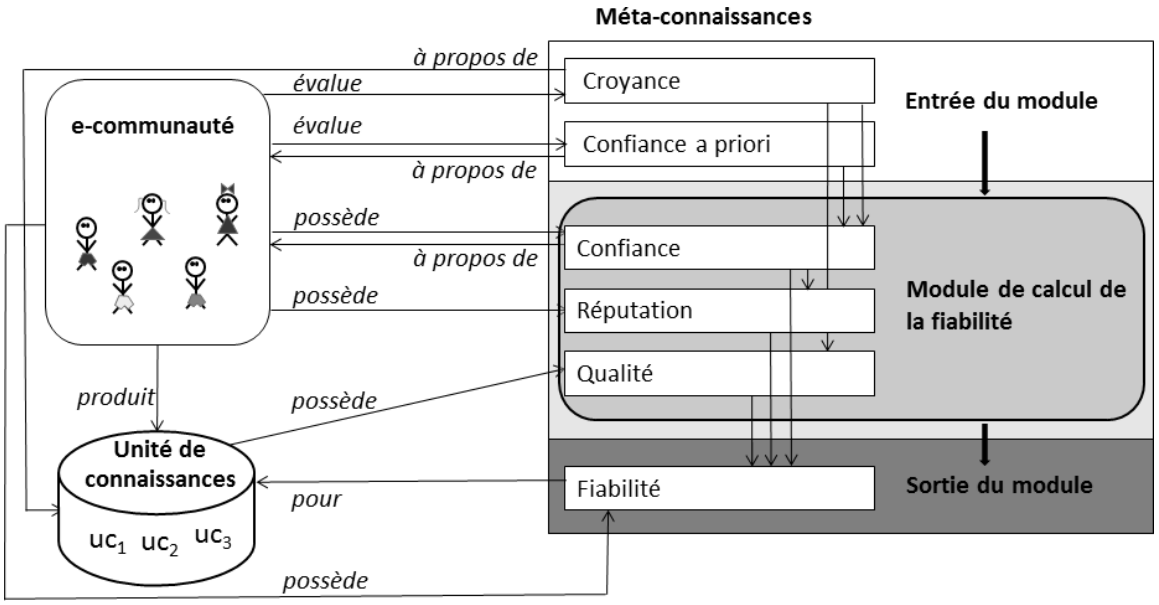


FIGURE 3.2 – MKM : dépendances entre utilisateurs, unités de connaissances et métaconnaissances.

Le module de calcul de la fiabilité est au cœur du modèle MKM. Au regard d'un utilisateur donné, la figure 3.2 montre que ce module prend en entrée les métaconnaissances stockées dans le conteneur de métaconnaissances et calcule un ensemble de métaconnaissances qui permettent de déterminer, en sortie, la fiabilité de chaque UC de la base de connaissances. Dans MKM, nous représentons chacune des métaconnaissances par un score compris entre 0 et 1. D'après la définition 6 de la section 1.2.6 du chapitre 1, la fiabilité, que nous représentons par un score entre 0 et 1, d'une UC uc pour un utilisateur u correspond à l'utilité et l'exactitude de uc pour u. Les métaconnaissances calculées par le module de calcul de la fiabilité découlent directement des interactions de la communauté en ligne dans le système. D'après la définition 6, la fiabilité d'une UC uc pour u dépend de la qualité de uc vis-à-vis de la communauté, de la confiance de u envers v qui a produit uc et de la réputation de v dans la communauté. Dans MKM, la fiabilité est donc représentée grâce aux métaconnaissances suivantes : la qualité, la confiance et

la réputation. Le détail du calcul des différentes métaconnaissances représentées dans MKM est décrit par la suite.

La figure 3.2 montre qu'un utilisateur u peut interagir dans le système de trois manières :

1. en produisant une UC ;
2. en évaluant un autre utilisateur ;
3. en évaluant une UC.

De plus, la figure 3.2 montre le lien entre les différentes métaconnaissances entrant en jeu dans le module de calcul de la fiabilité de MKM :

- un utilisateur u peut évaluer une UC uc d'un conteneur de connaissances par un score de *croyance* qui représente la croyance de u que uc est crédible, exacte et pertinente.
- Un utilisateur u peut évaluer un autre utilisateur v par un score de *confiance a priori*[24] qui représente la perception que u a envers v.
- un score de *qualité* d'une UC uc, qui représente la perception globale de la communauté sur l'exactitude et la crédibilité de uc, est calculé à partir de tous les scores de *croyance* attribués à uc par tous les utilisateurs de la communauté en ligne.
- un score de *confiance* d'un utilisateur u envers un autre utilisateur v, qui représente la perception de u envers v et envers les UC produites par v en fonction de leurs interactions passées, est calculé à partir du score de *confiance a priori* que u a assigné à v et à partir des scores de *croyance* que u a attribué aux différentes UC produites par v.
- un score de *réputation* d'un utilisateur v, qui représente la perception globale de la communauté sur v, est calculé à partir de tous les scores de *confiance* que les utilisateurs de la communauté ont envers v.
- un score de *fiabilité* d'une UC uc pour un utilisateur u, qui représente l'utilité et de l'exactitude de uc pour u est calculé à partir de la *qualité*, de la *confiance* et de la *réputation*.

Pour résumer la figure 3.2, les métaconnaissances représentées sur fond blanc (croyance et *confiance a priori*) sont les métaconnaissances stockées dans le conteneur de métaconnaissances et sont acquises à partir des interactions des utilisateurs avec le système. Les métaconnaissances sur fond gris clair (qualité, confiance et réputation) sont les métaconnaissances calculées par le module ; leur score respectif change en fonction des interactions des utilisateurs dans le système. Les scores de qualité, de confiance et de réputation ne sont pas stockés et sont calculés à nouveau à chaque fois que le score de fiabilité d'une UC doit être calculé. Finalement, la fiabilité représentée sur fond gris foncé, est la métaconnaissance en sortie du modèle. Le score de fiabilité n'est pas non plus stocké et est calculé à chaque fois que le score de fiabilité d'une UC doit être connu. Par exemple, si MKM étend un système de RÀPC, le score de fiabilité est calculé à chaque fois qu'un utilisateur interroge le système de RÀPC.

Les sections suivantes présentent en détail la justification de chacune des métaconnaissances qui viennent d'être introduites ainsi que leur calcul. Nous nous appuyons sur les définitions 1 à 6 que nous avons proposées dans le chapitre 1 et qui reposent sur notre étude bibliographique faite dans ce même chapitre.

24. À partir de maintenant, nous écrivons *a priori* sans utiliser l'écriture italique, pour alléger la lecture.

3.2.2 La croyance

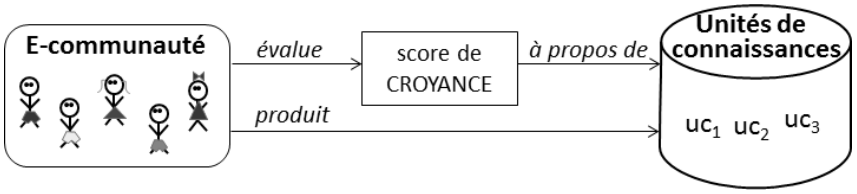


FIGURE 3.3 – Productions et évaluations d'UC. Le score de croyance résulte des évaluations des utilisateurs à propos des UC dans le système.

D'après la définition 1 de la section 1.2.1 du chapitre 1, la croyance d'une UC uc pour un utilisateur u correspond au degré d'acceptation subjective que uc soit exacte ou crédible pour u. Nous avons étudié dans les sections 1.2.1 et 1.4 du chapitre 1, que certains systèmes représentent la croyance d'un utilisateur envers une information par un score collecté grâce à l'évaluation de l'utilisateur à propos de cette information. Soit $\mathtt{U}$ l'ensemble des utilisateurs et soit $\mathtt{UC}$ l'ensemble des UC produites. D'après les précédentes constatations, dans MKM, quand un utilisateur $u \in \mathtt{U}$ évalue une $uc \in \mathtt{UC}$, u assigne un score de croyance à uc, $\mathtt{croyance}(u,uc)$.

$$\mathtt{croyance} : \mathtt{U} \times \mathtt{UC} \rightarrow [0,1] \cup \{\mathtt{?}\}$$

où ? représente la valeur inconnue ($\mathtt{croyance}(u,uc) = \mathtt{?}$ signifie que u n'a pas évalué uc).

D'après la figure 3.3, le score de croyance correspond à l'évaluation d'un utilisateur à propos d'une UC qui a été produite par un utilisateur de la communauté en ligne. Les utilisateurs peuvent évaluer les UC avec un système d'évaluation, par exemple à étoiles comme sur le site d'e-Bay [25]. Un exemple d'intégration d'un système d'évaluation des UC produites par une communauté en ligne est présenté dans la section 3.3 de ce chapitre. Dans le modèle MKM, et comme l'indique la fonction `croyance`, ces scores d'évaluation sont normalisées dans l'intervalle [0,1]. Le score de croyance entre un utilisateur u et une UC uc est unique. Ce score peut être modifié par une nouvelle action d'évaluation de u envers uc. Par ailleurs, lorsqu'un utilisateur u produit une UC uc, un score de croyance de 1 est automatiquement attribué à uc. Cependant, si u vient à ne plus croire (partiellement ou totalement) uc , alors u peut effectuer une action d'évaluation vers uc pour mettre à jour son score de croyance à propos de uc.

3.2.3 La confiance a priori

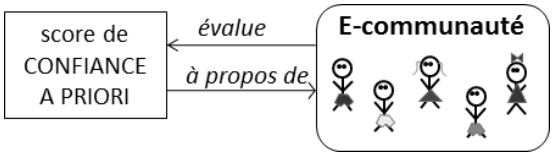


FIGURE 3.4 – Évaluations des utilisateurs. Le score de confiance a priori résulte des évaluations des utilisateurs entre eux.

25. `http://ebay.fr/`

D'après la définition 4 de la section 1.2.4 du chapitre 1, la confiance qu'a un utilisateur u sur un autre utilisateur v correspond à la perception que u a de v et de la perception qu'a u envers les connaissances produites par v. Nous nous intéressons dans ce paragraphe à la perception que u a de v qui peut être directement représentée grâce à l'évaluation de u à propos de v dans le système. En effet, nous avons vu dans la section 1.3 du chapitre 1 que certains systèmes représentent la perception que u a de v par un score collecté grâce à l'évaluation de u à propos de v. D'après ces constatations, dans MKM, quand un utilisateur u évalue un autre utilisateur v, u assigne un score de confiance a priori à l'utilisateur v. L'utilisateur u assigne un score de confiance a priori envers v par rapport aux informations subjectives que u possède à propos de v : par exemple, la description de v sur le site collaboratif ou les expériences passées qui permettent à u de se faire une opinion sur v. La fonction `confiance_a_priori` retourne le score de confiance a priori de l'utilisateur u envers l'utilisateur v, pour u,$v \in \mathtt{U}$ et $u \neq v$. La fonction `confiance_a_priori` est représentée par :

$$\mathtt{confiance\_a\_priori} : \mathtt{U} \times \mathtt{U} \rightarrow [0,1] \cup \{?\}$$

où ? représente la valeur inconnue : $\mathtt{confiance\_a\_priori}(u,v) = ?$ signifie que u n'a pas évalué v.

D'après la figure 3.4, la confiance a priori correspond au score d'évaluation d'un utilisateur à propos d'un autre utilisateur.

Les utilisateurs peuvent évaluer les autres utilisateurs de la communauté en ligne grâce au même système d'évaluation que pour l'évaluation des UC. Un exemple d'intégration d'un système d'évaluation des utilisateurs d'une communauté en ligne est présenté dans la section 3.3 de ce chapitre. Dans le modèle MKM, et comme l'indique la fonction `confiance_a_priori`, ces scores d'évaluation sont normalisées dans l'intervalle [0,1].

Les scores de croyance et de confiance a priori vont permettre de calculer les scores de qualité, de confiance et de réputation.

3.2.4 La qualité.

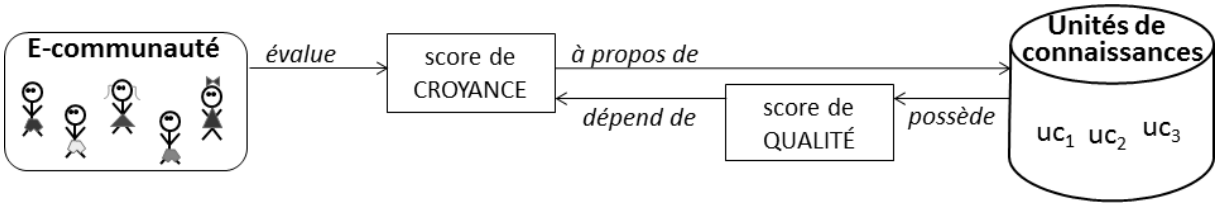


FIGURE 3.5 – La qualité d'une UC uc pour la communauté en ligne.

D'après la définition 2 de la section 1.2.6 du chapitre 1, la qualité d'une UC uc représente la perception de la communauté sur l'exactitude et la crédibilité de uc. De plus, d'après la définition 1, la croyance de uc pour un utilisateur u représente le degré d'acceptation que la connaissance soit crédible et exacte pour u. D'après ces deux définitions, nous observons dans la figure 3.5 que le score de qualité de uc dépend des scores de croyance assignés à uc par les utilisateurs de la communauté.

Le multiensemble des scores de croyance d'une UC uc, noté $\mathtt{croyance\_uc}(uc)$, représente toutes les évaluations que la communauté a assigné à uc :

$$\begin{aligned} \texttt{croyance\_uc} : \texttt{UC} &\to \mathcal{M}([0,1]) \\ uc &\mapsto \{\texttt{croyance}(u,uc) \mid u \in \texttt{U}\} \setminus \{?\} \end{aligned}$$

où $\mathcal{M}([0,1])$ désigne l'ensemble des multi-ensembles sur l'ensemble [0,1]. La notation des multi-ensembles est introduit en annexe C.

Nous définissons la qualité d'une UC uc, notée $\texttt{qualité\_communautaire}(uc)$, comme la moyenne arithmétique de $\texttt{croyance\_uc}(uc)$:

$$\begin{aligned} \texttt{qualité\_communautaire} : \texttt{UC} &\to [0,1] \cup ? \\ uc &\mapsto \frac{\sum\{s \in \texttt{croyance\_uc}(uc) \mid s \neq ?\}}{|\texttt{croyance\_uc}(uc)|} \end{aligned} \tag{3.1}$$

Nous avons choisi d'utiliser la moyenne arithmétique pour calculer la qualité par commodité. En revanche, d'autres possibilités comme l'utilisation de la fonction minimum ou maximum sont envisageables. Cependant, l'utilisation du minimum ou du maximum pourrait entraîner certains biais car elle favoriserait les avis extrêmes et ainsi l'avis d'utilisateurs malveillants ou d'utilisateurs « non experts » seraient uniquement pris en compte. Une solution à ce problème pourrait être l'utilisation de la médiane. L'influence du choix d'une telle fonction d'agrégation fait l'objet de futurs travaux.

3.2.5 La confiance.

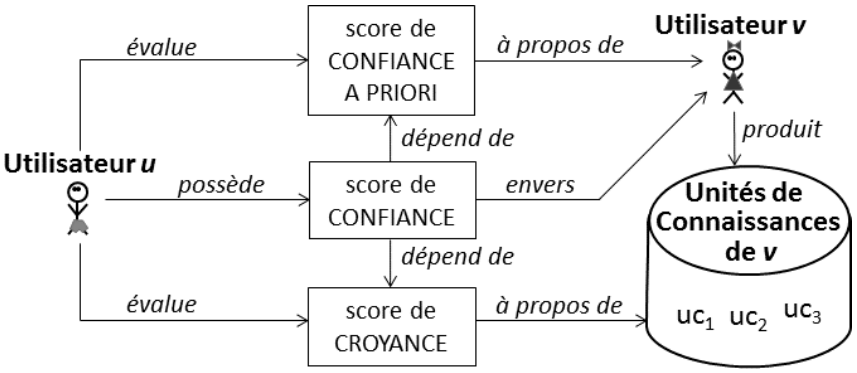


FIGURE 3.6 – La confiance d'un utilisateur u envers un utilisateur v dans la communauté en ligne.

D'après la définition 4 de la section 1.2.4 du chapitre 1, la confiance d'un utilisateur u envers un autre utilisateur v correspond à la perception que u a de v et de la perception que u a sur les connaissances produites par v. Dans la section précédente, la perception que u a de v est représentée par la confiance a priori de u envers v. De plus, la perception que u a sur une connaissance uc produite par v correspond au score de croyance, s'il existe, de u à propos de uc. Ainsi, la perception de u sur les connaissances produites par v est représentée par tous les scores de croyance que u a attribué aux UC produites par v. À partir de ces constatations, nous observons sur la figure 3.6 que le score de confiance dépend du score de confiance a priori assigné par u pour v, et de tous les scores de croyance assignés par u aux connaissances produites par v.

Pour calculer le score de confiance qu'a u envers v, il est nécessaire de retourner l'ensemble des UC produites par v. La fonction `uc_de` retourne les UC produites par l'utilisateur u. Pour $u,v \in \texttt{U}$, $\texttt{uc\_de}(u) \cap \texttt{uc\_de}(v) = \emptyset$ (une $uc \in \texttt{UC}$ est associée à un seul producteur) :

$$\texttt{uc\_de} : \texttt{U} \to 2^{\texttt{UC}}$$

où $2^{\texttt{UC}}$ est l'ensemble des sous-ensembles de `UC`. Cette notation est introduite en annexe C.

Le multiensemble des scores de croyance que u a assigné aux connaissances produites par v est retourné par la fonction $\texttt{croyance\_utilisateur}(u,v)$; il résulte des fonctions $\texttt{uc\_de}(v)$ et $\texttt{croyance}(u,uc)$ où uc est une connaissance que v a produit :

$$\begin{aligned}\texttt{croyance\_utilisateur} : \texttt{U} \times \texttt{U} &\to \mathcal{M}([0,1]) \\ (u,v) &\mapsto \{\texttt{croyance}(u,uc) \mid uc \in \texttt{uc\_de}(v)\} \setminus \{\texttt{?}\}\end{aligned}$$

$\texttt{confiance}(u,v)$ est la mesure de la confiance de u envers v.

$$\texttt{confiance} : \texttt{U} \times \texttt{U} \to [0,1] \cup \{\texttt{?}\}$$

Soit n le nombre de scores de croyance que u a attribué aux UC produites par v : $n = |\texttt{croyance\_utilisateur}(u,v)|$. Le score de confiance est calculé comme suit, pour $u,v \in \texttt{U}$:

- Si u a un score de confiance a priori envers v, c'est-à-dire si $\texttt{confiance\_a\_priori}(u,v) \neq \texttt{?}$ et que u a un score de croyance envers au moins une UC produite par v, c'est-à-dire si $n \neq 0$, alors la confiance que u a envers v est calculée par la combinaison du score de confiance a priori que u a envers v et des scores de croyance que u a envers les UC produites par v :

$$\begin{aligned}\texttt{confiance}(u,v) = \alpha_n \, &\texttt{confiance\_a\_priori}(u,v) \\ &+ (1-\alpha_n)\,\frac{1}{n} \sum_{s \in \texttt{croyance\_utilisateur}(u,v)} s \qquad (3.2)\end{aligned}$$

$$\text{où } \alpha_n = \frac{1}{n+1}$$

- Sinon,
  - Si u n'a pas assigné de score de confiance a priori envers v (i.e., $\texttt{confiance\_a\_priori}(u,v) = \texttt{?}$) et si u a déjà attribué un score de croyance envers les UC produites par v, c'est-à-dire si $n \neq 0$, alors, la confiance est la moyenne des scores de croyance que u a assigné aux UC produites par v.

$$\texttt{confiance}(u,v) = \frac{1}{n} \sum \{s \in \texttt{croyance\_utilisateur}(u,v) | s \neq \texttt{?}\} \qquad (3.3)$$

    Dans l'équation (3.2), plus le nombre de scores de croyance assignés par u aux UC produites par v est important, moins le score de confiance a priori a d'influence (car $\lim_{n\to\infty} \alpha_n = 0$ et le score de confiance a priori est borné) : asymptotiquement, les expressions (3.3) et (3.2) sont équivalentes.
  - Sinon,
    - Si u n'a jamais attribué de score de croyance envers les UC produites par v, c'est-à-dire si $n = 0$, mais si u a assigné un score de confiance a priori envers v alors $\texttt{confiance}(u,v) = \texttt{confiance\_a\_priori}(u,v) \in [0,1] \cup \{\texttt{?}\}$.
    - Sinon lorsque $n = 0$ et $\texttt{confiance\_a\_priori}(u,v) = \texttt{?}$, alors la valeur de la confiance est égale à la valeur inconnue `?`.

Dans MKM, nous rejetons la transitivité de la confiance (notion discutée dans la section 1.3.3 du chapitre 1). Nous ne supposons rien sur le score de confiance de u envers w si nous connaissons le score de confiance de u envers v et le score de confiance de v envers w. En effet, par exemple Pierre peut avoir un score de confiance proche de 1 envers Bob car il a attribué des scores de croyance élevés pour des recettes produites par v mais Bob peut être de mauvais conseils sur les autres utilisateurs ou les autres recettes. Nous avons vu dans la section 1.3.3 du chapitre 1 que ce problème de transitivité de la confiance peut être contourné en utilisant la confiance de recommandation. Cette solution requiert d'ajouter des scores sur les capacités de recommandations des utilisateurs.

Dans le cadre de cette thèse, nous avons fait le choix de ne pas utiliser cette méthode et rejeter la transitivité de la confiance pour deux raisons :

- acquérir ou calculer des scores de recommandation nécessiterait un modèle beaucoup plus complexe et difficile à mettre en place dans une application.
- la réputation d'un utilisateur v, présentée dans le paragraphe suivant, comble l'absence d'un score de confiance de u envers v.

Cependant, la prise en compte de la confiance par transitivité pourra faire l'objet de futurs travaux.

3.2.6 La réputation

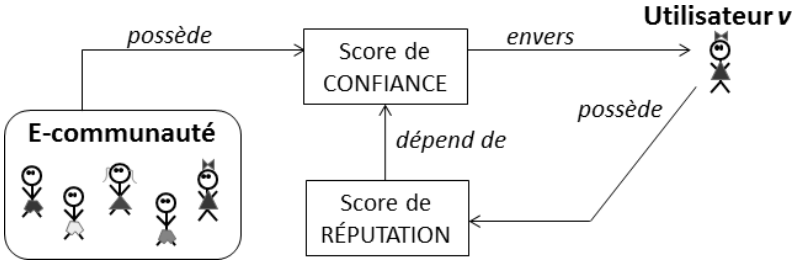


FIGURE 3.7 – La réputation d'un utilisateur v pour la communauté en ligne.

D'après la définition 5 de la section 1.2.4 du chapitre 1, la réputation d'un utilisateur v correspond à la perception collective de v et des connaissances produites par v. Comme la confiance d'un utilisateur u envers v correspond à la perception que u a de v et de la perception que u a des connaissances produites par v (cf. section précédent), alors le score de réputation de v dépend des scores de confiance de chaque utilisateur de la communauté envers v. Nous observons cette dépendance sur la figure 3.7.

Le multiensemble des scores de confiance de la communauté envers v est retourné par la fonction `confiance_communauté` :

$$\begin{aligned} \texttt{confiance\_communauté} : \texttt{U} &\rightarrow \mathcal{M}([0,1]) \\ v &\mapsto \{\texttt{confiance}(u,v) \mid u \in \texttt{U}\} \setminus \{\texttt{?}\} \end{aligned}$$

Le multiensemble des scores de confiance contient chaque score de confiance de chaque utilisateur u de la communauté envers v si ce score de confiance n'est pas égal à la valeur inconnue, c'est-à-dire si u a déjà assigné un score de croyance aux UC produites par v ou si u a assigné un score de confiance a priori à v.

réputation(v) est une estimation de la mesure de confiance qu'a toute la communauté envers v.

$$\texttt{réputation} : \texttt{U} \rightarrow [0,1] \cup \{\texttt{?}\}$$

Pour $v \in \texttt{U}$, le score de réputation est calculée comme suit :
– si $|\texttt{confiance\_communauté}(v)| \geqslant \tau$, alors

$$\texttt{réputation}(v) = \frac{\sum_{u \in \texttt{U}, u \neq v} \texttt{confiance}(u,v)}{|\texttt{confiance\_communauté}(v)|} \tag{3.4}$$

– sinon

$$\texttt{réputation}(v) = \texttt{?} \tag{3.5}$$

Le score de réputation de v est la moyenne arithmétique des scores de confiance où $\texttt{confiance}(u,v) \neq \texttt{?}$. Pour les utilisateurs qui ont obtenu un nombre d'évaluations inférieur à un seuil donné τ, le score de réputation est de valeur inconnue, et est noté ?.

Dans le cas où le score de confiance de u envers v existe, le score de réputation de v permet de renforcer ou d'atténuer le score de confiance dans le calcul du score de fiabilité qu'a u envers une UC produite par v.

De la même façon que pour la qualité, d'autres opérateurs que la moyenne arithmétiques pourraient être envisagés pour calculer le score de réputation.

3.2.7 La fiabilité

Pour un utilisateur u, la fiabilité d'une UC uc produite par un utilisateur v est la représentation de l'utilité et de l'exactitude de uc et dépend du score de la qualité de uc, du score de la confiance que u a envers v et du score de la réputation de v.

$$\texttt{fiabilité} : \texttt{U} \times \texttt{UC} \rightarrow [0,1]$$

Si l'utilisateur u a évalué l'UC uc, c'est-à-dire, $\texttt{croyance}(v,uc) \neq \texttt{?}$, alors :

$$\texttt{fiabilité}(u,uc) = \texttt{croyance}(v,uc)$$

Sinon :

$$\begin{aligned} \texttt{fiabilité}(u,uc) = \; & w_{qualité}\ \texttt{qualité\_communautaire}(uc) \\ + \; & w_{confiance}\texttt{confiance}(u,v) \\ + \; & w_{réputation}\ \texttt{réputation}(v) \end{aligned} \tag{3.6}$$

où $w_{qualité} + w_{confiance} + w_{réputation} = 1$. $w_{qualité}$ est le poids du score de qualité, $w_{confiance}$ est le poids du score de confiance et $w_{réputation}$ est le poids du score de réputation et dans le calcul du score de fiabilité. Si $w_{confiance} \neq 0$, alors la fiabilité est fondée sur la confiance de u envers v. Par conséquent, le score de fiabilité d'une connaissance pour un utilisateur est personnalisé car le score de confiance envers v varie d'un utilisateur à l'autre. Si un système de RÀPC qui utilise

les connaissances provenant d'une communauté en ligne est étendu avec MKM et que les scores de fiabilité calculés par MKM prennent en compte la confiance, alors ce système de RàPC est un système de RàPC personnalisé.

Cette fonction suppose que `qualité_communautaire` (uc) $\neq$ `?`, `confiance`(u,v) $\neq$ `?` et `réputation` (v) $\neq$ `?`. Si `qualité_communautaire`(uc) = `?`, `confiance`(u,v) = `?` et/ou `réputation` (v) $\neq$ `?` alors ils ne sont pas pris en compte dans la somme. Il peut arriver que `qualité_communautaire`(uc) = `?`, `confiance`(u,v) = `?` et `réputation` (v) $\neq$ `?` lorsque l'utilisateur v qui a produit uc n'a jamais produit d'UC évaluée par les autres utilisateurs et n'a jamais été évalué par une autre utilisateur. Dans ce cas, `fiabilité`(u,uc) = 0,5.

Il pourrait être envisagé d'utiliser des scores de recommandation d'une UC uc d'un utilisateur v pour calculer la fiabilité de uc pour un utilisateur u. Plus précisément, le score de croyance d'un utilisateur v pourrait être utilisé pour calculer la fiabilité de uc pour u s'il existe un score de confiance de u envers v. Cependant, l'utilisation de tels scores demande un modèle complexe. En effet, en fonction du score de confiance de u envers v, le score de croyance de uc pour v devra être utilisé différemment. Par exemple, avec un score de confiance de 0,9 de u envers v, le score de recommandation de uc pour v pourrait être égal au score de croyance de uc pour v. Cependant, lorsque le score de confiance est inférieur, le calcul du score de recommandation est plus discutable. L'influence du choix d'utiliser des scores de recommandation fait l'objet de futurs travaux.

3.3 Extension d'eTaaable par le module de calcul de la fiabilité de MKM

Pour étendre eTaaable avec MKM, le module de calcul de la fiabilité de MKM a été ajouté entre aTaaable, le site web collaboratif contenant les UC utilisées par le moteur eTaaable, et le moteur d'eTaaable. Ce module de calcul de la fiabilité demande de prendre en entrée les métaconnaissances d'un conteneur de métaconnaissances. Le conteneur de métaconnaissances a été ajouté aux conteneurs de connaissances que eTaaable utilise. Ce conteneur de métaconnaissances, tout comme les autres conteneurs, est enrichi sur aTaaable. L'ajout des métaconnaissances au conteneur de métaconnaissances est permis grâce aux actions de production et d'évaluation des utilisateurs de la communauté en ligne de aTaaable. À chaque fois qu'un utilisateur u produit une UC uc, la métaconnaissance reliant u comme étant le producteur de uc est stockée dans le conteneur de métaconnaissances. De plus, un système de notation des UC et des utilisateurs de aTaaable a été créé afin de permettre aux utilisateurs d'évaluer chaque UC et chaque utilisateur d'aTaaable. Ce système d'évaluation permet, à chaque action d'évaluation par un utilisateur, de stocker une métaconnaissance dans le conteneur de métaconnaissances de aTaaable. Cette métaconnaissance correspond à un score de croyance dans le cas où une UC a été évaluée, et à un score de confiance a priori dans le cas où un utilisateur a été évalué.

3.3.1 Intégration d'un système de notations des unités de connaissance dans aTaaable

Afin que les utilisateurs inscrits puissent évaluer les UC et les autres utilisateurs, un système de notation a été intégré au site aTaaable. Le système de notation permet aux utilisateurs de donner une réponse sur l'échelle de Lickert contenant 5 émoticônes correspondant à 5 choix possibles.

Le système de notation apparaît au niveau de chaque UC à évaluer, c'est-à-dire sur chaque page de recette permettant d'évaluer la recette, sur chaque page de règle de substitution permettant d'évaluer la règle de substitution et sur chaque page représentant le nom d'une classe permettant d'évaluer la relation de subsomption avec sa super-classe. La figure 3.8 présente le système de notation intégré à la page de la recette « Tarte framboise et mascarpone ». L'affirmation « Tarte framboise et mascarpone est une bonne recette » est évaluée. Pour guider l'utilisateur, la signification de l'émoticône le plus négatif est indiquée : « Je ne suis pas du tout d'accord », et la signification de l'émoticône le plus positif est indiquée : « Je suis tout à fait d'accord ». Dans la figure, un zoom est fait sur le système d'évaluation afin de montrer la correspondance entre chaque émoticône et le score normalisé correspondant au score de croyance compris dans l'ensemble des valeurs $\{0; 0{,}25; 0{,}5; 0{,}75; 1{,}0\}$. Pour les pages des règles de substitution, l'affirmation « La substitution est correcte » est évaluée de la même façon, et pour la relation de subsomption entre une sous-classe `B` et sa classe `A`, l'affirmation « `B` est de la catégorie `A` » est évaluée de la même façon. Sur l'interface de aTaaable, le terme « catégorie » est utilisé car il est plus facilement compréhensible par des utilisateurs qui ne sont pas experts de la représentation des connaissances.

De plus, pour qu'un utilisateur puisse noter un autre utilisateur, le système de notation est intégré à chacune des pages d'utilisateurs où la correspondance entre les émoticônes et les scores normalisés est la même. L'affirmation « L'utilisateur est un expert de la cuisine » est évaluée. En revanche, les scores normalisés correspondent aux scores de confiance a priori.

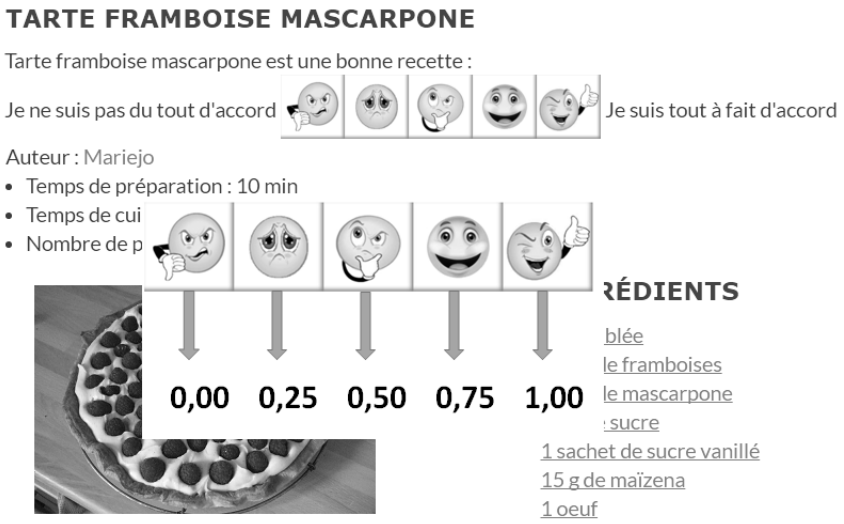


Figure 3.8 – Intégration du système de notation pour la recette « Tarte framboise mascarpone ».

3.3.2 Illustration du calcul de fiabilité dans Taaable

Un utilisateur, Bob, a produit quatre UC qui correspondent aux recettes R_1 et R_2, à la règle de substitution S_1 et à la relation de subsomption L_1. D'autres utilisateurs de la communauté ont évalué les UC produites par Bob ainsi que Bob lui-même grâce au système de notation intégré sur le site aTaaable. Lorsqu'un des utilisateurs clique sur une des 5 émoticônes du système de notation, une action d'évaluation est réalisée. Le tableau 3.1 présente les différentes actions

d'évaluation effectuées par les utilisateurs Anne, Pierre, Émilie, Marie et David à propos des UC produites par Bob et à propos de Bob lui-même.

Le résultat d'une action d'évaluation est un score normalisé entre 0 et 1 qui correspond à un score de croyance si l'évaluation porte sur une UC ou un score de confiance a priori si l'évaluation porte sur un utilisateur de la communauté.

Par exemple, Pierre a évalué L_1 en cliquant sur l'émoticône neutre () qui est associé au score normalisé 0,5 ; ce score correspond à la croyance de Pierre accordée à L_1. Anne, quant à elle, a évalué Bob avec le score (normalisé) 0,75 ; ce score correspond à la confiance a priori de Anne envers Bob.

À partir du tableau 3.1, les scores de croyance envers les UC produites par Bob et les scores de confiance a priori envers Bob sont présentés dans le tableau 3.2. Bob n'a évalué aucune des UC qu'il a produite ; le score de croyance de Bob envers chaque UC est de 1,0.

	Scores de croyance				**Score de confiance a priori**
	R_1	R_2	S_1	L_1	Bob
Anne					
Pierre					
Émilie					
Marie					
David					

Tableau 3.1 – Évaluations sur les UC produites par Bob et évaluations sur Bob.

	Scores de croyance				**Score de confiance a priori**
	R_1	R_2	S_1	L_1	Bob
Anne	0,75	1,0	?	?	0,75
Pierre	?	1,0	1,0	0,5	?
Émilie	?	?	0,75	?	?
Marie	?	?	?	0,25	?
David	?	?	?	?	?
Bob	1,0	1,0	1,0	1,0	×

Tableau 3.2 – Scores de croyance attribués aux UC produites par Bob et scores de confiance a priori attribués à Bob.

Le tableau 3.3 présente les scores de qualité des UC produites par Bob et le tableau 3.4 présente les scores de confiance des utilisateurs de la communauté envers Bob. Les scores de ces deux tableaux sont calculés à partir du tableau 3.2. Alors que la qualité de R_2 est maximale (1,0) car R_2 n'a obtenu que des scores de croyance maximaux (1,0), la qualité de L_1 est basse (0,38) car les scores de croyance attribués à L_1 sont faibles (0,25 et 0,5). La confiance de David envers

	Qualité
R_1	0,75
R_2	1,00
S_1	0,88
L_1	0,38

Tableau 3.3 – Scores de qualité des UC produites par Bob.

	Confiance en Bob
Anne	0,83
Pierre	0,83
Émilie	0,75
Marie	0.25
David	?

Tableau 3.4 – Scores de confiance des utilisateurs envers Bob.

Bob ne peut pas être calculée car David n'a jamais évalué Bob ni les UC que Bob a produites. Les scores de confiance des utilisateurs, excepté David, envers Bob permettent de calculer la réputation de Bob. Avec un seuil $\tau = 3$,

$$\texttt{réputation}(Bob) = \frac{0{,}83 + 0{,}83 + 0{,}75 + 0{,}25}{4} \simeq 0{,}67$$

Les scores de qualité, de confiance et de réputation permettent de calculer un score de fiabilité pour chaque utilisateur à propos de chaque UC produite par Bob. Si un utilisateur a directement évalué une UC, donc qu'un score de croyance existe entre l'utilisateur et l'UC, sa fiabilité envers cette UC sera équivalente. Par exemple, la fiabilité de Anne envers R_1 est $\texttt{fiabilité}(Anne, R_1) = 0{,}75$. Dans le cas où le score de croyance entre l'utilisateur et l'UC n'existe pas, la fiabilité doit être calculée. Selon l'équation (3.6), les poids des différentes métaconnaissances doivent être fixés. Si un utilisateur n'a encore jamais interagi avec l'auteur d'une UC, la confiance n'est pas prise en compte dans le calcul de la fiabilité. Par exemple, comme David n'a jamais noté les UC produites par Bob, alors le poids de la confiance dans le calcul de la fiabilité est nul ; la somme des poids pour la réputation et pour la qualité est de 1.

Le tableau 3.5 présente les scores de fiabilité pour les utilisateurs envers les UC produites par Bob avec $w_{qualité} = 0{,}2$, $w_{confiance} = 0{,}4$ et $w_{réputation} = 0{,}4$. Les poids des scores de fiabilité du tableau 3.5 permettent de privilégier la confiance et la réputation dans le calcul de la fiabilité. Le tableau 3.6 présente les scores de fiabilité pour les utilisateurs envers les UC produites par Bob avec $w_{qualité} = 0{,}6$, $w_{confiance} = 0{,}2$ et $w_{réputation} = 0{,}2$. Les poids des scores de fiabilité du tableau 3.6 permettent de privilégier la prise en compte de la qualité dans le calcul de la fiabilité.

Les deux tableaux font ressortir différentes constatations sur l'impact des poids associés à la qualité, à la confiance et à la réputation dans le calcul de la fiabilité :

- Pour un utilisateur u, la différence des scores de fiabilité entre deux UC produites par un même utilisateur v provient seulement de la différence des scores de qualité de ces deux UC. Par exemple, dans le tableau 3.5, la différence des scores de fiabilité pour Anne entre R_1 (0,75) et L_1 (0,68) produites par Bob provient de la différence des scores de qualité de R_1 (0,75) et L_1 (0,38). Comme dans le tableau 3.6 le poids de la qualité est plus important, la différence du score de fiabilité de R_1 et L_1 pour Anne est plus importante (0,22).
- La différence des scores de fiabilité qu'ont deux utilisateurs u et v envers une même UC uc produite par l'utilisateur w provient de la différence entre le score de confiance de u envers w et le score de confiance de v envers w. Par exemple, dans le tableau 3.5, pour R_1, la différence entre le score de fiabilité de R_1 pour Anne (0,75) et le score de fiabilité de R_1 pour Marie (0,52) égale à 0,23 provient seulement de la différence entre le score de confiance de Anne envers Bob (0,83) et le score de confiance de Marie envers Bob (0,25).

	R_1	R_2	S_1	L_1
Anne	**0,75**	1.00	0,78	**0,68**
Pierre	0,75	1.00	1.00	0,50
Émilie	0,72	0,77	0,75	0,64
Marie	**0,52**	0,57	0,54	0,25
Bob	1,0	1,0	1,0	1,0
David	0,69	0,73	0,71	0,61

TABLEAU 3.5 – Scores de fiabilité des UC produites par Bob pour les différents utilisateurs. Avec $w_{qualité} = 0{,}2$, $w_{confiance} = 0{,}4$ et $w_{réputation} = 0{,}4$ pour tous les utilisateurs sauf David et $w_{qualité} = 0{,}2$, $w_{confiance} = 0{,}0$ et $w_{réputation} = 0{,}8$ pour David.

	R_1	R_2	S_1	L_1
Anne	**0,75**	1,00	0,83	**0,53**
Pierre	0,75	1,00	1,00	0,50
Émilie	0,73	0,88	0,75	0,51
Marie	0,63	0,78	0,71	0,25
Bob	1,0	1,0	1,0	1,0
David	0,72	0,87	0,79	0,49

TABLEAU 3.6 – Scores de fiabilité des UC produites par Bob pour les différents utilisateurs. Avec $w_{qualité} = 0{,}6$, $w_{confiance} = 0{,}2$ et $w_{réputation} = 0{,}2$ pour tous les utilisateurs sauf David et $w_{qualité} = 0{,}6$, $w_{confiance} = 0{,}0$ et $w_{réputation} = 0{,}4$ pour David.

- Plus le poids de la qualité dans le calcul du score de fiabilité est important, moins le score de fiabilité sera personnalisé. Dans le tableau 3.6, la différence du score de fiabilité de R_1 entre Anne et Marie est plus importante que pour le tableau 3.5 car le poids de la qualité est moins important et le poids de la confiance est plus important. Ainsi, dans le tableau 3.6, les scores de fiabilité des utilisateurs pour les UC produites par Bob reflètent moins la confiance de ces utilisateurs envers Bob.
- Si un utilisateur avec une bonne réputation produit une UC de mauvaise qualité, alors moins le poids de la qualité est important dans le calcul du score de fiabilité de uc pour un utilisateur u, meilleure sera le score de fiabilité de uc. Par exemple, Bob, a une bonne réputation malgré la production de L_1 qui est de mauvaise qualité (0,38). Pour Anne, Émilie et David qui n'ont pas évalué L_1, le score de fiabilité de L_1 augmente entre le tableau 3.5 et le tableau 3.6 car le poids de la qualité dans le tableau 3.6 est moins important. De façon analogue, si un utilisateur v avec une mauvaise réputation produit une UC uc de bonne qualité, alors plus le poids de la qualité est important dans le calcul du score de fiabilité de uc pour un utilisateur u, meilleur est le score de fiabilité de uc.

Suite à ces différentes constatations, la détermination des poids doit se faire en fonction de ce qui doit être privilégié dans le calcul du score de fiabilité. Pour la suite, nous avons choisi de privilégier la qualité afin que pour une UC uc ayant une mauvaise qualité, même si son producteur v a une bonne réputation et/ou u a une confiance élevée envers v, la fiabilité de uc pour u soit faible.

Dans eTAAABLE, pour calculer le score de fiabilité que u a envers l'UC uc produite par v, nous posons $w_{qualité} = 0{,}6$, $w_{confiance} = 0{,}2$ et $w_{réputation} = 0{,}2$, si le score de confiance qu'a u envers v n'est pas nul et $w_{qualité} = 0{,}6$, $w_{confiance} = 0{,}0$ et $w_{réputation} = 0{,}4$, si le score de confiance qu'a u envers v est nul.

3.4 Fonction de filtre et fonction de classement

Lorsqu'un système de RÀPC est étendu avec MKM, le module de calcul de fiabilité de MKM permet de calculer un score de la fiabilité qui est utilisé dans une fonction de filtre et une fonction de classement en amont et en aval du système.

3.4.1 La fonction de filtre

La fonction de filtre est utilisée pour sélectionner l'ensemble des UC suffisamment fiables pour l'utilisateur qui interroge le système de RÀPC. Toutes les UC avec un score de fiabilité supérieur à un seuil donné sont sélectionnées pour être disponibles afin d'être utilisées dans le moteur du système de RÀPC, tandis que les UC avec un score inférieur ou égal à ce seuil ne sont pas disponible pour être utilisées par le moteur.

La fonction de filtre utilise des seuils différents pour chaque type d'UC (cas, connaissances du domaine, connaissances d'adaptation et connaissances de similarité). Le seuil pour les cas est noté $\texttt{seuil}_{cas}$, le seuil pour les connaissances du domaine est noté $\texttt{seuil}_{domaine}$, le seuil pour les connaissances d'adaptation est noté $\texttt{seuil}_{adaptation}$ et le seuil pour les connaissances de similarité est noté $\texttt{seuil}_{similarité}$ [26].

3.4.2 La fonction de classement

La fonction de classement est utilisée pour ordonner l'ensemble des réponses retournées par le système de RÀPC grâce à la fiabilité. Le score de fiabilité d'une réponse `Rép` est notée `fiabilité(Rép)` et peut être vue intuitivement comme la probabilité que `Rép` soit satisfaisante pour l'utilisateur qui interroge le système de RÀPC. En cuisine, une réponse satisfaisante pour un utilisateur correspond à la probabilité que l'utilisateur trouve que la recette adaptée est bonne. De façon générale, la probabilité qu'une réponse soit satisfaisante dépend de la probabilité que le cas source remémoré ainsi que son adaptation soient satisfaisants pour l'utilisateur. La probabilité que le cas (respectivement l'adaptation) soit satisfaisant pour l'utilisateur correspond au score de la fiabilité du cas (respectivement de l'adaptation).

Fiabilité de l'adaptation. D'après la section 2.1 du chapitre 3, dans eTaaable, l'adaptation d'un cas remémoré pour répondre à une requête utilisateur consiste à appliquer une ou plusieurs substitutions consistant à remplacer certains ingrédients par d'autres. Par exemple, une adaptation peut consister à remplacer le kiwi par de la poire et la vanille par du jus de citron dans la recette intitulée « Tarte aux kiwis ». Une adaptation peut résulter soit d'une règle de substitution soit d'un processus de généralisation-spécialisation (cf. section 2.4 du chapitre 2). La fiabilité d'une adaptation pour un utilisateur dépend de la fiabilité de chacune des substitutions qui composent l'adaptation. La fiabilité de l'adaptation d'un cas remémoré pour un utilisateur correspond au produit du score de la fiabilité pour cet utilisateur de chacune des substitutions qui la compose. Le score de la fiabilité d'une substitution dépend du processus dont elle résulte. Par exemple, pour répondre à la requête $\texttt{Q}_{ex}$ = `Tarte` ⊓ `Poire` ⊓ `JusDeCitron`, la recette R_{ex1} intitulée « Tarte aux Pommes » et dont l'index est $idx(R_{ex1})$ = `Recette` ⊓ `Tarte` ⊓ `PâteFeuilletée` ⊓ `Pomme` ⊓ `JusDeCitron` ⊓ `Sucre` et la recette R_{ex2} intitulée « Tarte aux kiwis » et dont l'index est $idx(R_{ex2})$ = `Tarte` ⊓ `PâteSablée` ⊓ `Kiwi` ⊓ `Vanille` ⊓ `CrèmePâtissiere` sont remémorées. R_{ex1} est adaptée grâce à l'adaptation A_{ex1} composée d'une substitution `Pomme` ⇝ `Poire` résultant d'une règle de substitution et R_{ex2} est adaptée grâce à l'adaptation A_{ex2} composée des substitutions `Kiwi` ⇝ `Poire` et `Vanille` ⇝ `JusDeCitron`, chacune issue d'un processus de généralisation-spécialisation.

26. Actuellement, dans le système de RÀPC eTaaable, une connaissance de similarité correspond au coût de généralisation entre une sous-classe et sa super-classe calculé par la fonction de coût, et n'est donc pas apportée par un utilisateur. Ainsi, $\texttt{seuil}_{similarité}$ n'est pas utilisé. En revanche, dans une prochaine version de eTaaable, il est envisageable que le coût de généralisation entre une sous-classe et sa classe ne soit plus représenté par un coût calculé automatiquement, mais corresponde à une UC apportée par les utilisateurs.

Le score de fiabilité d'une substitution issue d'une règle de substitution correspond au score de fiabilité de la règle de substitution qui est une UC ; ce score dépend de l'utilisateur (cf. équation (3.6)). Par exemple, soit la règle de substitution dont le contexte est les plats de tarte, l'ingrédient substitué est la pomme et l'ingrédient substituant est la poire. Si cette règle de substitution a une fiabilité de 0,8 pour u, alors `Pomme` $\rightsquigarrow$ `Poire` qui est la substitution permettant d'adapter R_{ex1} a une fiabilité de 0,8 pour u.

Le score de fiabilité d'une substitution issue d'une généralisation-spécialisation de la forme `A` $\rightsquigarrow$ `C`, qui a été obtenue par généralisation de `A` en `B` et par spécialisation de `B` en `C` dépend de plusieurs éléments. Utiliser une substitution trop risquée, c'est-à-dire, ayant un coût élevé, a pour conséquence de diminuer le score de fiabilité de la substitution quel que soit l'utilisateur. De plus, la fiabilité des UC utilisées pour produire la substitution a une conséquence sur la fiabilité de la substitution, et dépend également de l'utilisateur : moins les UC utilisées dans la substitution sont fiables pour l'utilisateur, plus la fiabilité de la substitution est faible pour cet utilisateur. Lors d'une substitution par généralisation-spécialisation, les UC impliquées sont des relations de subsomption. Par conséquent, la fiabilité d'une substitution `A` $\rightsquigarrow$ `C` va dépendre de la fiabilité des relations de subsomption qui ont conduit au chemin de généralisation-spécialisation. Ainsi, pour un utilisateur, la fiabilité de la substitution `A` $\rightsquigarrow$ `C` obtenue par généralisation-spécialisation dépend du risque encouru par la substitution et de la fiabilité du chemin de généralisation-spécialisation pour cet utilisateur. Par exemple, pour u, la fiabilité de la substitution `Kiwi` $\rightsquigarrow$ `Poire` issue de la généralisation de `Kiwi` en `Fruit` et de la spécialisation de `Fruit` en `Poire` dépend du risque de cette substitution et de la fiabilité du chemin de généralisation de `Kiwi` en `Fruit` et de la fiabilité du chemin de spécialisation de `Fruit` en `Poire` pour u. Ainsi, nous devons représenter le risque encouru par la substitution ainsi que la fiabilité du chemin de généralisation-spécialisation.

1. **Risque encourue par la substitution.** Ce risque correspond au coût de la substitution qui est égal au coût de généralisation dont elle est issue (cf. section 2.4.2 du chapitre 2). Plus le coût d'une substitution est grand, plus la fiabilité de la substitution diminue. Dans eTaaable, soit `Q` une requête et `A` $\rightsquigarrow$ `C` une substitution appliquée à la recette remémorée `R` issue d'un processus de généralisation-spécialisation où $\Gamma = \gamma_n \circ \ldots \circ \gamma_2 \circ \gamma_1$ est la fonction de généralisation et $\gamma_1 =$ `A` $\rightsquigarrow$ `B` est la généralisation de la classe `A` de la requête `Q` en une classe `B`. Le coût de la substitution `A` $\rightsquigarrow$ `C` est égal au coût de généralisation de `A` en `B`, c'est-à-dire à `coût`$(\gamma_1 =$ `A` $\rightsquigarrow$ `B`$)$. Par exemple, si la fonction de généralisation permettant de remémorer R_{ex1} pour résoudre $\mathtt{Q}_{ex}$ est $\Gamma =$ `Poire` $\rightsquigarrow$ `Fruit` $\circ$ `JusDeCitron` $\rightsquigarrow$ `Aromatisant`, alors le coût de la substitution `Kiwi` $\rightsquigarrow$ `Poire` est égal à `coût`(`Poire` $\rightsquigarrow$ `Fruit`). Cependant, ce coût de substitution compris dans l'intervalle $[0, +\infty]$ doit être normalisé dans [0,1]. Le coût normalisé d'une substitution σ correspond à `coûtNormalisé`$(\sigma) = e^{-\texttt{coût}(\sigma)}$. Dans l'exemple, si `coût`(`Poire` $\rightsquigarrow$ `Fruit`) = 0,45, alors `coûtNormalisé`(`Kiwi` $\rightsquigarrow$ `Poire`) $= e^{-0,45} \simeq 0,638$. Plus une substitution est coûteuse, plus son coût normalisé tendra vers 0 et moins une substitution est coûteuse, plus son coût normalisé tendra vers 1.
2. **Fiabilité du chemin de généralisation-spécialisation.** Elle dépend des UC impliquées dans ce chemin, qui sont les relations de subsomption entre classes. Utiliser un chemin de généralisation-spécialisation impliquant une ou plusieurs relations de subsomption non fiables pour un utilisateur dans une substitution diminue la fiabilité de la substitution pour cet utilisateur. Pour une substitution, le score de fiabilité du chemin de généralisation-spécialisation pour l'utilisateur u correspond au produit de la fiabilité de chaque UC impliquée dans le chemin. Par exemple, si `Kiwi` $\rightsquigarrow$ `Poire` résulte de la généralisation de

Kiwi en Fruit grâce aux UC Kiwi $\sqsubseteq$ Baie et Baie $\sqsubseteq$ Fruit et de la spécialisation de Fruit en Poire grâce aux UC Poire $\sqsubseteq$ FruitÀPépins et FruitÀPépins $\sqsubseteq$ Fruit, alors la fiabilité de Kiwi $\rightsquigarrow$ Poire dépend de la fiabilité de chacune de ces relations de subsomption.

Soit la fonction Chemins(A,B) qui retourne l'ensemble des chemins de A à B dans le diagramme de Hasse ($\mathcal{H}_{Al}$,r) correspondant à la hiérarchie des aliments ($\mathcal{H}_{Al}$, $\sqsubseteq$).
Dans l'exemple,

$$\texttt{Chemins(Kiwi,Fruit)} = \{(\texttt{Kiwi} \rightarrow \texttt{Baie} \rightarrow \texttt{Fruit})\},$$
$$\texttt{Chemins(Pomme,Fruit)} = \{(\texttt{Pomme} \rightarrow \texttt{FruitÀPépins} \rightarrow \texttt{Fruit})\}.$$

Pour A,B tels que A $\sqsubseteq$ B $\notin$ CD et $\models_{BC}$ A $\sqsubseteq$ B :

$$\texttt{fiabilité}(u,\texttt{A} \sqsubseteq \texttt{B}) = \max\{\texttt{fiabilité\_chemin}(u,c) \mid c \in \texttt{Chemins}(\texttt{A},\texttt{B})\}$$

où fiabilité_chemin(u,c) est la fonction qui retourne le score de fiabilité d'un chemin c pour un utilisateur u et est définie par :

$$\texttt{fiabilité\_chemin}(u,(\texttt{A}_0 \rightarrow \texttt{A}_1 \rightarrow ... \rightarrow \texttt{A}_n)) = \prod_{i=1}^{n} \texttt{fiabilité}(u,\ \texttt{A}_{i-1} \sqsubseteq \texttt{A}_i) \qquad (3.7)$$

avec $\texttt{A}_{i-1} \sqsubseteq \texttt{A}_i \in$ CD et fiabilité(u, $\texttt{A}_{i-1} \sqsubseteq \texttt{A}_i$) défini précédemment dans la section 3.2.7. D'après la figure 3.9 présentant une partie de la hiérarchie des aliments, le score de fiabilité du chemin de généralisation qui relie Kiwi à Fruit est :

$$\begin{aligned}\texttt{fiabilité}(u,\texttt{Kiwi} \sqsubseteq \texttt{Fruit}) &= \texttt{fiabilité}(u,\texttt{Kiwi} \sqsubseteq \texttt{Baie}) \\ &\quad \times \texttt{fiabilité}(u,\texttt{Baie} \sqsubseteq \texttt{Fruit}) \\ &= 0{,}6 \times 0{,}1 = 0{,}54\end{aligned}$$

fiabilité(u,Pomme $\sqsubseteq$ Fruit) = 0,9 est calculé de la même façon que fiabilité(u,Kiwi $\sqsubseteq$ Fruit). Ainsi, le score de fiabilité du chemin de généralisation-spécialisation de Kiwi à Pomme est égale à fiabilité(u,Kiwi $\sqsubseteq$ Fruit)$\times$fiabilité(u,Pomme $\sqsubseteq$ Fruit) = 0,6$\times$0,9 = 0,54.

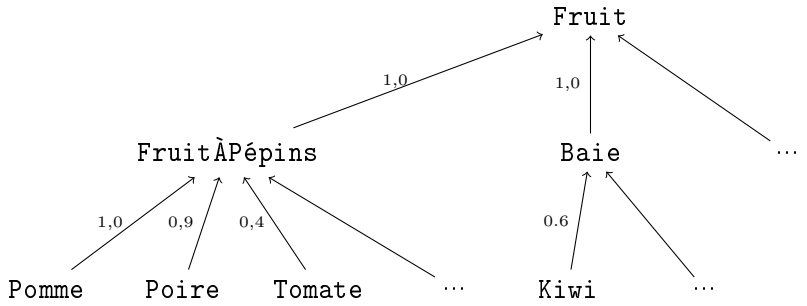


FIGURE 3.9 – Partie de la hiérarchie des aliments de l'ontologie du domaine utilisée par le système eTAAABLE. B $\xrightarrow{x}$ A signifie que B $\sqsubseteq$ A $\in$ CD et fiabilité(B,A) = x.

Le score de fiabilité de la substitution pour un utilisateur u est alors le produit du coût normalisé de la substitution avec le score de fiabilité pour u du chemin de généralisation-spécialisation

produisant la substitution. Ainsi, dans l'exemple, le score de fiabilité de `Kiwi` $\rightsquigarrow$ `Poire` pour l'utilisateur u correspond à $\texttt{fiabilité}(u,\texttt{Kiwi} \rightsquigarrow \texttt{Poire}) = \texttt{coûtNormalisé}(\texttt{Kiwi} \rightsquigarrow \texttt{Poire}) \times \texttt{fiabilité}(u,\texttt{Kiwi} \rightsquigarrow \texttt{Fruit}) \times \texttt{fiabilité}(u,\texttt{Pomme} \rightsquigarrow \texttt{Fruit}) = 0{,}638 \times 0{,}54 \simeq 0{,}345$. $\texttt{fiabilité}(u,\texttt{Vanille} \rightsquigarrow \texttt{JusDeCitron}) = 0{,}7$ est calculée de la même façon que $\texttt{fiabilité}(u,\texttt{Kiwi} \rightsquigarrow \texttt{Poire})$.

Pour un utilisateur donné, le score de fiabilité de l'adaptation d'un cas remémoré correspond au produit des scores de fiabilité de chacune des substitutions qui la composent. Dans l'exemple, pour u, le score de fiabilité de l'adaptation $A_{ex1} = \texttt{Pomme} \rightsquigarrow \texttt{Poire}$ de la recette R_{ex1} correspond à :

$$\begin{aligned}\texttt{fiabilité}(u,A_{ex1}) &= \texttt{fiabilité}(u,\texttt{Pomme} \rightsquigarrow \texttt{Poire}) \\ &= 0{,}8\end{aligned}$$

Pour u, le score de la fiabilité de l'adaptation $A_{ex2} = \texttt{Kiwi} \rightsquigarrow \texttt{Poire} \circ \texttt{Vanille} \rightsquigarrow \texttt{JusDeCitron}$ de la recette R_{ex2} correspond à :

$$\begin{aligned}\texttt{fiabilité}(u,A_{ex2}) &= \texttt{fiabilité}(u,\texttt{Kiwi} \rightsquigarrow \texttt{Poire}) \\ &\quad \times \texttt{fiabilité}(u,\texttt{Vanille} \rightsquigarrow \texttt{JusDeCitron}) \\ &\simeq 0{,}345 \times 0{,}7 \simeq 0{,}242\end{aligned}$$

Fiabilité d'une réponse. La probabilité que le cas remémoré soit satisfaisant et que l'adaptation soit satisfaisante sont considérées comme indépendantes l'une de l'autre. Ainsi, la probabilité qu'une réponse soit satisfaisante correspond au produit du score de fiabilité du cas remémoré par celui de la fiabilité de l'adaptation. Dans l'exemple, pour la réponse $Rép_{ex1}$ qui correspond à la recette R_{ex1} adaptée par A_{ex1}, le score de fiabilité de $Rép_{ex1}$ pour u est le produit du score de fiabilité de R_{ex1} avec le score de fiabilité de A_{ex1} pour u. Si $\texttt{fiabilité}(u,R_{ex1}) = 0{,}9$, alors :

$$\begin{aligned}\texttt{fiabilité}(u,Rép_{ex1}) &= \texttt{fiabilité}(u,R_{ex1}) \times \texttt{fiabilité}(u,A_{ex1}) \\ &= 0{,}9 \times 0{,}8 = 0{,}72\end{aligned}$$

Pour la réponse $Rép_{ex2}$ qui correspond à la recette R_{ex2} adaptée par A_{ex2}, la fiabilité de $Rép_{ex2}$ pour u est le produit de la fiabilité de R_{ex2} par la fiabilité de A_{ex2} pour u. Si $\texttt{fiabilité}(u,R_{ex2}) = 0{,}8$, alors :

$$\begin{aligned}\texttt{fiabilité}(u,Rép_{ex2}) &= \texttt{fiabilité}(u,R_{ex2}) \times \texttt{fiabilité}(u,A_{ex2}) \\ &= 0{,}8 \times 0{,}242 = 0{,}194\end{aligned}$$

Dans le cas particulier où une réponse correspond au cas remémoré sans adaptation, alors le score de fiabilité de la réponse pour l'utilisateur correspond au score de fiabilité du cas pour ce même utilisateur.

3.5 Conclusion

Afin de pouvoir garantir la qualité des réponses retournées par un système de RÀPC utilisant des connaissances provenant d'une communauté en ligne, un modèle, nommé MKM, a été proposé afin de gérer la fiabilité des UC dans le système de RÀPC.

Même si MKM a été construit dans le but de représenter la fiabilité des UC utilisées dans un système de RàPC, MKM est indépendant de l'utilisation prévue des UC. MKM calcule la fiabilité de chaque UC grâce à un module de calcul de la fiabilité. Le score de fiabilité d'une UC uc pour un utilisateur u est calculé à partir de différentes méta-connaissances représentées sous forme de scores : le score de qualité de uc, la confiance que u a envers l'utilisateur v qui a produit uc et la réputation de v. Ces différents scores sont calculés à partir des scores de croyance qu'ont les utilisateurs envers uc et à partir du score de confiance a priori que u a envers v. Les scores de croyance et de confiance a priori proviennent des interactions des utilisateurs avec le système qui sont stockées dans un conteneur de méta-connaissances qui est ajouté. Les scores de croyance proviennent de l'évaluation des utilisateurs envers les UC et les scores de confiance a priori proviennent de l'évaluation des utilisateurs envers les autres utilisateurs.

Plusieurs choix ont été effectués lors de la construction de MKM. Nous avons choisi de ne pas utiliser de confiance par transitivité car l'acquisition et le calcul de scores de recommandations nécessiteraient un modèle beaucoup plus complexe. Cependant, la prise en compte de la confiance par transitivité pourrait être envisagée, notamment lors de la construction d'un système de RàPC personnalisé. De plus, le score de qualité d'une UC uc est calculé grâce à la moyenne arithmétique des scores de croyance attribués à uc et le score de réputation d'un utilisateur v est calculé grâce à la moyenne arithmétique des scores de confiance des utilisateurs de la communauté envers v . Il serait intéressant d'utiliser des opérateurs d'agrégations tels que le minimum, le maximum ou la médiane. Afin de calculer la fiabilité d'une UC uc pour un utilisateur u, il pourrait aussi être envisagé de prendre en compte des scores de recommandation des utilisateurs avec qui u possède un score de confiance.

L'extension d'un système de RàPC utilisant des connaissances provenant d'une communauté en ligne avec MKM se traduit par l'ajout d'une fonction de filtre en amont du moteur de RàPC et par l'ajout d'une fonction de classement en aval du moteur de RàPC. La fonction de filtre permet de filtrer les UC insuffisamment fiables (c'est-à-dire, les UC avec un score de fiabilité inférieur ou égal à un seuil donné). Après l'application de la fonction de filtre, le moteur de RàPC utilise une base de connaissances réduite, ne contenant que les UC suffisamment fiables. La fonction de classement permet au système de RàPC de classer les réponses retournées par le moteur de RàPC, par fiabilité décroissante des réponses. La fiabilité d'une réponse correspond au produit du score de fiabilité du cas remémoré et du score de fiabilité de l'adaptation. La fiabilité d'une adaptation dépend de si l'adaptation résulte d'une règle de substitution ou d'un processus de généralisation-spécialisation. Si l'adaptation résulte d'une règle de substitution, la fiabilité de l'adaptation est égale à la fiabilité de la règle de substitution. Si l'adaptation résulte d'un processus de généralisation-spécialisation, la fiabilité de l'adaptation est égale au produit du coût d'adaptation normalisé entre 0 et 1 et de la fiabilité du chemin de généralisation-spécialisation.

Il serait également intéressant d'envisager la prise en compte de la fiabilité dans le moteur de RàPC. La fiabilité du chemin de généralisation pourrait directement être prise en compte dans le coût de généralisation. En revanche, la fiabilité du chemin de spécialisation ne peut être prise en compte dans un système de RàPC que si la remémoration prend en compte le coût de spécialisation. Une modification du moteur actuel de eTaaable, Tuuurbine, devrait être faite pour prendre en compte la fiabilité du chemin de spécialisation dans eTaaable. En effet, Tuuurbine ne prend pas en compte de coût de spécialisation durant la remémoration. Par exemple, lorsqu'une requête $\mathtt{Q}_{ex} = \mathtt{Tarte} \sqcap \mathtt{Fraise}$ est généralisée en $\Gamma(\mathtt{Q}_{ex}) = \mathtt{Tarte} \sqcap \mathtt{Baie}$, alors seul le coût de la généralisation de `Fraise` en `Baie` est prise en compte. Le coût de spécialisation n'est pas prise en compte puisqu'il n'y a pas de différence de coût pour remémorer une recette

de myrtille qui est une baie ou une recette de raisin qui est une baie. Ainsi, en état, la fiabilité que `Myrtille` est une baie et que `Raisin` est une baie, ne peut pas être prise en compte dans le moteur utilisé de eTaaable.

Le prochain chapitre illustre l'intégration de la fonction de filtre et de classement dans eTaaable. Par ailleurs, une évaluation de l'apport de MKM pour étendre eTaaable est réalisée.

Chapitre 4

Gestion de la fiabilité des connaissances dans eTaaable et évaluation de MKM

Sommaire

La section 4.1 de ce chapitre introduit l'intégration des fonctions de filtre et de classement présentées en section 3.4 du chapitre 3 pour étendre le système de RàPC eTaaable avec MKM à travers un exemple. La section 4.2 présente l'évaluation de l'apport de MKM dans le système eTaaable.

4.1 Exemple d'intégration des fonctions de filtre et de classement dans le système de RàPC eTaaable étendu avec MKM

Cette section illustre une comparaison entre la version standard de eTaaable non étendue par MKM, notée $\text{eTaaable}_{\text{std}}$, et la version de eTaaable étendue par MKM, notée $\text{eTaaable}_{\text{f+c}}$. Soit $\mathtt{Q}_{ex}$ = `Recette` ⊓ `Tarte` ⊓ `Poire` ⊓ `JusDeCitron` une requête soumise par un utilisateur u. Pour cette illustration de fonctionnement, la base de cas de eTaaable est réduite à 8 recettes de tartes présentées dans le tableau 4.1 [27]. Cette table contient les identifiants R_i des recettes, leur index $idx(R_i)$ et leur fiabilité pour u, notée `fiabilité`(u,R_i), calculée grâce à MKM selon l'équation (3.6) de la section 3.2.7 du chapitre 3.

27. Pour alléger le tableau, la classe `Recette` n'est pas spécifiée dans l'index de chaque recette.

R_i	$idx(R_i)$
R_1	TarteSucrée ⊓ PâteBrisée ⊓ Poire ⊓ JusDeCitron ⊓ Banane ⊓ Chocolat
R_2	TarteSalée ⊓ PâteBrisée ⊓ Saumon ⊓ JusDeCitron ⊓ BouillonDePoisson
R_3	TarteSalée ⊓ PâteBrisée ⊓ Tomate ⊓ JusDeCitron ⊓ Mozzarella ⊓ Basilic
R_4	TarteSucrée ⊓ PâteFeuilletée ⊓ Pomme ⊓ JusDeCitron ⊓ Cannelle ⊓ Œuf
R_5	TarteSucrée ⊓ PâteFeuilletée ⊓ Pomme ⊓ JusDeCitron ⊓ Caramel
R_6	TarteSucrée ⊓ PâteSablée ⊓ Kiwi ⊓ JusDeCitron ⊓ Flan ⊓ Rhum
R_7	TarteSucrée ⊓ PâteFeuilletée ⊓ Abricot ⊓ JusDeCitron ⊓ Œuf
R_8	TarteSucrée ⊓ PâteFeuilletée ⊓ Mirabelle ⊓ JusDeCitron ⊓ Sucre

Tableau 4.1 – Base de cas de eTaaable réduite à 8 recettes.

Une partie de la hiérarchie des aliments de aTaaable, liée aux classes utilisées dans cet exemple, est présentée en figure 4.1. Les coûts de généralisation sont représentés sur les arcs entre les classes. Par exemple, la généralisation de `Poire` en `FruitÀPépins` a pour coût de généralisation 0,05, tandis que la généralisation de `Poire` en `Fruit` a pour coût $0{,}05 + 0{,}40 = 0{,}45$.

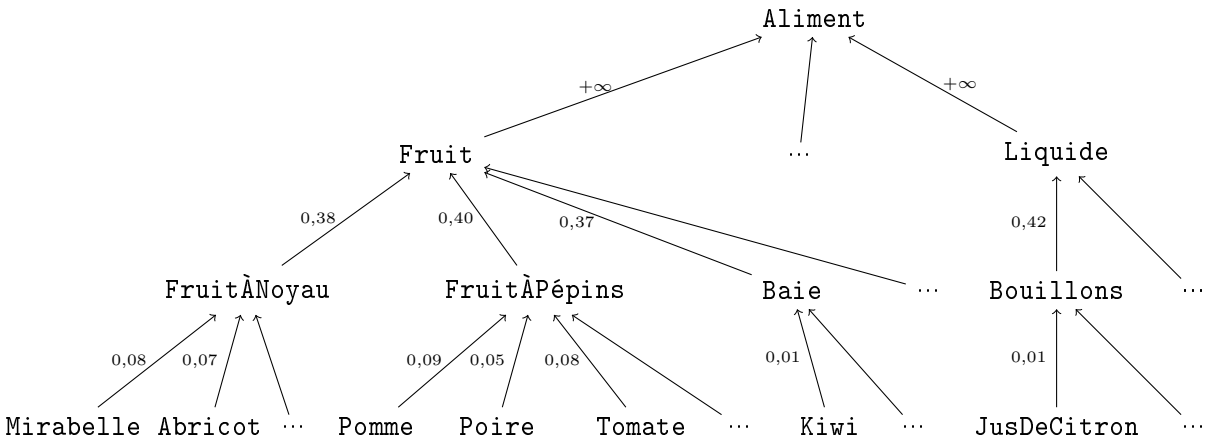


Figure 4.1 – Partie de la hiérarchie des aliments de l'ontologie du domaine utilisée par le système eTaaable.. $\texttt{B} \xrightarrow{x} \texttt{A}$ signifie que $\texttt{B} \sqsubseteq \texttt{A} \in \texttt{CD}$ et $\texttt{coût(B,A)} = x$.

Le tableau 4.2 montre les réponses retournées par $\text{eTaaable}_{\text{std}}$. Nous rappelons que le moteur du système $\text{eTaaable}_{\text{f+c}}$, est identique au moteur de $\text{eTaaable}_{\text{std}}$. Les réponses retournées à la sortie des moteurs de $\text{eTaaable}_{\text{std}}$ et de $\text{eTaaable}_{\text{f+c}}$ sont les mêmes et sont retournées dans le même ordre. Les réponses du tableau 4.2 impliquent les cas présentés dans le tableau 4.1. Le tableau 4.2 contient pour chaque ligne, l'identifiant de la réponse $\texttt{Rép}_i$, l'identifiant de la recette remémorée $\texttt{R}_i$, les identifiants des substitutions $\texttt{S}_j$ impliquées dans l'adaptation si la recette a été adaptée et présentées dans le tableau 4.3, l'ordre dans laquelle la réponse est retournée par le moteur de $\text{eTaaable}_{\text{std}}$, la fonction de généralisation et le numéro d'étape de généralisation lors de la remémoration, et le coût de généralisation lors de la remémoration. Par exemple, $\texttt{Rép}_6$ a été obtenue grâce à la $3^{ème}$ étape de généralisation correspondant à généraliser `Poire` en `Fruit`

$\texttt{Rép}_i$	$\texttt{R}_i$	$\texttt{S}_j$	ordre	Γ_k	$\texttt{coût}(\Gamma_k)$
$\texttt{Rép}_1$	$\texttt{R}_1$	×	1	×	0
$\texttt{Rép}_2$	$\texttt{R}_2$	$\texttt{S}_1$	2	Γ_1 : JusDeCitron $\rightsquigarrow$ Bouillon	0,01
$\texttt{Rép}_3$	$\texttt{R}_3$	$\texttt{S}_2$	2	Γ_1 : Poire $\rightsquigarrow$ FruitÀPépins	0,05
$\texttt{Rép}_4$	$\texttt{R}_4$	$\texttt{S}_3$	3	Γ_2 : Poire $\rightsquigarrow$ FruitÀPépins	0,05
$\texttt{Rép}_5$	$\texttt{R}_5$	$\texttt{S}_3$	3	Γ_2 : Poire $\rightsquigarrow$ FruitÀPépins	0,05
$\texttt{Rép}_6$	$\texttt{R}_6$	$\texttt{S}_4$	4	Γ_3 : Poire $\rightsquigarrow$ Fruit	0,45
$\texttt{Rép}_7$	$\texttt{R}_7$	$\texttt{S}_5$	4	Γ_3 : Poire $\rightsquigarrow$ Fruit	0,45
$\texttt{Rép}_8$	$\texttt{R}_8$	$\texttt{S}_6$	4	Γ_3 : Poire $\rightsquigarrow$ Fruit	0,45

Tableau 4.2 – Ensemble des réponses retournées pour la requête $\texttt{Q}_{ex}$. La colonne ordre correspond à l'ordre dans lesquelles les réponses sont retournées par $\text{eTaaable}_{\text{std}}$. Γ_k correpond à la $k^{ème}$ étape de généralisation tandis que $\texttt{coût}(\Gamma_k)$ correspond au coût de cette généralisation.

avec un coût de 0,45. La généralisation de Poire en Fruit a permis de remémorer entre autres $\texttt{R}_6$ indexée par la classe Kiwi qui est subsumée par Fruit.

$\texttt{S}_j$	ingrédient(s) substitué(s)	ingrédient(s) substituant(s)	$\texttt{coût}(\texttt{S}_j)$
$\texttt{S}_1$	BouillonDePoisson	JusDeCitron	0,01
$\texttt{S}_2$	Tomate	Poire	0,05
$\texttt{S}_3$	Pomme	Poire	0,05
$\texttt{S}_4$	Kiwi	Poire	0,45
$\texttt{S}_5$	Abricot	Poire	0,45
$\texttt{S}_6$	Mirabelle	Poire	0,45

Tableau 4.3 – Les substitutions impliquées dans les réponses présentées dans le tableau 4.2 permettant d'adapter les recettes présentées dans le tableau 4.1 pour satisfaire $\texttt{Q}_{ex}$.

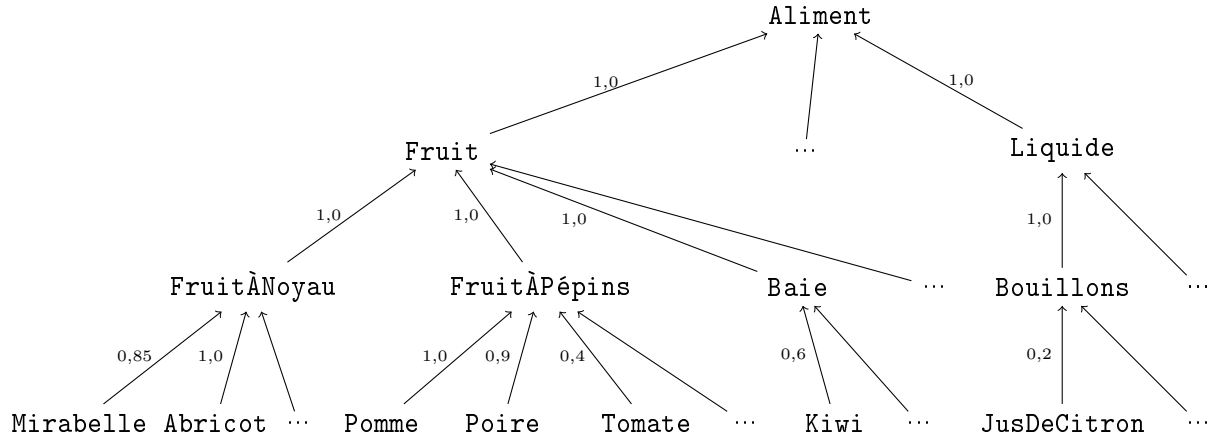


Figure 4.2 – Partie de la hiérarchie des aliments de l'ontologie du domaine utilisée par le système eTaaable. $\texttt{B} \xrightarrow{x} \texttt{A}$ signifie que $\texttt{B} \sqsubseteq \texttt{A} \in \texttt{CD}$ et $\texttt{fiabilité}(\texttt{B},\texttt{A}) = x$.

Le tableau 4.3 présente l'ensemble des substitutions, qui résultent toutes d'un processus de généralisation-spécialisation, impliquées dans les réponses du tableau 4.2 et appliquées aux recettes du tableau 4.1. Ce tableau de substitutions contient pour chaque ligne l'identifiant de la substitution $\mathtt{S}_j$, les ingrédients substitués, les ingrédients substituants, et le coût de la substitution $\mathtt{coût}(\mathtt{S}_j)$. Nous avons vu en section 2.4.2 du chapitre 2 que le coût d'une substitution $\mathtt{A} \rightsquigarrow \mathtt{C}$ issue de la généralisation de A en B et de la spécialisation de B en C appliquée à une recette indexée par A et remémorée grâce à une requête généralisée où C a été généralisée en B, est égal au coût de généralisation de C en A. Ainsi, dans l'exemple, la substitution $\mathtt{S}_4$: $\mathtt{Kiwi} \rightsquigarrow \mathtt{Poire}$ issue de la généralisation de Kiwi en Fruit et de la spécialisation de Fruit en Poire, appliquée à la recette $\mathtt{R}_6$ remémorée grâce à la généralisation de Poire en Fruit, a un coût égal à $\mathtt{coût}(\mathtt{Poire} \rightsquigarrow \mathtt{Fruit}) = 0{,}45$. Pour chaque substitution obtenue par généralisation-spécialisation, la fiabilité du chemin de généralisation-spécialisation qui a permis de générer la substitution est calculée à partir des scores de fiabilité donnés en figure 4.2. Par exemple, pour l'utilisateur u, d'après la figure 4.2 et l'équation (3.7) la fiabilité du chemin de généralisation-spécialisation pour $\mathtt{S}_4 = \mathtt{Kiwi} \rightsquigarrow \mathtt{Poire}$ est :

$$
\begin{aligned}
\mathtt{fiabilité}(u,\mathtt{Kiwi} \sqsubseteq \mathtt{Fruit}) \times \mathtt{fiabilité}(u,\mathtt{Pomme} \sqsubseteq \mathtt{Fruit}) & \\
= \mathtt{fiabilité}(u,\mathtt{Kiwi} \sqsubseteq \mathtt{Baie}) & \\
\times \mathtt{fiabilité}(u,\mathtt{Baie} \sqsubseteq \mathtt{Fruit}) & \\
\times \mathtt{fiabilité}(u,\mathtt{Poire} \sqsubseteq \mathtt{FruitÀPépins}) & \\
\times \mathtt{fiabilité}(u,\mathtt{FruitÀPépins} \sqsubseteq \mathtt{Fruit}) & \\
= 0{,}6 \times 1{,}0 \times 0{,}9 \times 1{,}0 & \\
= 0{,}54 &
\end{aligned}
$$

eTaaable$_{\text{std}}$ retourne les réponses par ordre croissant du coût de la généralisation qui a permis de retourner chaque réponse. Dans le tableau 4.2, l'ordre n de la réponse correspond à la $(n-1)^{ème}$ étape de généralisation. $\mathtt{Rép}_1$ n'utilise pas d'adaptation et donc $\mathtt{Rép}_1$ est la première réponse retournée. $\mathtt{Rép}_1$ est la seule réponse possible qui n'implique pas d'adaptation. La deuxième réponse $\mathtt{Rép}_2$ résulte de la généralisation $\mathit{\Gamma}(\mathtt{Q}_{ex}) = \mathtt{JusDeCitron} \rightsquigarrow \mathtt{Bouillon}$ ayant un coût de 0,01. Les réponses $\mathtt{Rép}_3$, $\mathtt{Rép}_4$ et $\mathtt{Rép}_5$ résultent de la généralisation $\mathit{\Gamma}(\mathtt{Q}_{ex}) = \mathtt{Poire} \rightsquigarrow \mathtt{FruitÀPépins}$ ayant un coût de 0,15. Ainsi $\mathtt{Rép}_3$, $\mathtt{Rép}_4$ et $\mathtt{Rép}_5$ sont retournées dans le troisième ensemble de réponses. Les réponses $\mathtt{Rép}_6$, $\mathtt{Rép}_7$ et $\mathtt{Rép}_8$ résultent de la généralisation $\mathit{\Gamma}(\mathtt{Q}_{ex}) = \mathtt{Poire} \rightsquigarrow \mathtt{Fruit}$ ayant un coût de 0,45 d'après le tableau 4.3. Ainsi $\mathtt{Rép}_6$, $\mathtt{Rép}_7$ et $\mathtt{Rép}_8$ sont retournées dans le quatrième ensemble de réponses.

Les scores de fiabilité des recettes calculés par MKM, pour u, notés $\mathtt{fiabilité}(u,\mathtt{R}_i)$, sont présentés dans le tableau 4.4. Les scores de fiabilité des substitutions qui sont impliquées dans les réponses du tableau 4.2 sont présentés dans le tableau 4.5 et notés $\mathtt{fiabilité}(u,\mathtt{S}_j)$. Pour chaque substitution, le score de fiabilité est calculé à partir du coût normalisé de la substitution (cf. section 3.4.2 du chapitre 3) et de la fiabilité du chemin de généralisation-spécialisation dont la substitution est issue, présentés dans le tableau 4.5. Dans la version de eTaaable$_{\text{f+c}}$, les UC non fiables pour l'utilisateur u qui interroge le système sont filtrées. Dans cet exemple, avec $\mathtt{seuil}_{cas} = 0{,}5$ et $\mathtt{seuil}_{domaine} = 0{,}3$. $\mathtt{R}_1$ et $\mathtt{S}_1$ sont filtrées car $\mathtt{fiabilité}(u,\mathtt{R}_1) = 0{,}25 < \mathtt{seuil}_{cas}$ et $\mathtt{fiabilité}(u,\mathtt{S}_1) = 0{,}20 < \mathtt{seuil}_{adaptation}$. $\mathtt{R}_1$ ne peut donc pas être remémorée et $\mathtt{S}_1$ ne peut pas être utilisée dans une adaptation lorsque u interroge le système. L'UC $\mathtt{Tomate} \sqsubseteq \mathtt{FruitÀPépins}$ est aussi filtrée car d'après la figure 4.2, $\mathtt{fiabilité}(\mathtt{Tomate} \sqsubseteq \mathtt{FruitÀPépins},u) = 0{,}30 \leqslant \mathtt{seuil}_{domaine}$. Ainsi, lorsque u interroge eTaaable$_{\text{f+c}}$ avec la re-

R_i	fiabilité(u,R_i)
R_1	0,25
R_2	0,80
R_3	0,90
R_4	0,85
R_5	0,80
R_6	0,65
R_7	0,90
R_8	0,95

Tableau 4.4 – Fiabilité des recettes présentées dans le tableau 4.2.

S_j	coûtNormalisé(S_j)	fiabilité du chemin	fiabilité(u,S_j)
S_1	filtrée	filtrée	
S_2	filtrée	filtréé	
S_3	0,155	0,90	0,861
S_4	0,155	0,54	0,344
S_5	0,555	0,90	0,574
S_6	0,555	0,76	0,485

Tableau 4.5 – Coût des substitutions présentées dans le tableau 4.2, fiabilité du chemin de généralisation-spécialisation, ainsi que la fiabilité des substitutions elles-mêmes.

$Rép_i$	fiabilité(u,Rép)
$Rép_1$	filtré
$Rép_2$	filtré
$Rép_3$	filtré
$Rép_4$	0,732
$Rép_5$	0,689
$Rép_6$	0,224
$Rép_7$	0,517
$Rép_8$	0,461

Tableau 4.6 – Fiabilité des réponses retournées par eTaaable$_{std}$.

quête Q_{ex}, l'adaptation d'une recette remémorée, dont l'index contient la classe C, ne peut être composée d'une substitution de la forme Tomate $\rightsquigarrow$ C si cette substitution a été obtenue par un chemin de généralisation-spécialisation qui passe par la classe FruitÀPépins. Par conséquent, dans eTaaable$_{f+c}$ interrogé par l'utilisateur u, les réponses $Rép_1$, $Rép_2$ et $Rép_3$ ne peuvent plus être produites car ces réponses impliquent des UC filtrées.

La fonction de classement permet d'ordonner $Rép_4$, $Rép_5$, $Rép_6$, $Rép_7$ et $Rép_8$ par score de fiabilité décroissant, pour l'utilisateur u. Ainsi, d'après le tableau 4.2, eTaaable$_{f+c}$ retourne les réponses dans l'ordre suivant : $Rép_4$, $Rép_5$, $Rép_7$, $Rép_8$ et $Rép_6$.

4.2 Évaluation de l'apport du modèle MKM dans le système de RàPC eTaaable

Cette section évalue l'apport de MKM dans un système de RàPC, le système eTaaable.

4.2.1 Hypothèses

Afin de valider l'apport de MKM dans un système de RàPC, nous posons une hypothèse globale, notée (H) : étendre un système de RàPC utilisant des connaissances provenant d'une

communauté en ligne avec MKM en intégrant d'une part une fonction de filtre et d'une autre part une fonction de classement améliore les réponses du système.

Deux hypothèses liées à (H) sont :

(H1) l'intégration d'une fonction de filtre permettant de filtrer les UC non fiables dans un système de RàPC utilisant des connaissances provenant d'une communauté en ligne améliore les réponses du système.

(H2) l'intégration d'une fonction de classement permettant de classer les réponses par fiabilité dans un système de RàPC utilisant des connaissances provenant d'une communauté en ligne améliore les réponses du système.

4.2.2 Méthode de comparaison

Quatre versions de eTaaable sont comparées :

- eTaaable$_{\text{std}}$: la version standard de eTaaable qui n'utilise pas MKM, et donc, qui ne gère pas la fiabilité des UC ;
- eTaaable$_{\text{f}}$: la version de eTaaable qui implémente la fonction de filtre pour filtrer les UC non fiables ;
- eTaaable$_{\text{c}}$: la version de eTaaable qui implémente la fonction de classement pour classer les réponses retournées par eTaaable par ordre de fiabilité décroissante ;
- eTaaable$_{\text{f+c}}$: la version de eTaaable qui implémente la fonction de filtre et la fonction de classement pour filtrer les UC non fiables et classer les réponses du système par fiabilité.

La comparaison de ces quatre versions de eTaaable doit permettre de valider les différentes hypothèses :

- (H1) sera validée si eTaaable$_{\text{f}}$ retourne significativement de meilleures réponses que eTaaable$_{\text{std}}$, ce qui signifie que la fonction de filtre seule améliore significativement les réponses.
- (H2) sera validée si eTaaable$_{\text{c}}$ retourne significativement de meilleures réponses que eTaaable$_{\text{std}}$, ce qui signifie que la fonction de classement seule améliore significativement les réponses.
- (H) sera validée si :
  - (H1) et (H2) sont validées
  - eTaaable$_{\text{f+c}}$ retourne significativement de meilleures réponses que eTaaable$_{\text{std}}$, ce qui signifie que la fonction de filtre combinée à la fonction de classement améliore significativement les réponses.
  - eTaaable$_{\text{f+c}}$ retourne significativement de meilleures réponses que eTaaable$_{\text{c}}$, ce qui signifie que l'ajout de la fonction de filtre en plus de la fonction de classement améliore significativement les réponses.
  - eTaaable$_{\text{f+c}}$ retourne significativement de meilleures réponses que eTaaable$_{\text{f}}$, ce qui signifie que l'ajout de la fonction de classement en plus de la fonction de filtre améliore significativement les réponses.

  Les deux dernières comparaisons permettent de valider si les deux fonctions sont nécessaires pour améliorer au mieux les réponses ou si une seule fonction est suffisante.

4.2.3 Paramétrage des systèmes

Le calcul des réponses de eTaaable$_{\text{f}}$ et eTaaable$_{\text{f+c}}$ a été réalisé avec $\texttt{seuil}_{cas}$= 0,5 et $\texttt{seuil}_{domaine}$=0,3. $\texttt{seuil}_{adaptation}$ n'a pas été fixé car aucune règle de substitution n'a été utilisée.

Le nombre de réponses calculées par les moteurs de eTaaable$_c$ et eTaaable$_{f+c}$ est fixé à 50, pour pouvoir classer les réponses par fiabilité.

Les scores de fiabilité des UC utilisés par eTaaable$_f$, eTaaable$_c$ et eTaaable$_{f+c}$, ont été calculés pour un utilisateur n'ayant jamais interagi dans le système : la confiance n'a pas été prise en compte dans le calcul du score de fiabilité et ainsi le score de fiabilité n'est pas personnalisé. Ce choix a été effectué pour des raisons pratiques d'évaluation. En effet, en personnalisant le score de fiabilité, les réponses de eTaaable$_f$, eTaaable$_c$ et eTaaable$_{f+c}$ auraient été différentes pour chaque utilisateur. Ainsi, les réponses de eTaaable$_f$, eTaaable$_c$ et eTaaable$_{f+c}$ auraient dû être calculées pour chaque évaluateur. De plus, chaque évaluateur aurait dû être un utilisateur actif du site aTaaable, afin d'avoir noté un nombre suffisant d'UC et de pouvoir obtenir des réponses personnalisées. Toutes ces contraintes nous ont donc amenées à décider de ne pas personnaliser les réponses et de calculer un ensemble de réponses indépendantes de l'évaluateur.

4.2.4 Construction du plan de test pour l'évaluation

id	requête[28]
1	`Tarte ⊓ Ananas`
2	`Tarte ⊓ Banane`
3	`Tarte ⊓ CompoteDePommes`
4	`Tarte ⊓ Datte`
5	`Tarte ⊓ Framboise`
6	`Tarte ⊓ Rhubarbe`
7	`Tarte ⊓ Abricot ⊓ Amande`
8	`Tarte ⊓ Cannelle ⊓ Pomme`
9	`Tarte ⊓ Cerise ⊓ ChocolatBlanc`
10	`Tarte ⊓ Citron ⊓ Mirabelle`
11	`Tarte ⊓ Citron ⊓ NoixDeCoco`
12	`Tarte ⊓ Fraise ⊓ Mascarpone`
13	`Tarte ⊓ FromageBlanc ⊓ Vanille`
14	`Tarte ⊓ Pomme ⊓ Rhubarbe`
15	`Tarte ⊓ Aubergine`
16	`Tarte ⊓ Endive`
17	`Tarte ⊓ Thon`
18	`Tarte ⊓ Asperge ⊓ Parmesan`
19	`Tarte ⊓ Boeuf ⊓ Courgette`
20	`Tarte ⊓ Brocoli ⊓ Lardons`
21	`Tarte ⊓ Carotte ⊓ Oignon`
22	`Tarte ⊓ Épinard ⊓ Saumon`
23	`Tarte ⊓ FromageDeChèvre ⊓ Figue`
24	`Tarte ⊓ FromageDeChèvre ⊓ Poireau`
25	`Tarte ⊓ Mozzarella ⊓ Tomate`

Tableau 4.7 – Ensemble des requêtes de tartes du plan de test.

28. Pour alléger les tableaux, la classe `Recette` n'est pas spécifiée pour chaque requête.

id	requête
26	Cocktail ⊓ Tequila ⊓ SucreVanillé
27	Cocktail ⊓ Tequila ⊓ JusDePommes
28	Cocktail ⊓ Champagne ⊓ SiropDeFruits
29	Cocktail ⊓ Curaçao ⊓ SucreVanillé
30	Cocktail ⊓ JusDeGoyave ⊓ Grenadine
31	Cocktail ⊓ JusDeGoyave ⊓ NoixDeCoco
32	Cocktail ⊓ Martini ⊓ SucreRoux
33	Cocktail ⊓ Vodka ⊓ Perrier
34	Cocktail ⊓ Whisky
35	Cocktail ⊓ Cidre
36	Cocktail ⊓ Gin
37	Cocktail ⊓ JusDeTomates
38	Cocktail ⊓ VinBlanc
39	Cocktail ⊓ JusDeBananes
40	Cocktail ⊓ JusDeCitron ⊓ SelDeCélerie
41	Cocktail ⊓ Bière
42	Cocktail ⊓ Malibu ⊓ JusAbricots
43	Cocktail ⊓ Rhum ⊓ Vodka
44	Cocktail ⊓ CrèmeDeCassis ⊓ Limonade
45	Cocktail ⊓ JusDePommes ⊓ Menthe
46	Cocktail ⊓ LiqueurDeFraises
47	Cocktail ⊓ VinRosé ⊓ SucreDeCanne
48	Cocktail ⊓ Tequila ⊓ Schweppes
49	Cocktail ⊓ Rhum ⊓ SiropDeMenthe
50	Cocktail ⊓ Vodka

Tableau 4.8 – Ensemble des requêtes de cocktails du plan de test.

Le plan de test comprend 50 requêtes : 25 requêtes de tartes et 25 de cocktails. Chaque requête porte sur un type de plat (`Tarte` ou `Cocktail`) et un ou deux ingrédients. Le type de plat permet de restreindre le nombre de cas de la base de cas qui peuvent potentiellement être remémorés.

En effet, afin de ne remémorer que des cas qui sont indexés par le type de plat demandé (`Tarte` ou `Cocktail`), un coût de généralisation très élevé est fixé entre les classes de la hiérarchie des types de plats. La base de cas contient 122 recette de tartes et 112 recettes de cocktails.

Un premier ensemble de requêtes avait été construit en sélectionnant aléatoirement les ingrédients. Nous avons cependant rejeté cette méthode de construction des requêtes car certaines requêtes n'étaient pas réalistes quant à la simulation d'une requête utilisateur, par exemple, `Recette` ⊓ `Tarte` ⊓ `Banane` ⊓ `Mozzarella`. Avec de telles requêtes, des biais auraient été insérés dans les expérimentations. En effet, les scores de satisfaction à propos des réponses obtenues par cette requête pourraient être mauvais à cause de la requête et non pas à cause du système ayant retourné les réponses. Avec la requête donnée en exemple, il est facile d'imaginer que les recettes adaptées retournées en réponse qui contiennent de la banane et de la mozzarella n'obtiendront pas de bons scores de satisfaction quel que soit le système qui en est à l'origine. Ainsi, nous avons finalement construit manuellement l'ensemble des requêtes correspondant à des requêtes utilisateurs plausibles.

Le tableau 4.7 contient l'ensemble des requêtes de tartes et le tableau 4.8 contient l'ensemble des requêtes de cocktails. Une requête de tarte est composée de classes de fruits, de légumes, de viandes, de fromages, d'épices ou d'arômes. Une requête de cocktail est composée de classes d'alcools, de jus de fruits, de sirops, d'autres liquides, de fruits ou d'arômes.

Pour chaque requête et chaque système à comparer, les 5 premières réponses avec adaptation sont évaluées. Chaque réponse représentant une adaptation, qui correspond à une recette et ses substitutions, est évaluée grâce à l'interface d'évaluation présentée en section 2.5 du chapitre 2. Pour chaque réponse, la moyenne des scores de satisfaction obtenus est calculée. Ensuite, pour chaque requête et chaque système à comparer, la moyenne des scores de satisfaction attribués aux 5 premières réponses par les évaluateurs est calculée.

4.2.5 Préparation des données

Dans aTaaable, les UC de l'ontologie du domaine ont principalement été créées par des ingénieurs des connaissances. En effet, le temps de construire une communauté en ligne qui participe activement n'a pas été suffisant pour cette thèse. Ainsi, les connaissances du domaine sont plutôt consensuelles et fiables. De ce fait, des erreurs ont volontairement été introduites dans les connaissances du domaine, correspondant aux relations de subsomption, afin de simuler le comportement d'utilisateurs peu experts (indiquant par exemple que, en cuisine, la tomate est un fruit à pépins) ou malveillants (par exemple, considérant que la noix est une baie) et tester si MKM est capable de gérer de telles erreurs. Nous entendons par relation de subsomption fausse, une relation de subsomption que même un « non expert » du domaine ne pourrait pas énoncer sauf s'il est malveillant et nous entendons par relation de subsomption discutable une relation de subsomption qu'un « non expert » du domaine pourrait énoncer. 43 relations de subsomptions fausses, présentées dans le tableau 4.9, et 42 relations de subsomption discutables, présentées dans le tableau 4.10, ont été ajoutées à l'ensemble des UC. Les deux tableaux présentent le score de fiabilité de chaque relation de subsomption. Le calcul du score de fiabilité a été présenté en section 3.2.7 du chapitre 3.

4.2.6 Analyse brute des réponses retournées par les systèmes

Grâce aux scores de fiabilité et avec $\texttt{seuil}_{domaine} = 0{,}3$, la fonction de filtre permet de filtrer 41 des 44 relations de subsomption fausses. Concernant les relations de subsomption discutables, 22 sur 42 sont filtrées. Les relations de subsomption discutables sont proportionnellement moins filtrées que les relations de subsomption fausses car les scores d'évaluation (c'est-à-dire, les scores de croyance) donnés par les utilisateurs aux relations de subsomption montrent que les utilisateurs ont des avis partagés quant à l'exactitude de ces relations de subsomption. En effet, dans le tableau 4.10, les scores de fiabilité différent entre 0,00 et 0,59 ce qui montre que les utilisateurs ont évalué les relations de subsomption discutables avec des scores de croyance très différents.

Le tableau 4.11 présente, pour les quatre systèmes, le nombre de réponses pour les requêtes de tartes et de cocktails qui utilisent des relations de subsomption fausses. Les 3 relations de subsomption fausses qui ont une fiabilité supérieure à $\texttt{seuil}_{domaine} = 0{,}3$ ne sont utilisées par aucun des quatre systèmes. Le tableau 4.12 présente, pour les quatre systèmes, le nombre de réponses pour les requêtes de tartes et de cocktails qui utilisent des relations de subsomption discutables avec un score de fiabilité inférieur ou égal à $\texttt{seuil}_{domaine} = 0{,}3$. Le tableau 4.13 présente, pour les quatre systèmes, le nombre de réponses pour les requêtes de tartes et de cocktails qui utilisent des relations de subsomption discutables avec un score de fiabilité supérieur à $\texttt{seuil}_{domaine} = 0{,}3$.

Relation de subsomption	Fiabilité
Amande ⊑ Herbes	0,16
Cerise ⊑ Assaisonnement	0,05
Figue ⊑ Condiment	0,16
Framboise ⊑ LégumeInfloresence	0,00
Noix ⊑ Baie	0,05
Pistache ⊑ Céréale	0,16,
Prune ⊑ LégumeRacine	0,05
Rhubarbe ⊑ LégumeBulbe	0,20
Ail ⊑ FruitSec	0,11
Asperge ⊑ Céréale	0,21
Céleri ⊑ FruitExotique	0,00
Épinard ⊑ ProtéineVégérale	0,16
Gingembre ⊑ Baie	0,00
BeurreDeCacahuete ⊑ ProtéineVégérale	0,16
CompoteDePommes ⊑ MatièreGrasse	0,08
SauceTomate ⊑ Confiture	0,00
Mascarpone ⊑ Lait	0,16
Mozzarella ⊑ MatièreGrasse	0,00
VinBlanc ⊑ Sauce	0,16
JusAnanas ⊑ Alcool	0,00
JusDeMangue ⊑ FruitExotique	0,29
JusDeMangue ⊑ The	0,08
JusDePommes ⊑ Liqueur	0,03
JusDeTomates ⊑ Bouillon	0,23
Sirop ⊑ JusDeFruits	0,16
Grenadine ⊑ JusDeFruits	0,16
Coca-cola ⊑ Sirop	0,00
Coca-cola ⊑ Eau	0,05
Limonade ⊑ Assaisonnement	0,24
Limonade ⊑ Jus	0,16
Perrier ⊑ Liqueur	0,00
Perrier ⊑ LiqueurAmère	0,04
Bière ⊑ BoissonGazeuseNonAlcoolisée	0,24
Champagne ⊑ Assaisonnement	0,05
Cidre ⊑ Spiritueux	0,21
CrèmeDeCassis ⊑ SiropDeFruits	0,16
Liqueur ⊑ Assaisonnement	0,16
Liqueur ⊑ JusAbricots	0,20
Malibu ⊑ JusDeFruits	**0,40**
Whisky ⊑ Bière	**0,32**
Rhum ⊑ CrèmeDeLiqueur	0,08
RhumBlanc ⊑ Jus	**0,40**
Spiritueux ⊑ Sirop	0,16
Vodka ⊑ JusDeFruits	0,00

Tableau 4.9 – Scores de fiabilité des relations de subsomption fausses. Les scores de fiabilité en gras sont supérieurs à 0,30.

Relation de subsomption	Fiabilité
Avocat ⊑ FruitÀNoyau	**0,48**
Banane ⊑ FruitÀPépins	0,03
Banane ⊑ FruitSec	0,13
Betterave ⊑ LégumeFruit	**0,37**
Citron ⊑ FruitÀPépins	**0,31**
CompoteDePommes ⊑ Pomme	**0,43**
Concombre ⊑ FruitÀPépins	0,13
Confiture ⊑ Fruit	0,24
Datte ⊑ FruitÀNoyau	0,16
Figue ⊑ FruitÀPépins	**0,52**
Figue ⊑ FruitTropical	0,22
Kiwi ⊑ Baie	**0,58**
Kiwi ⊑ FruitÀPépins	0,16
Litchi ⊑ FruitÀNoyau	**0,52**
Mangue ⊑ FruitÀNoyau	**0,42**
Olive ⊑ FruitÀNoyau	**0,64**
PâteAmande ⊑ Fruit	0,18
Poivron ⊑ FruitÀPépins	0,26
Potiron ⊑ FruitÀPépins	**0,32**
Raisin ⊑ Baie	**0,44**
Raisin ⊑ FruitÀPépins	**0,59**
Tomate ⊑ FruitÀPépins	**0,30**
TomateSéchée ⊑ FruitSec	0,40
Artichaut ⊑ LégumeInfloresence	0,26
Betterave ⊑ Chou	0,21
Céleri ⊑ LégumeTige	**0,43**
Endive ⊑ LégumeInfloresence	0,16
Gingembre ⊑ LégumeRacine	**0,40**
Poireau ⊑ LégumeTige	**0,35**
Rhubarbe ⊑ LégumeBulbe	0,20
Rhubarbe ⊑ LégumeTige	0,16
Caramel ⊑ PâteÀTartiner	0,20
Ketchup ⊑ Parfum	0,05
Pesto ⊑ PâteÀTartiner	**0,36**
Coriandre ⊑ Herbes	**0,58**
NoixDeMuscade ⊑ Herbes	0,13
Paprika ⊑ Herbes	0,29
Poivre ⊑ Herbes	**0,32**
Sel ⊑ Herbes	0,00
JusDeCitrons ⊑ SiropDeFruits	**0,36**
Miel ⊑ SiropChimique	0,16
SiropErable ⊑ SiropChimique	0,24

Tableau 4.10 – Scores de fiabilité des relations de subsomption discutables. Les scores de fiabilité en gras sont supérieurs à 0,30.

	utilisée pour les tartes	utilisée pour les cocktails	Total
eTaaable$_{\text{std}}$	26	42	68
eTaaable$_{\text{f}}$	0	0	0
eTaaable$_{\text{c}}$	11	1	12
eTaaable$_{\text{f+c}}$	0	0	0

Tableau 4.11 – Utilisation des relations de subsomption fausses par chacun des quatre systèmes. La $2^{ème}$ colonne (respectivement la $3^{ème}$ colonne) représente le nombre de réponses utilisant des relations de subsomption fausses pour les tartes (respectivement les cocktails) dans chacun des systèmes.

	utilisée pour les tartes	utilisée pour les cocktails	Total
eTaaable$_{\text{std}}$	3	0	3
eTaaable$_{\text{f}}$	0	0	0
eTaaable$_{\text{c}}$	4	0	4
eTaaable$_{\text{f+c}}$	0	0	0

Tableau 4.12 – Utilisation des relations de subsomption discutables avec un score de fiabilité inférieur ou égal à $\texttt{seuil}_{domaine} = 0{,}3$, par chacun des quatre systèmes. La $2^{ème}$ colonne (respectivement la $3^{ème}$ colonne) représente le nombre de réponses utilisant des relations de subsomption discutables avec un score de fiabilité inférieur ou égal à $\texttt{seuil}_{domaine} = 0{,}3$ pour les tartes (respectivement les cocktails) dans chacun des systèmes.

	utilisée pour les tartes	utilisée pour les cocktails	Total
eTaaable$_{\text{std}}$	31	0	31
eTaaable$_{\text{f}}$	27	0	27
eTaaable$_{\text{c}}$	17	0	17
eTaaable$_{\text{f+c}}$	18	0	18

Tableau 4.13 – Utilisation des relations de subsomption discutables avec un score de fiabilité supérieur à $\texttt{seuil}_{domaine} = 0{,}3$, par chacun des quatre systèmes. La $2^{ème}$ colonne (respectivement la $3^{ème}$ colonne) représente le nombre de réponses utilisant des relations de subsomptions discutables avec un score de fiabilité supérieur à $\texttt{seuil}_{domaine} = 0{,}3$ pour les tartes (respectivement les cocktails) dans chacun des systèmes.

Le tableau 4.14, respectivement le tableau 4.15, donne le nombre et la proportion de réponses communes retournées par les quatre systèmes pour les requêtes de tartes, respectivement les requêtes de cocktails. Le tableau 4.16 donne le nombre et la proportion de réponses communes totales retournées par les quatre systèmes. Chaque système a retourné 125 réponses pour les requêtes de tartes, avec 5 réponses pour chacune des 25 requêtes, et 125 réponses pour les requêtes de cocktails, avec 5 réponses pour chaque des 25 requêtes. Les quatre systèmes ont retourné au total 557 réponses différentes.

La différence des réponses entre eTaaable$_{\text{f}}$ et eTaaable$_{\text{f+c}}$ provient de la fonction de classement par fiabilité qui modifie considérablement l'ordre des réponses retournées. En effet, des réponses qui ne sont pas dans les 5 premières réponses retournées par eTaaable$_{\text{f}}$ apparaissent dans les 5 premières réponses de eTaaable$_{\text{f+c}}$, et inversement.

Les réponses de eTaaable$_c$ et eTaaable$_{f+c}$ sont très proches notamment pour les cocktails où seule une réponse diffère sur les 125. Ces résultats indiquent que les réponses impliquant des UC non fiables ne sont majoritairement pas classées dans les 5 premières réponses : 16 sur 250 réponses seulement de eTaaable$_c$ impliquent des UC non fiables tandis que 71 réponses de eTaaable$_{std}$ impliquent des UC non fiables.

	eTaaable$_{std}$	eTaaable$_f$	eTaaable$_c$	eTaaable$_{f+c}$
eTaaable$_{std}$	125 (100%)	86 (69%)	48 (38%)	38 (30%)
eTaaable$_f$	×	125 (100%)	50 (40%)	51 (41%)
eTaaable$_c$	×	×	125 (100%)	89 (71%)
eTaaable$_{f+c}$	×	×	×	125 (100%)

Tableau 4.14 – Réponses communes entre les quatre systèmes pour les requêtes de tartes, où chaque système a retourné 125 réponses.

	eTaaable$_{std}$	eTaaable$_f$	eTaaable$_c$	eTaaable$_{f+c}$
eTaaable$_{std}$	125 (100%)	72 (58%)	13 (10%)	13 (10%)
eTaaable$_f$	×	125 (100%)	16 (13%)	16 (13%)
eTaaable$_c$	×	×	125 (100%)	124 (99%)
eTaaable$_{f+c}$	×	×	×	125 (100%)

Tableau 4.15 – Réponses communes entre les quatre systèmes pour les requêtes de cocktails, où chaque système a retourné 125 réponses.

	eTaaable$_{std}$	eTaaable$_f$	eTaaable$_c$	eTaaable$_{f+c}$
eTaaable$_{std}$	250 (100%)	158 (63%)	61 (24%)	51 (20%)
eTaaable$_f$	×	250 (100%)	66 (27%)	67 (27%)
eTaaable$_c$	×	×	250 (100%)	213 (85%)
eTaaable$_{f+c}$	×	×	×	250 (100%)

Tableau 4.16 – Réponses communes entre les quatre systèmes pour les requêtes de tartes et de cocktails, où chaque système a retourné 250 réponses.

4.2.7 Analyse des résultats de l'expérimentation

Le tableau 4.17 présente la moyenne des scores de satisfaction des 5 premières réponses retournées pour les requêtes de tartes et de cocktails par les quatre systèmes. Pour l'ensemble des requêtes, nous observons que eTaaable$_{f+c}$ satisfait le mieux les utilisateurs avec une moyenne de 0,65. Ce tableau montre que la fonction de filtre améliore les réponses car les réponses retournées par eTaaable$_f$ sont plus satisfaisantes que les réponses retournées par eTaaable$_{std}$. De la même manière, la fonction de classement améliore les réponses car les réponses retournées par eTaaable$_c$ sont plus satisfaisantes que les réponses retournées par eTaaable$_{std}$.

De plus, l'utilisation conjointe des fonctions de filtre et de classement dans eTaaable$_{f+c}$ améliore encore plus la satisfaction des réponses car les réponses retournées par eTaaable$_{f+c}$ sont meilleures que les réponses retournées par eTaaable$_f$ et eTaaable$_c$.

Système	Requêtes de tartes	Requêtes de cocktails	Total
$\text{eTaaable}_{\text{std}}$	0,16	0,21	0,19
$\text{eTaaable}_{\text{f}}$	0,38	0,42	0,40
$\text{eTaaable}_{\text{c}}$	0,56	0,52	0,54
$\text{eTaaable}_{\text{f+c}}$	0,77	0,52	0,65

Tableau 4.17 – Moyenne des scores de satisfaction pour les réponses retournées pour les requêtes de tartes et de cocktails, pour les quatre systèmes.

Système	Requêtes de tartes	Requêtes de cocktails	Total
$\text{eTaaable}_{\text{std}}$	0,10	0,33	0,21
$\text{eTaaable}_{\text{f}}$	0,34	0,48	0,48
$\text{eTaaable}_{\text{c}}$	0,52	0,42	0,50
$\text{eTaaable}_{\text{f+c}}$	0,62	0,42	0,60

Tableau 4.18 – Médiane des scores de satisfaction pour les réponses retournées pour les requêtes de tartes et de cocktails, pour les quatre systèmes.

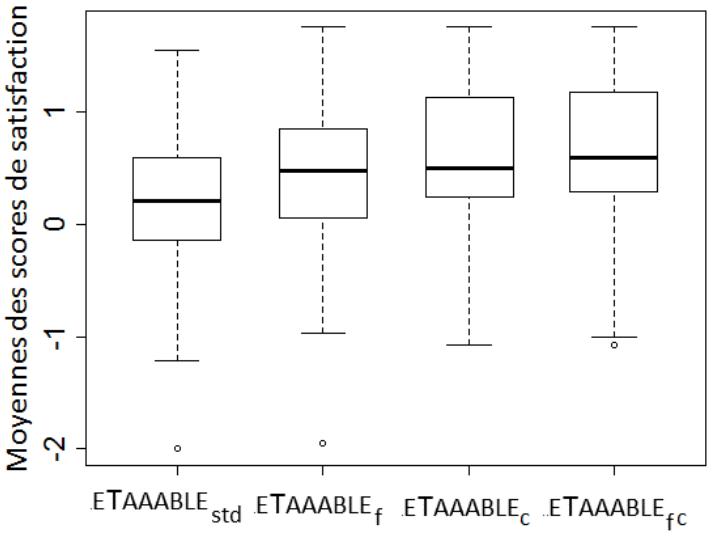


Figure 4.3 – Diagrammes en boîte des moyennes des scores de satisfaction obtenus par les 5 premières réponses retournées par les quatre systèmes pour l'ensemble des requêtes. Les ronds correspondent aux valeurs extrêmes.

Le tableau 4.18 présente la médiane des moyennes des scores de satisfaction des 5 premières réponses retournées pour les requêtes de tartes et de cocktails par les quatre systèmes. La meilleure médiane, équivalente à 0,60, est obtenue par $\text{eTaaable}_{\text{f+c}}$ tandis que la médiane la plus faible, équivalente à 0,21, est obtenue par $\text{eTaaable}_{\text{std}}$. La figure 4.3 montre les diagrammes en boîte des moyennes des scores de satisfaction attribués aux réponses des quatre systèmes, pour l'ensemble des requêtes. En outre des différences de médianes, nous pouvons observer que les moyennes des scores de satisfaction attribués aux réponses de $\text{eTaaable}_{\text{c}}$ et $\text{eTaaable}_{\text{f+c}}$ sont les moins dispersées allant de -1 à 1,76. 50% des moyennes des scores de

satisfaction pour eTaaable$_{f+c}$ sont au-dessus de 0,60 (la médiane) et 25% (le premier quartile) sont au-dessus de 1, tandis que pour eTaaable$_{std}$ moins de 25% des moyennes des scores de satisfaction sont au-dessus de 0,60 et moins de 10% sont au-dessus de 1.

id	Requête	eT_{std}	eT_f	eT_c	eT_{f+c}	Δ	$\uparrow \oplus \pm ?$
1	`Ananas`	1,56	1,76	1,76	1,76	0,20	$\uparrow$
2	`Banane`	−0,77	1,23	0,60	1,23	2,00	$\pm \uparrow \oplus$
3	`CompoteDePommes`	0,10	0,10	0,45	0,45	0,35	$\uparrow$
4	`Datte`	−0,75	0,34	0,15	0,15	0,90	$\pm \uparrow$
5	`Framboise`	−0,05	0,25	1,46	1,46	1,51	$\pm \uparrow \oplus$
6	`Rhubarbe`	−2,00	−1,95	−0,40	−0,13	1,87	$\uparrow \oplus$
7	`Abricot ⊓ Amande`	0,05	0,05	0,89	0,95	0,90	$\uparrow \oplus$
8	`Cannelle ⊓ Pomme`	−0,14	0,06	0,29	0,29	0,43	$\pm \uparrow$
9	`Cerise ⊓ ChocolatBlanc`	0,04	−0,09	−0,08	0,53	0,48	$\uparrow$
10	`Citron ⊓ Mirabelle`	−0,33	−0,95	0,26	0,26	0,59	$\pm \uparrow$
11	`Citron ⊓ NoixDeCoco`	−0,53	−0,57	0,62	0,62	1,15	$\pm \uparrow \oplus$
12	`Pomme ⊓ Rhubarbe`	0,93	0,87	1,17	1,26	0,33	$\uparrow$
13	`Fraise ⊓ Mascarpone`	0,18	0,69	−0,6	0,47	0,29	$\uparrow$
14	`FromageBlanc ⊓ Vanille`	−0,55	−0,40	0,62	0,62	1,17	$\pm \uparrow \oplus$
15	`Aubergine`	0,80	0,73	0,52	0,52	−0,28	
16	`Endive`	0,37	0,32	0,23	0,23	−0,14	
17	`Thon`	1,29	1,33	1,31	1,31	0,02	$\uparrow$
18	`Asperge ⊓ Parmesan`	0,87	0,29	0,69	0,69	−0,18	
19	`Boeuf ⊓ Courgette`	0,49	0,49	0,55	0,55	0,06	$\uparrow$
20	`Brocoli ⊓ Lardons`	1,40	1,40	1,51	1,51	0,11	$\uparrow$
21	`Carotte ⊓ Oignon`	0,64	0,98	0,96	0,96	0,32	$\uparrow$
22	`Épinard ⊓ Saumon`	0,46	0,55	0,30	0,98	0,52	$\uparrow$
23	`FromageDeChèvre ⊓ Figue`	0,10	−0,52	−0,20	0,42	0,32	$\uparrow$
24	`FromageDeChèvre ⊓ Poireau`	1,01	1,01	0,48	0,48	−0,53	
25	`Mozzarella ⊓ Tomate`	−1,11	1,50	−0,80	1,40	2,51	$\pm \uparrow \oplus$
		0,16	0,38	0,56	0,77	0,60	

Tableau 4.19 – Moyennes des scores de satisfaction des réponses de eT_{std} = eTaaable$_{std}$, eT_f = eTaaable$_f$, eT_c = eTaaable$_c$ et eT_{f+c} = eTaaable$_{f+c}$ pour les requêtes de tartes. La colonne Δ correspond à la différence des moyennes entre eTaaable$_{f+c}$ et eTaaable$_{std}$, la dernière colonne décrit cette différence avec des marqueurs : $\uparrow$ marque une différence positive, $\pm$ marque le passage d'une moyenne négative à une moyenne positive et $\oplus$ marque une différence de plus de 1 point. Si aucun marqueur n'est présent sur une ligne, alors la moyenne des scores de satisfaction de eTaaable$_{std}$ est meilleure que la moyenne des scores de satisfaction de eTaaable$_{f+c}$.

Le tableau 4.19 donne le détail des résultats de l'évaluation pour chaque requête de tartes [29] pour chacun des quatre systèmes ; plusieurs constatations peuvent être faites :

– Par rapport à eTaaable$_{std}$, les réponses de eTaaable$_{f+c}$ satisfont mieux les utilisateurs pour 21 requêtes sur 25.

29. Pour alléger les tableaux, la classe `Recette` et la classe correspondant au type de plat ne sont pas spécifiées pour chaque requête.

- Parmi les réponses qui sont plus satisfaisantes pour eTaaable$_{f+c}$, la moyenne des scores de satisfaction devient, avec eTaaable$_{f+c}$, positive pour 8 requêtes en étant négative avec eTaaable$_{std}$, ce qui signifie que l'on passe de réponses « plutôt insatisfaisantes » à des réponses « plutôt satisfaisantes », et la différence avec la moyenne des scores de satisfaction obtenus par eTaaable$_{std}$ est de plus de 1 pour 7 requêtes, ce qui augmente grandement la satisfaction. La raison expliquant cette différence pour ces 7 requêtes vient du fait que les réponses de eTaaable$_{std}$ impliquent des relations de subsomption fausses ou discutables.
- Pour les 6 requêtes dont la moyenne des scores de satisfactions décroît entre eTaaable$_{std}$ et eTaaable$_{f+c}$, la différence est plutôt faible (inférieure à 0,3), exceptée pour la requête 24. Pour 5 de ces 6 requêtes, eTaaable$_{std}$ n'a pas utilisé de relations de subsomption fausses ou discutables dans les 5 premières réponses retournées. Pour la requête 24 : `FromageDeChèvre` $\sqcap$ `Poireau`, les réponses de eTaaable$_{std}$ sont des réponses où les adaptations remplacent un fromage par du fromage de chèvre ce qui satisfait mieux les évaluateurs que les réponses de eTaaable$_{f+c}$ qui sont des réponses où les adaptations remplacent de l'oignon par du poireau.
- Par comparaison avec eTaaable$_{std}$, eTaaable$_{f}$ améliore la satisfaction des réponses pour 12 requêtes sur 25. Les moyennes des scores de satisfaction sont les mêmes pour 5 requêtes car les réponses sont les mêmes pour les deux systèmes. Les réponses sont plus satisfaisantes avec eTaaable$_{f+c}$ qu'avec eTaaable$_{f}$ pour 4 de ces 5 requêtes.
- Par comparaison avec eTaaable$_{std}$, eTaaable$_{c}$ améliore la satisfaction des réponses pour 17 requêtes sur 25. Cependant, les réponses de eTaaable$_{f+c}$ sont toujours plus satisfaisantes que celles de eTaaable$_{c}$.

Le tableau 4.20 donne le détail des résultats de l'évaluation pour chaque requête sur les cocktails pour chacun des quatre systèmes ; plusieurs constatations peuvent être faites :

- Par rapport à eTaaable$_{std}$, les réponses de eTaaable$_{f+c}$ satisfont mieux les utilisateurs pour 16 requêtes sur 25.
- Parmi les réponses qui sont plus satisfaisantes pour eTaaable$_{f+c}$, la moyenne des scores de satisfaction devient, avec eTaaable$_{f+c}$, positive pour 1 requête en étant négative avec eTaaable$_{std}$ et la différence avec la moyenne des scores de satisfaction obtenus par eTaaable$_{std}$ est de plus de 1 pour 4 requêtes.
- Pour les 9 requêtes dont la moyenne des scores de satisfaction décroît entre eTaaable$_{std}$ et eTaaable$_{f+c}$, la différence est plutôt faible (inférieure à 0,3), exceptée pour les requêtes 33 et 49. Pour 6 de ces 9 requêtes, eTaaable$_{std}$ n'a pas utilisé de relations de subsomption fausses ou discutables dans les 5 premières réponses retournées. Pour la requête 33 : `Bière`, les réponses de eTaaable$_{std}$ sont des réponses dans lesquelles les adaptations remplacent une boisson gazeuse non alcoolisée par de la bière ce qui satisfait mieux les évaluateurs que les réponses de eTaaable$_{f+c}$ qui sont des réponses dans lesquelles les adaptations remplacent un alcool par de la bière. Ainsi, même si `Bière` $\sqsubseteq$ `BoissonGazeuseNonAlcoolisée` n'est pas fiable, l'utilisation de cette relation de subsomption dans les réponses améliore les réponses. Nous expliquons cette constatation par le fait que la bière s'accommode mal avec les autres alcools dans les cocktails. Pour la requête 49 : `SucreDeCanne` $\sqcap$ `VinRosé`, eTaaable$_{f+c}$ obtient des réponses moins satisfaisantes car une réponse composée d'une recette très fiable et d'une adaptation avec un chemin de généralisation-spécialisation très fiable a compensé le fait que le coût de généralisation était trop élevé. On rappelle que la fiabilité d'une réponse dépend de la fiabilité du cas remémoré et de la fiabilité de l'adaptation. Lorsque l'adaptation provient d'un processus de généralisation-spécialisation, la fiabilité de l'adaptation dépend de la fiabilité du chemin de généralisation-spécialisation

ainsi que du coût de généralisation de la requête à l'origine de la réponse (cf. section 3.4.2 du chapitre 3).

- Par comparaison avec eTaaable$_{\text{std}}$, eTaaable$_{\text{f}}$ améliore la satisfaction des réponses pour 14 requêtes sur 25. Les moyennes des scores de satisfaction sont les mêmes pour 4 requêtes car les réponses sont les mêmes pour les deux systèmes. Les réponses sont plus satisfaisantes avec eTaaable$_{\text{f+c}}$ qu'avec eTaaable$_{\text{f}}$ pour 3 de ces 4 requêtes.
- Comme les réponses de eTaaable$_{\text{c}}$ ne différent que pour une seule réponse d'une seule requête (38), les remarques pour eTaaable$_{\text{c}}$ sont les mêmes que pour eTaaable$_{\text{f+c}}$, sauf pour la requête 38. L'unique réponse de eTaaable$_{\text{c}}$ qui diffère de eTaaable$_{\text{f+c}}$, pour les 125 réponses, implique une UC non fiable, ce qui impacte la moyenne des scores de satisfaction de la requête 38.

id	Requête	eT_{std}	eT_{f}	eT_{c}	$\text{eT}_{\text{f+c}}$	Δ	$\uparrow \oplus \underline{\pm} ?$
26	JusDeBananes	0,41	0,93	1,20	1,20	0,79	$\uparrow$
27	JusDeTomates	−1,21	−0,97	−1,00	−1,00	0,21	$\uparrow$
28	Grenadine ⊓ JusDeGoyave	0,33	0,99	1,57	1,57	1,24	$\uparrow \oplus$
29	JusAbricots ⊓ Malibu	0,45	0,45	1,18	1,18	0,73	$\uparrow$
30	JusAnanas ⊓ Menthe	0,11	0,31	0,39	0,39	0,28	$\uparrow$
31	JusDeCitron ⊓ Tabasco	−0,88	−0,32	−0,55	−0,55	0,33	$\uparrow$
32	JusDeMangue ⊓ NoixDeCoco	0,60	0,40	0,80	0,80	0,20	$\uparrow$
33	Bière	−0,33	−0,85	−1,07	−1,07	−0,74	
34	Cidre	1,08	1,16	0,93	0,93	−0,15	
35	Gin	0,11	0,08	0,03	0,03	−0,08	
36	VinBlanc	0,19	0,19	1,33	1,33	1,14	$\uparrow \oplus$
37	Vodka	0,12	0,70	1,27	1,27	1,15	$\uparrow \oplus$
38	Whisky	−0,21	−0,20	−0,37	−0,23	−0,02	
39	Champagne ⊓ SiropDeFruits	0,94	0,97	1,63	1,63	0,69	$\uparrow$
40	CrèmeDeCassis ⊓ Limonade	0,08	0,02	0,70	0,70	0,62	$\uparrow$
41	Curaçao ⊓ SiropDeCerise	0,51	0,47	0,25	0,25	−0,26	
42	JusDePommes ⊓ Tequila	0,50	0,47	0,30	0,30	−0,20	
43	LiqueurDeFramboises	0,49	0,49	0,33	0,33	−0,16	
44	Martini ⊓ SucreRoux	0,50	0,50	0,65	0,65	0,15	$\uparrow$
45	Perrier ⊓ Vodka	0,09	0,50	0,36	0,36	0,27	$\uparrow$
46	Rhum ⊓ SiropDeMenthe	0,62	0,73	1,13	1,13	0,51	$\uparrow$
47	Rhum ⊓ Vodka	−0,65	−0,03	0,97	0,97	1,62	$\underline{\pm} \uparrow \oplus$
48	Schweppes ⊓ Tequila	0,22	0,48	0,42	0,42	0,20	$\uparrow$
49	SucreDeCanne ⊓ VinRosé	0,52	0,62	0,22	0,22	−0,30	
50	SucreVanillé ⊓ Tequila	0,75	0,62	0.25	0,25	−0,28	
		0,21	0,42	0,52	0,52	0,31	

Tableau 4.20 – Moyennes des scores de satisfaction des réponses de eT_{std} = eTaaable$_{\text{std}}$, eT_{f} = eTaaable$_{\text{f}}$, eT_{c} = eTaaable$_{\text{c}}$ et $\text{eT}_{\text{f+c}}$ = eTaaable$_{\text{f+c}}$ pour les requêtes sur les recettes de cocktails. La colonne Δ correspond à la différence des moyennes entre eTaaable$_{\text{f+c}}$ et eTaaable$_{\text{std}}$, la dernière colonne décrit cette différence avec des marqueurs : $\uparrow$ marque une différence positive, $\underline{\pm}$ marque le passage d'une moyenne négative à une moyenne positive et $\oplus$ marque une différence de plus de 1 point. Si aucun marqueur n'est présent sur une ligne, alors la moyenne des scores de satisfaction de eTaaable$_{\text{std}}$ est meilleure que la moyenne des scores de satisfaction de eTaaable$_{\text{f+c}}$.

4.2.8 Validation des résultats

La première hypothèse (H1) supposant que filtrer les UC non fiables dans un système de RàPC utilisant des connaissances produites par une communauté en ligne améliore le système est validée dans eTaaable grâce au test des rangs signés de Wilcoxson. Avec une p-valeur de $2{,}78.10^{-3}$ (inférieure à 0,05, ce qui indique que l'hypothèse nulle est rejetée), la différence entre les moyennes des scores de satisfaction des réponses de $\text{eTaaable}_{\text{f}}$ et $\text{eTaaable}_{\text{std}}$ est significative.

La deuxième hypothèse (H2) supposant que classer les réponses par fiabilité d'un système de RàPC utilisant des connaissances produites par une communauté en ligne améliore le système est aussi validée dans eTaaable. Avec une p-valeur de $8{,}52.10^{-4}$ (inférieure à 0,05), la différence entre les moyennes des scores de satisfaction des réponses de $\text{eTaaable}_{\text{c}}$ et $\text{eTaaable}_{\text{std}}$ est significative.

Nous pouvons conclure que les scores de satisfaction pour les réponses de $\text{eTaaable}_{\text{f+c}}$ sont significativement meilleurs que les scores de satisfaction pour les réponses de $\text{eTaaable}_{\text{std}}$. En effet, avec une p-valeur de $5{,}98.10^{-5}$ (inférieure à 0,05), la différence entre les moyennes des scores de satisfaction des réponses de $\text{eTaaable}_{\text{f+c}}$ et $\text{eTaaable}_{\text{std}}$ est significative, avec une p-valeur de $1{,}05.10^{-3}$ (inférieure à 0,05), la différence entre les moyennes des scores de satisfaction des réponses de $\text{eTaaable}_{\text{f+c}}$ et $\text{eTaaable}_{\text{f}}$ est significative et avec une p-valeur de $1{,}16.10^{-2}$ (inférieure à 0,05), la différence entre les moyennes des scores de satisfaction des réponses de $\text{eTaaable}_{\text{f+c}}$ et $\text{eTaaable}_{\text{c}}$ est significative. Ainsi, l'hypothèse globale (H) supposant qu'étendre un système de RàPC utilisant des connaissances produites par une communauté en ligne avec les fonctions de filtre et les fonctions de classement améliore les réponses du système, est validée dans eTaaable.

4.3 Conclusion

Nous avons montré expérimentalement dans ce chapitre que l'extension du système de RàPC eTaaable utilisant des connaissances produites par une communauté en ligne avec MKM en ajoutant une fonction de filtre et une fonction de classement améliore les réponses de façon significative. Par ailleurs, nous avons montré que dans eTaaable étendu avec MKM, d'une part, la fonction de filtre seule améliore les réponses de façon significative, et que d'autre part, la fonction de classement seule améliore les réponses de façon significative.

Dans ces évaluations, nous avons fixés à la main les poids du score de qualité, du score de confiance et du score de réputation pour calculer le score de fiabilité, ainsi que les différents seuils utilisés par la fonction de filtre. D'autres évaluations pourraient être réalisées afin de fixer expérimentalement les poids et les seuils les plus optimaux.

Il serait envisageable de valider l'apport de MKM et des fonctions de filtre et de classement dans d'autres systèmes de RàPC — que le système eTaaable — qui utilisent des connaissances produites par une communauté en ligne. De plus, l'apport de MKM pourrait aussi être évalué dans d'autres systèmes — que les systèmes de RàPC — à base de connaissances qui utilise des connaissances produites par une communauté en ligne, pour valider l'aspect générique de MKM.

Chapitre 5

Gestion de la typicalité des connaissances

Sommaire

Nous avons vu dans les chapitres 3 et 4 que les UC apportées par une communauté en ligne pouvaient être non fiables. Ainsi, pour garantir la qualité des réponses retournées par un système de RÀPC qui utilise des UC provenant d'une communauté en ligne, la fiabilité des UC devait être gérée.

Ce chapitre montre qu'acquérir des UC provenant d'une communauté en ligne a d'autres conséquences. En effet, la collecte de différents avis à propos d'une même UC est facilité grâce à l'utilisation d'une communauté en ligne. Nous nous intéressons dans ce chapitre aux UC qui sont des relations de subsomptions. L'acquisition des différents avis à propos d'une relation de subsomption permet de représenter la gradualité des classes de l'ontologie. Dans notre travail, représenter la gradualité d'une classe revient à discriminer les sous-classes $\texttt{B}_i$ d'une classe `A` par rapport à `A` en fonction de la typicalité de chaque $\texttt{B}_i$ par rapport à `A`. Par exemple, les sous-classes `Olive`, `Mangue` et `Cerise` de la classe `FruitÀNoyau` peuvent être discriminées par rapport à `FruitÀNoyau` en fonction de leur typicalité par rapport à `FruitÀNoyau`.

Nous avons étudié la notion de typicalité en section 1.5 du chapitre 1 . Nous nous plaçons selon le point de vue de l'exemplaire où une classe peut être représentée par plusieurs prototypes qui sont les exemples les plus représentatifs de la classe ou encore les exemples les plus typiques de la classe. Les exemples les moins représentatifs d'une classe sont dits atypiques par rapport à cette classe. Dans l'étude de littérature de la section 1.5, les exemples représentent les instances des classes ; la typicalité est représentée pour discriminer les instances d'une classe par rapport à cette classe. Dans notre travail, nous représentons la typicalité entre classes plutôt qu'entre instances et classes. Cependant, les principes utilisés de la typicalité sont les mêmes ; les sous-

classes d'une classe `A` les plus représentatives par rapport à `A` sont les sous-classes typiques et les sous classes les moins représentatives par rapport à `A` sont les sous-classes atypiques.

L'acquisition des connaissances via une communauté en ligne permet d'acquérir plus facilement la typicalité d'une sous-classe `B` pour sa classe `A`. Nous verrons que la typicalité d'une sous-classe `B` par rapport à sa classe `A` peut être obtenue à l'aide de questions en ligne. L'acquisition de la typicalité des connaissances du domaine va permettre :

- dans un premier temps, de modifier l'ontologie du domaine en la raffinant grâce à la prise en compte de la typicalité ;
- dans un deuxième temps, de modifier le moteur du système de RÀPC en sélectionnant le(s) cas le(s) plus typique(s) durant la phase de remémoration par rapport au problème cible pour le ou les adapter et le ou les retourner en premier(s).

La méthode d'acquisition de la typicalité d'une sous-classe `B` par rapport à sa classe `A` est présentée dans la section 5.1. Le raffinement de l'ontologie grâce à la typicalité est présentée dans la section 5.2. La section 5.3 présente la prise en compte de la typicalité dans le moteur de RÀPC.

5.1 Typicalité d'une sous-classe dans une classe

Une façon d'acquérir le degré de typicalité d'une sous-classe dans une classe consiste à mesurer à quel point cette sous-classe est un bon exemple de sa superclasse. Certains psychologues ont proposé d'acquérir le degré de typicalité grâce à des tâches cognitives ou des questionnaires. Avec [Barsalou et Sewell, 1985], un nom de classe est donné aux participants d'une expérimentation, par exemple `Fruit`, et les participants doivent lister les instances de cette classe. Les instances les plus citées sont les plus typiques par rapport à leur classe. [Rosch, 1973], quant à elle, donne un nom de classe aux participants de son expérimentation, par exemple `Fruit`, en addition à une liste d'instances comme `Pomme`, `Mangue` ou `Avocat`. Les participants doivent noter sur une échelle à 7 modalités à quel degré une instance est un bon exemple pour sa classe.

Fondé sur ces travaux, nous proposons d'acquérir le degré de typicalité d'une sous-classe dans sa superclasse par des questions de la forme « À quel point $\texttt{B}_i$ est-il un bon exemple de `A` ? ». Une échelle d'évaluation sous forme de trois émoticônes est utilisée, où un score numérique est associé à chaque émoticône :

- -1 pour l'émoticône en désaccord () ;
- 0 pour l'émoticône neutre () ;
- 1 pour l'émoticône en accord ().

Pour chaque $\texttt{B}_i$, la moyenne des scores numériques associés aux évaluations des utilisateurs est calculée et est considérée comme le degré de typicalité de $\texttt{B}_i$ par rapport à `A`. Le degré de typicalité de $\texttt{B}_i$ par rapport à `A` est noté $\texttt{typicalité}(\texttt{B}_i,\texttt{A}) \in [-1,1]$.

Acquisition de la typicalité dans eTaaable. Un questionnaire a été soumis à 14 utilisateurs via le site aTaaable pour acquérir les degrés de typicalité pour les sous-classes des classes `FruitÀNoyau`, `FruitÀPépins` et `Baie`. Par exemple, l'acquisition du degré de typicalité de `Abricot` par rapport à `FruitÀNoyau` se fait via la question « À quel point `Abricot` est un bon exemple de `FruitÀNoyau` ? » Avant de débuter le questionnaire, le contexte de la cuisine est précisé aux utilisateurs. L'influence du contexte dans la typicalité est étudié en section 1.5.3 du chapitre 1. En effet, les réponses à la question « À quel point `Abricot` est un bon exemple

de FruitÀNoyau ? » seront différentes que le contexte soit la cuisine ou la botanique car les utilisateurs n'ont pas la même représentation des exemples prototypiques de FruitÀNoyau.

Le tableau 5.1 présente trois sous-tableaux représentant les degrés de typicalité acquis pour les sous-classes de FruitÀNoyau, FruitÀPépins et Baie. Les sous-classes de FruitÀNoyau, FruitÀPépins et Baie ont été acquises grâce à l'édition collaborative de la communauté en ligne d'aTaaable, ce qui explique par exemple que les classes Potiron, Tomate, Poivron et Concombre ont été classées comme sous-classes de FruitÀPépins. Pour chaque tableau, les sous-classes sont classées par ordre de degré de typicalité décroissant.

FruitÀNoyau	
Abricot	1,00
Mirabelle	1,00
Pêche	1,00
Cerise	0,92
Datte	0,42
Litchi	0,17
Mangue	0,08
Avocat	−0,17
Olive	−0,25

FruitÀPépins	
Pomme	0,67
Poire	0,50
Raisin	0,33
Citron	0,17
Melon	0,08
Orange	−0,17
Potiron	−0,58
Tomate	−0,58
Poivron	−0,67
Concombre	−1,00

Baie	
Framboise	1,00
Mûre	1,00
Myrtille	1,00
Groseille	0,67
Fraise	0,17
Raisin	−0,33
Kiwi	−0,92

Tableau 5.1 – Degrés de typicalité pour les sous-classes de FruitÀNoyau, FruitÀPépins et Baie.

Remarque : comme eTaaable ne manipule pas les instances mais les sous-classes de l'ontologie dans son raisonnement, nous représentons la typicalité d'une sous-classe par rapport à sa classe alors que [Barsalou et Sewell, 1985] et [Rosch, 1973] représentent la typicalité d'une instance par rapport à sa classe. Les deux méthodes sont cependant similaires car nous ne représentons que la typicalité de sous-classes qui indexent les recettes et dont le nom pourrait être celui d'une instance. Par exemple, la typicalité de Fraise par rapport à Baie peut être représentée tandis que la typicalité de FruitÀNoyau par rapport à Fruit ne peut pas être représentée dans eTaaable.

De plus, dans ce travail, la typicalité est seulement représentée pour les sous-classes $\mathtt{B}_i$ d'une classe A telles que $\mathtt{B}_i \sqsubseteq \mathtt{A} \in \mathtt{CD}$. Cette restriction peut soulever des problèmes qu'il serait envisageable de traiter dans de futurs travaux. En effet, une classe B pourrait être atypique de A car ce caractère atypique est hérité de classes plus génériques à A dans la hiérarchie. Par exemple, on pourrait considérer que Avocat est une classe atypique de FruitÀNoyau car Avocat est une classe atypique de Fruit. Dans ce travail, l'origine du caractère atypique d'une classe B pour sa classe A n'est pas prise en compte.

5.2 Raffinement de l'ontologie grâce à la typicalité

Dans cette section, nous proposons de diviser les sous-classes d'une superclasse en plusieurs sous-ensembles au regard de leur typicalité par rapport à la superclasse. Nous avons utilisé trois manières différentes pour établir les différents sous-ensembles des sous-classes $\mathtt{B}_i$ de A.

1. La première méthode consiste à utiliser des seuils sur la valeur de typicalité(B_i,A) afin de constituer trois sous-ensembles : le sous-ensemble des sous-classes typiques, le sous-ensemble des sous-classes normales et le sous-ensemble des sous-classes atypiques.
2. La deuxième méthode consiste à utiliser la méthode des k-moyennes sur la valeur de typicalité(B_i,A), afin de faire k sous-ensembles des sous-classes de B_i de A.
3. La troisième méthode consiste à construire un sous-ensemble différent des sous-classes B_i de A pour chaque valeur différente de typicalité(B_i,A).

Chacune de ces méthodes permet de séparer les sous-classes B_i en plusieurs sous-ensembles, en fonction de leur typicalité par rapport à A. Ces trois méthodes sont évaluées par la suite pour sélectionner la meilleure.

Ces sous-ensembles de sous-classes d'une superclasse sont ensuite utilisés pour raffiner l'ontologie du domaine en ajoutant des sous-classes intermédiaires à la superclasse dans sa hiérarchie. Raffiner l'ontologie a pour but de détailler davantage le processus de généralisation de la requête durant la phase de remémoration.

5.2.1 Raffinement de l'ontologie par la méthode des seuils

Le but de cette méthode est de diviser les sous-classes B_i de A en 3 sous-ensembles : le sous-ensemble des classes typiques, le sous-ensemble des classes normales et le sous-ensemble des classes atypiques. La construction de ces 3 sous-ensembles de sous-classes B_i a pour but de créer deux nouvelles classes subsumées par A : $A_{typique}$ et $A_{\substack{normal \\ ou\ typique}}$, afin de réorganiser la hiérarchie sous A. La figure 5.1 illustre le principe de réorganisation de la hiérarchie dont la racine est A.

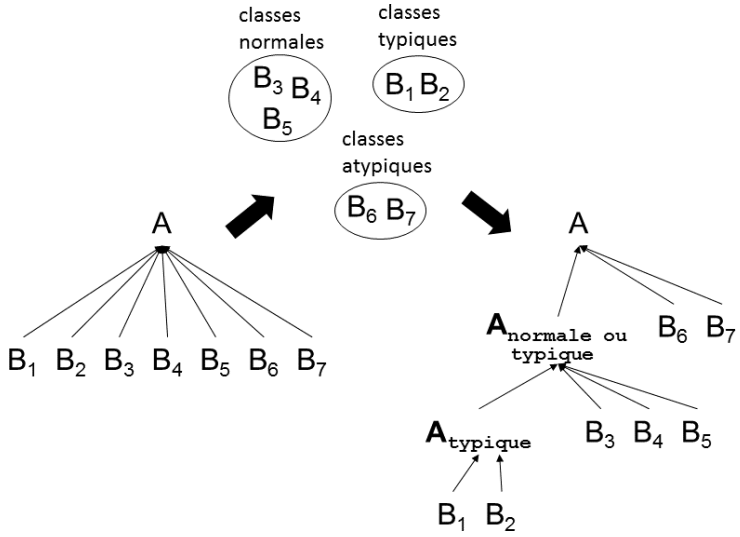


FIGURE 5.1 – Réorganisation des sous-classes B_i de A dans la partie de la hiérarchie dont la racine est A par la méthode des seuils.

La division des sous-classes B_i de A se fait sur le principe que, dans un même sous-ensemble, les sous-classes B_i ont toutes un degré de typicalité minimale par rapport à A. Pour définir chaque sous-ensemble, nous utilisons des seuils appliqués aux degrés de typicalité des sous-classes B_i.

– le sous-ensemble des sous-classes typiques correspond aux sous-classes B_i où typicalité(B_i,A) $\geqslant \beta_2$

- le sous-ensemble des sous-classes normales correspond aux sous-classes $\mathtt{B}_i$ où $\beta_1 \leqslant \mathtt{typicalité}(\mathtt{B}_i,\mathtt{A}) < \beta_2$.
- le sous-ensemble des sous-classes atypiques correspond aux sous-classes $\mathtt{B}_i$ où $\mathtt{typicalité}(\mathtt{B}_i,\mathtt{A}) < \beta_1$.

Sachant que $\mathtt{typicalité}(\mathtt{B}_i,\mathtt{A}) \in [-1,1]$, nous posons les seuils $\beta_1 = 0{,}0$ et $\beta_2 = 0{,}5$. Le premier seuil $\beta_1 = 0{,}0$ permet de séparer les sous-classes atypiques de A des sous-classes non atypiques de A. Le deuxième seuil $\beta_2 = 0{,}5$ permet de séparer les sous-classes non atypiques en sous-classes normales et en sous-classes typiques.

Comme l'illustre la figure 5.1, la construction de ces 3 sous-ensembles de sous-classes $\mathtt{B}_i$ de A permet de créer les deux nouvelles classes $\mathtt{A}_{\mathtt{typique}}$ et $\mathtt{A}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}}$ qui sont organisées de la façon suivante : $\mathtt{A}_{\mathtt{typique}}$ subsume l'ensemble des sous-classes typiques, $\mathtt{A}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}}$ subsume l'ensemble des sous-classes typiques et normales et A subsume l'ensemble des sous-classes typiques, normales et atypiques. Par conséquent :

- Si B est une classe typique de A, on aura $\mathtt{B} \sqsubseteq \mathtt{A}_{\mathtt{typique}} \in \mathtt{CD}$, $\mathtt{B} \sqsubseteq \mathtt{A}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}} \notin \mathtt{CD}$ et $\mathtt{B} \sqsubseteq \mathtt{A} \notin \mathtt{CD}$.
- Si B est une classe normale de A, on aura $\mathtt{B} \sqsubseteq \mathtt{A}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}} \in \mathtt{CD}$ et $\mathtt{B} \sqsubseteq \mathtt{A} \notin \mathtt{CD}$.
- Si B est une classe atypique de A, on aura $\mathtt{B} \sqsubseteq \mathtt{A} \in \mathtt{CD}$.

On aura aussi $\mathtt{A}_{\mathtt{typique}} \sqsubseteq \mathtt{A}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}} \in \mathtt{CD}$ et $\mathtt{A}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}} \sqsubseteq \mathtt{A} \in \mathtt{CD}$.

Application du raffinement de l'ontologie par la méthode des seuils à eTaaable. Le tableau 5.2 présente la répartition des sous-classes des 3 classes FruitÀNoyau, FruitÀPépins et Baie par la méthode des seuils à partir des degrés de typicalité obtenus dans le tableau 5.1. Par exemple :

- Avocat est un fruit à noyau atypique, c'est-à-dire $\mathtt{Avocat} \sqsubseteq \mathtt{FruitÀNoyau} \in \mathtt{CD}$, car $\mathtt{typicalité}(\mathtt{Avocat},\mathtt{FruitÀNoyau}) = -0{,}17 < 0$;
- Mangue est un fruit à noyau normal, c'est-à-dire $\mathtt{Mangue} \sqsubseteq \mathtt{FruitÀNoyau}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}} \in \mathtt{CD}$, car $0 \leqslant \mathtt{typicalité}(\mathtt{Mangue},\mathtt{FruitÀNoyau}) = 0{,}08 < 0{,}5$;
- Abricot est un fruit à noyau typique, c'est-à-dire $\mathtt{Abricot} \sqsubseteq \mathtt{FruitÀNoyau}_{\mathtt{typique}} \in \mathtt{CD}$, car $\mathtt{typicalité}(\mathtt{Abricot},\mathtt{FruitÀNoyau}) = 1{,}00 \geqslant 0{,}5$.

Soit une partie de la hiérarchie des aliments dont la racine est la classe FruitÀNoyau à réorganiser, présentée dans la figure 5.2. Selon le tableau 5.2, pour FruitÀNoyau, les sous-classes typiques sont Abricot, Mirabelle, Pêche et Cerise, les sous-classes normales sont Datte, Mangue et Litchi et les sous-classes atypiques sont Avocat et Olive. Ainsi, nous avons :

$$\models_{BC} \mathtt{Abricot} \sqcup \mathtt{Mirabelle} \sqcup \mathtt{Pêche} \sqcup \mathtt{Cerise} \sqsubseteq \mathtt{FruitÀNoyau}_{\mathtt{typique}},$$

$$\models_{BC} \mathtt{Datte} \sqcup \mathtt{Mangue} \sqcup \mathtt{Litchi} \sqsubseteq \mathtt{FruitÀNoyau}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}},$$

$$\models_{BC} \mathtt{Avocat} \sqcup \mathtt{Olive} \sqsubseteq \mathtt{FruitÀNoyau}.$$

Par conséquent :

$$\models_{BC} \mathtt{Abricot} \sqcup \mathtt{Mirabelle} \sqcup \mathtt{Pêche} \sqcup \mathtt{Cerise} \sqsubseteq \mathtt{FruitÀNoyau}_{\mathtt{typique}},$$

$$\models_{BC} \mathtt{FruitÀNoyau}_{\mathtt{typique}} \sqcup \mathtt{Datte} \sqcup \mathtt{Mangue} \sqcup \mathtt{Litchi} \sqsubseteq \mathtt{FruitÀNoyau}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}},$$

$$\models_{BC} \mathtt{FruitÀNoyau}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}} \sqcup \mathtt{FruitÀNoyau}_{\mathtt{typique}} \sqcup \mathtt{Avocat} \sqcup \mathtt{Olive} \sqsubseteq \mathtt{FruitÀNoyau}.$$

FruitÀNoyau		
Fruits à noyau typiques	Abricot	1,00
	Mirabelle	1,00
	Pêche	1,00
	Cerise	0,92
Fruits à noyau normaux	Datte	0,42
	Litchi	0,17
	Mangue	0,08
Fruits à noyau atypiques	Avocat	−0,17
	Olive	−0,25

FruitÀPépins		
Fruits à pépins typiques	Pomme	0,67
	Poire	0,50
Fruits à pépins normaux	Raisin	0,33
	Citron	0,17
	Melon	0,08
Fruits à pépins atypiques	Orange	−0,17
	Potiron	−0,58
	Tomate	−0,58
	Poivron	−0,67
	Concombre	−1,00

Baie		
Baies typiques	Framboise	1,00
	Mûre	1,00
	Myrtille	1,00
	Groseille	0,67
Baies normales	Fraise	0,17
Baies atypiques	Raisin	−0,33
	Kiwi	−0,92

TABLEAU 5.2 – Répartition des sous-classes des classes FruitÀNoyau, FruitÀPépins et Baie par la méthode des seuils.

En appliquant la procédure de réorganisation à la partie de la hiérarchie des aliments dont la racine est FruitÀNoyau, la nouvelle partie de hiérarchie produite est présentée dans la figure 5.3. Deux nouvelles classes, $\texttt{FruitÀNoyau}_{\substack{\texttt{normal}\\\texttt{ou typique}}}$ et $\texttt{FruitÀNoyau}_{\texttt{typique}}$, en gras, ont été ajoutées.

Remarque : au niveau de l'ontologie, il aurait été plus intuitif de créer les classes $\texttt{A}_{\texttt{typique}}$, $\texttt{A}_{\texttt{normal}}$ et $\texttt{A}_{\texttt{atypique}}$ avec $\texttt{A}_{\texttt{typique}} \sqsubseteq \texttt{A}$, $\texttt{A}_{\texttt{normal}} \sqsubseteq \texttt{A}$ et $\texttt{A}_{\texttt{atypique}} \sqsubseteq \texttt{A}$. Cependant, le choix de créer les classes $\texttt{A}_{\texttt{typique}}$ et $\texttt{A}_{\substack{\texttt{normal}\\\texttt{ou typique}}}$ avec $\texttt{A}_{\texttt{typique}} \sqsubseteq \texttt{A}_{\substack{\texttt{normal}\\\texttt{ou typique}}} \in \texttt{CD}$ et $\texttt{A}_{\substack{\texttt{normal}\\\texttt{ou typique}}} \sqsubseteq \texttt{A} \in \texttt{CD}$ a été effectué par rapport à l'exploitation future de ces classes, c'est-à-dire par rapport à la phase de remémoration du système de RÀPC qui utilise la hiérarchie réorganisée.

Conséquences du raffinement de l'ontologie sur la remémoration. La réorganisation de l'ontologie a plusieurs conséquences sur la phase de remémoration.

- Pour une requête composée d'un élément qui est une sous-classe typique B d'une classe A, la première étape de généralisation consiste à généraliser B en $\texttt{A}_{\texttt{typique}}$ afin de remémorer des cas indexés par des sous-classes typiques de A. La deuxième étape de généralisation consiste à généraliser B en $\texttt{A}_{\substack{\texttt{normal}\\\texttt{ou typique}}}$ afin de remémorer des cas indexés par des sous-classes typiques ou normales de A. La troisième étape de généralisation consiste à généraliser B en A afin de remémorer des cas indexés par des sous-classes de A.
- Pour une requête composée d'un élément qui est une sous-classe normale B d'une classe A, la première étape de généralisation consiste à généraliser B en $\texttt{A}_{\substack{\texttt{normal}\\\texttt{ou typique}}}$ afin de remémorer des cas indexés par des sous-classes typiques ou normales de A. La deuxième étape de

généralisation consiste à généraliser B en A afin de remémorer des cas indexés par des sous-classes de A.

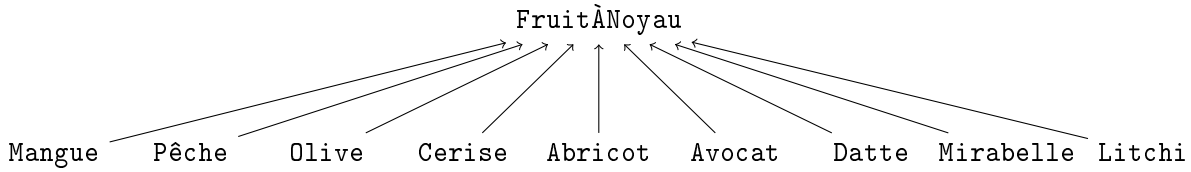


FIGURE 5.2 – Partie de la hiérarchie des aliments dont la racine est FruitÀNoyau.

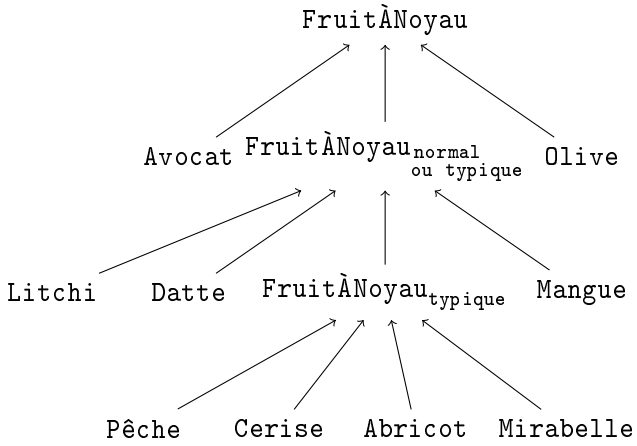


FIGURE 5.3 – Réorganisation de la hiérarchie des aliments dont la racine est FruitÀNoyau par la méthode des seuils.

Hypothèses sur l'apport du raffinement de l'ontologie par la méthode des seuils. Les hypothèses sous-jacentes au raffinement de l'ontologie grâce à la méthode des seuils sont :

(H1) avec une requête composée d'une sous-classe typique B_1 de A, les réponses pour lesquelles le cas remémoré est indexé par une sous-classe typique B_2 de A sont meilleures que les réponses pour lesquelles le cas remémoré est indexé par une sous-classe normale B_3 de A. Ainsi, les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe typique B_2 de A, indexant le cas remémoré, en $A_{typique}$ et spécialiser $A_{typique}$ en la sous-classe typique B_1 de A qui compose la requête sont meilleures que les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe normale B_3 de A, indexant le cas remémoré, en $A_{\substack{normal \\ ou\ typique}}$ et spécialiser $A_{\substack{normal \\ ou\ typique}}$ en la sous-classe typique B_1 de A qui compose la requête.

(H2) avec une requête composée d'une sous-classe typique B_1 de A, les réponses pour lesquelles le cas remémoré est indexé par une sous-classe typique ou normale B_2 de A sont meilleures que les réponses pour lesquelles le cas remémoré est indexé par une sous-classe atypique B_3 de A. Ainsi, les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe typique ou normale B_2 de A, indexant le cas remémoré, en $A_{\substack{normal \\ ou\ typique}}$ et spécialiser

$\mathtt{A}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}}$ en la sous-classe typique $\mathtt{B}_1$ de $\mathtt{A}$ qui compose la requête sont meilleures que les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe atypique $\mathtt{B}_3$ de $\mathtt{A}$, indexant le cas remémoré, en $\mathtt{A}$ et spécialiser $\mathtt{A}$ en la sous-classe typique $\mathtt{B}_1$ de $\mathtt{A}$ qui compose la requête.

(H3) avec une requête composée d'une sous-classe normale $\mathtt{B}_1$ de $\mathtt{A}$, les réponses pour lesquelles le cas remémoré est indexé par une sous-classe typique ou normale $\mathtt{B}_2$ de $\mathtt{A}$ sont meilleures que les réponses pour lesquelles le cas remémoré est indexé par une sous-classe atypique $\mathtt{B}_3$ de $\mathtt{A}$. Ainsi, les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe typique ou normale $\mathtt{B}_2$ de $\mathtt{A}$, indexant le cas remémoré, en $\mathtt{A}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}}$ et spécialiser $\mathtt{A}_{\substack{\mathtt{normal}\\ \mathtt{ou\ typique}}}$ en la sous-classe normale $\mathtt{B}_1$ de $\mathtt{A}$ qui compose la requête sont meilleures que les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe atypique $\mathtt{B}_3$ de $\mathtt{A}$, indexant le cas remémoré, en $\mathtt{A}$ et spécialiser $\mathtt{A}$ en la sous-classe normale $\mathtt{B}_1$ de $\mathtt{A}$ qui compose la requête.

Aucune hypothèse n'est faite sur les requêtes composées d'une sous-classe atypique $\mathtt{B}_1$ de $\mathtt{A}$ car il n'y a aucune raison de penser que remémorer un cas indexé par une sous-classe atypique $\mathtt{B}_2$ de $\mathtt{A}$ est meilleure que de remémorer un cas indexé par une sous-classe typique ou normale $\mathtt{B}_3$ de $\mathtt{A}$. Par conséquent, il n'y a aucune raison de penser que les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe atypique $\mathtt{B}_2$ de $\mathtt{A}$, indexant le cas remémoré, en $\mathtt{A}$ et spécialiser $\mathtt{A}$ en la sous-classe atypique $\mathtt{B}_1$ de $\mathtt{A}$ qui compose la requête sont meilleures que les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe typique ou normale $\mathtt{B}_3$ de $\mathtt{A}$, indexant le cas remémoré, en $\mathtt{A}$ et spécialiser $\mathtt{A}$ en la sous-classe atypique $\mathtt{B}_1$ de $\mathtt{A}$ qui compose la requête. Par exemple dans eTaaable, avec une requête composée de la classe atypique `Raisin` de la classe `Baie`, il n'y a aucune raison de penser que de remplacer l'ingrédient correspondant à la classe atypique `Kiwi` par l'ingrédient correspondant à `Raisin` dans une recette est meilleur ou plus mauvais que de remplacer l'ingrédient correspondant à la classe normale `Fraise` ou encore l'ingrédient correspondant à la classe typique `Framboise` par l'ingrédient correspondant à `Raisin` dans une recette.

C'est pourquoi, aucune classe additionnelle n'est ajoutée pour structurer les sous-classes atypiques ; dans la figure 5.1, les sous-classes atypiques sont directement subsumées par $\mathtt{A}$.

Avantages et critiques du raffinement de l'ontologie par la méthode des seuils. Avec le raffinement de l'ontologie par la méthode des seuils appliqués à la fonction de typicalité, le nombre des seuils et la valeur des seuils peut être modulable en fonction de la classe à réorganiser. Ainsi, les sous-classes $\mathtt{B}_i$ d'une classe $\mathtt{A}$ pourraient être divisées en seulement deux sous-ensembles permettant d'ajouter une seule classe additionnelle subsumée par $\mathtt{A}$, ou encore être divisée en plus de trois sous-ensembles permettant d'ajouter plus de deux classes additionnelles subsumées par $\mathtt{A}$.

Cependant, une contrainte à moduler le nombre de seuils et la valeur des seuils avec la méthode des seuils est que le nombre de seuils et la valeur des seuils doivent être fixés à la main. Ainsi, le nombre de seuils et la valeur des seuils ne sont pas toujours les plus pertinents. La méthode des k-moyennes est une solution à cette critique.

5.2.2 Raffinement de l'ontologie par la méthode des k-moyennes

La méthode des k-moyennes [MacQueen, 1967] est une méthode permettant de partitionner un ensemble de données numériques en k partitions. Chaque partition est représentée par un point

correspondant à la moyenne des données classées dans cette partition. Le but est de minimiser la somme des carrés des distances entre la moyenne de chaque partition et les données classées dans la partition. L'avantage de cette méthode est qu'il n'y a pas besoin de fixer les seuils à l'avance pour répartir des données en k partitions. De plus, même si la méthode des k-moyennes nécessite de fixer k à l'avance, différentes mesures existent afin de sélectionner k le plus optimal pour créer les partitions les plus satisfaisantes. Ainsi, contrairement à la méthode des seuils, k peut être fixé automatiquement. Les indices de Davies-Bouldin et de Calinski-Harabasz, sont parmi les mesures les plus utilisées pour déterminer k permettant de créer les partitions de meilleure qualité.

L'indice de Davies-Bouldin [Davies et Bouldin, 1979] mesure la qualité de k partitions en calculant pour chaque partition la similarité avec la partition pour laquelle elle est la plus proche. L'indice de Davies-Bouldin se calcule comme suit :

$$DB = \frac{1}{k}\sum_{i=1}^{k} max(\frac{I(P_i) + I(P_j)}{I(P_i,P_j)}),$$
$$\text{avec } I(P_i) = \frac{1}{n_i}\sum_{x \in P_i} |x - c_i|^2,$$
$$\text{et } I(P_i,P_j) = |c_i - c_j|.$$

où k est le nombre de partitions, P_i est la partition i, c_i est la moyenne de la partition P_i, n_i est le nombre de données de la partition P_i et $|\cdot|$ est la valeur absolue. Plus la valeur de DB est petite, meilleur est le partitionnement.

L'indice de Calinski-Harabasz [Caliński et Harabasz, 1974] prend en compte la variance interne des données des partitions et la variance entre partitions ainsi que le nombre de données et le nombre de partitions. L'indice de Calinski-Harabasz se calcule comme suit :

$$CH = \frac{B(k-1)}{W(n-k)} \text{ avec}$$
$$B = \sum_{i=1}^{k}\sum_{x \in P_i} n_i |c_i - \bar{X}|^2,$$
$$W = \sum_{i=1}^{k}\sum_{x \in P_i} |x - c_i|^2.$$

où k est le nombre de partitions, n est le nombre de données, P_i est la partition i, c_i est la moyenne de la partition P_i, n_i est le nombre de données de la partition P_i, $\bar{X}$ est la moyenne de toutes les données et $|\cdot|$ est la valeur absolue. Plus la valeur de CH est grande, meilleur est le partitionnement.

Chaque indice nécessite de calculer toutes les différentes combinaisons de partitions pour tous les k possibles afin de trouver le k final. Par exemple, pour la classe `FruitÀNoyau` contenant 9 sous-classes à classer, jusqu'à 8 différentes combinaisons de partitions sont calculées pour k allant de 2 à 9. Par exemple, d'après les degrés de typicalité donnés en tableau 5.1 :

- Pour $k = 2$, les 2 partitions calculées correspondent aux 2 sous-ensembles suivants : {`Abricot`,`Mirabelle`,`Pêche`,`Cerise`} et {`Datte`,`Mangue`,`Litchi` `Avocat`,`Olive`}.
- Pour $k = 3$, les 3 partitions calculées correspondent aux 3 sous-ensembles suivant : {`Abricot`,`Mirabelle`,`Pêche`,`Cerise`}, {`Datte`,`Mangue`,`Litchi`} et {`Avocat`,`Olive`}.
- ...

– Pour $k = 7$, les 7 partitions calculées correspondent aux 7 sous-ensembles suivant : {`Abricot`,`Mirabelle`,`Pêche`}, {`Cerise`}, {`Datte`}, {`Mangue`}, {`Litchi`}, {`Avocat`} et {`Olive`}. On s'arrête à $k = 7$ car les degrés de typicalité des sous-classes `Abricot`, `Mirabelle` et `Pêche` de `FruitÀNoyau` par rapport à `FruitÀNoyau` sont égaux.

Nous avons choisi d'utiliser l'indice de Calinski-Harabasz car il prend en compte le nombre de données et le nombre de partitions. Par exemple, pour la classe `FruitÀNoyau`, l'indice de Calinski-Harabasz désigne $k = 2$ pour calculer les partitions les plus satisfaisantes. On considère que quel que soit k, la partition 1 est la partition contenant les classes les moins typiques et la partition k contient les classes les plus typiques.

À l'instar de la méthode des seuils qui permet de créer deux nouvelles classes subsumées par `A` à partir des trois sous-ensembles des sous-classes typiques, des sous-classes normales et des sous-classes atypiques de `A`, la méthode des k-moyennes permet de créer $k-1$ nouvelles classes subsumées par `A` à partir des k partitions des sous-classes de `A`. Ces nouvelles classes subsumées par `A` sont $\mathtt{A}_1$, $\mathtt{A}_2$, ..., $\mathtt{A}_{k-1}$, où $\mathtt{A}_{i-1}$ subsume l'ensemble des sous-classes classées dans la partition i. Plus i est élevé, plus la partition i contient des sous-classes typiques de `A`. Ainsi, $\mathtt{A}_{k-1}$ subsumera les sous-classes $\mathtt{B}_i$ les plus typiques par rapport à `A`. Ajouter des classes intermédiaires $\mathtt{A}_i$ sous `A` permet de réorganiser la hiérarchie comme présenté en figure 5.4. Dans cette illustration, la méthode des k-moyennes permet de répartir les sous-classes dans 3 partitions. La $3^{ème}$ partition permet de créer la classe $\mathtt{A}_2$ et la $2^{ème}$ partition permet de créer la classe $\mathtt{A}_1$. Les classes $\mathtt{A}_1$ et $\mathtt{A}_2$ sont alors ajoutées sous la classe `A`. Si une sous-classe `B` a été classée dans la partition i, alors $\mathtt{B} \sqsubseteq \mathtt{A}_{i-1} \in \mathtt{CD}$. On aura aussi pour chaque i, $\mathtt{A}_i \sqsubseteq \mathtt{A}_{i-1} \in \mathtt{CD}$. La classe $\mathtt{A}_0$ est équivalente à la classe `A`, et n'est donc pas ajoutée sous `A`

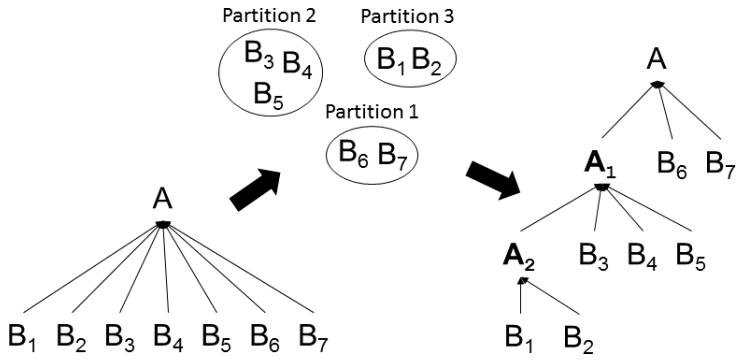


FIGURE 5.4 – Réorganisation des sous-classes de `A` dans la hiérarchie dont la racine est `A` par la méthode des k-moyennes.

Application du raffinement de l'ontologie par la méthode des k-moyennes à eTaaable. Le tableau 5.3 présente la répartition des sous-classes des 3 classes `FruitÀNoyau`, `FruitÀPépins` et `Baie` par la méthode des k-moyennes à partir des degrés de typicalité obtenus dans le tableau 5.1. Grâce à l'indice de Calinski-Harabasz, les sous-classes de `FruitÀNoyau`, `FruitÀPépins` et `Baie` ont été respectivement divisées en 2, 3 et 3 sous-ensembles. Par exemple, `Avocat` a été classée dans la $1^{ère}$ partition de `FruitÀNoyau` dont la moyenne est 0,05 tandis que `Abricot` a été classée dans $2^{ème}$ partition de `FruitÀNoyau` dont la moyenne est 0,98.

FruitÀNoyau			
partition 2	Abricot	1,00	0,98
	Mirabelle	1,00	
	Pêche	1,00	
	Cerise	0,92	
partition 1	Datte	0,42	0,05
	Litchi	0,17	
	Mangue	0,08	
	Avocat	−0,17	
	Olive	−0,25	

FruitÀPépins			
partition 3	Pomme	0,67	0,50
	Poire	0,50	
	Raisin	0,33	
partition 2	Citron	0,17	0,03
	Melon	0,08	
	Orange	−0,17	
partition 1	Potiron	−0,58	0,35
	Tomate	−0,58	
	Poivron	−0,67	
	Concombre	−1,00	

Baie			
partition 3	Framboise	1,00	0,91
	Mûre	1,00	
	Myrtille	1,00	
	Groseille	0,67	
partition 2	Fraise	0,17	−0,08
	Raisin	−0,33	
partition 1	Kiwi	−0,92	−0,92

TABLEAU 5.3 – Répartition des sous-classes de la classe FruitÀNoyau, FruitÀPépins et Baie par la méthode des k-moyennes. Les valeurs indiquées dans la dernière colonne de chaque tableau correspond à la moyenne de la partition i.

En appliquant la procédure de réorganisation à la partie de la hiérarchie dont la racine est FruitÀNoyau par la méthode des k-moyennes, la nouvelle partie de hiérarchie produite est présentée dans la figure 5.5. Une nouvelle classe, $\texttt{FruitÀNoyau}_1$, en gras, a été ajoutée.

Conséquences du raffinement de l'ontologie par la méthode des k-moyennes sur la remémoration. La première étape de généralisation d'une sous-classe B de la classe A permet de remémorer des cas indexés par des sous-classes de A dont le degré de typicalité est proche ou plus grand que celui de B. L'étape suivante de généralisation permet de retrouver les cas indexés par des sous-classes de B dont le degré de typicalité est plus faible que B mais pour lesquels la différence du degré de typicalité est minimale avec B. Chaque étape suivante de généralisation, jusqu'à la généralisation de B en A, permet de minimiser la différence de degré de typicalité entre B et les sous-classes indexant les cas remémorés.

Par exemple, d'après la figure 5.5, la première étape de généralisation de Cerise en $\texttt{FruitÀNoyau}_1$ permet de remémorer tous les cas indexés par les sous-classes Abricot, Mirabelle et Pêche ayant un degré de typicalité proche de Cerise. La deuxième étape de généralisation de Cerise en FruitÀNoyau permet de remémorer les cas indexés par FruitÀNoyau car seulement une classe intermédiaire a été ajoutée pour la classe FruitÀNoyau.

Hypothèse sur l'apport du raffinement de l'ontologie par la méthode des k-moyennes. L'hypothèse sous-jacente du raffinement de l'ontologie grâce à la méthode des k-moyennes est :

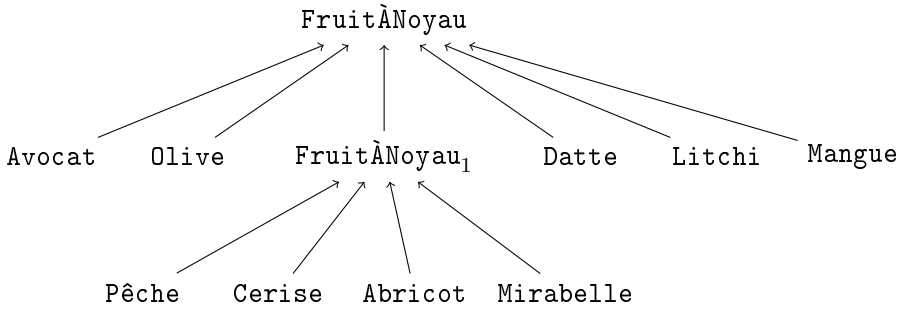


FIGURE 5.5 – Réorganisation de la hiérarchie des aliments dont la racine est `FruitÀNoyau` par la méthode des k-moyennes.

(H4) avec une requête composée d'une sous-classe $\mathtt{B}_1$ subsumée par $\mathtt{A}_i$, les réponses pour lesquelles le cas remémoré est indexé par une autre sous-classe $\mathtt{B}_2$ de $\mathtt{A}_i$ (avec $\models_{BC} \mathtt{B}_2 \sqsubseteq \mathtt{A}_i$) sont meilleures que les réponses pour lesquelles le cas remémoré est indexé par une sous-classe $\mathtt{B}_3$ de $\mathtt{A}_{i-j}$, avec $i \geqslant 1$ et $0 < j < i$. Ainsi, les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe $\mathtt{B}_2$ de `A`, indexant le cas remémoré, en $\mathtt{A}_i$ et spécialiser $\mathtt{A}_i$ en la sous-classe $\mathtt{B}_1$ de $\mathtt{A}_i$ qui compose la requête sont meilleures que les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe $\mathtt{B}_3$ de `A`, indexant le cas remémoré, en $\mathtt{A}_{i-j}$ et spécialiser $\mathtt{A}_{i-j}$ en la sous-classe $\mathtt{B}_1$ de $\mathtt{A}_i$ qui compose la requête.

5.2.3 Raffinement de l'ontologie par la méthode des niveaux

Une dernière solution pour diviser les sous-classes $\mathtt{B}_i$ d'une classe `A` en plusieurs sous-ensembles est de créer un sous-ensemble pour chaque degré de typicalité différent. Par exemple, pour la classe `FruitÀNoyau` et avec les degrés de typicalité donnés en tableau 5.1 pour les sous-classes de `FruitÀNoyau`, 7 sous-ensembles sont créés et ils correspondent à : {`Abricot`,`Mirabelle`,`Pêche`}, {`Cerise`}, {`Datte`}, {`Mangue`}, {`Litchi`}, {`Avocat`} et {`Olive`}. Les sous-classes `Abricot`, `Mirabelle` et `Pêche` sont dans le même sous-ensemble car elles ont toute le même degré de typicalité pour `FruitÀNoyau`, qui est de 1,0.

Chaque sous-ensemble de sous-classes $\mathtt{B}_i$ d'une classe `A` permet de créer une nouvelle classe subsumée par `A`. La méthode des niveaux appliquée aux sous-classes de `A` permet de générer $k-1$ nouvelles classes, où k est le nombre de degré de typicalité différents obtenus pour les sous-classes de `A`. Le $i^{ème}$ sous-ensemble dont le degré de typicalité est le $k-(i-1)^{ème}$ degré de typicalité le plus élevé pour la classe `A` permet la création de la classe $\mathtt{A}_{i-1}$. Plus i est grand, plus la classe $\mathtt{A}_i$ est typique pour `A`. Ainsi, $\mathtt{A}_{k-1}$ subsume les classes les plus typiques de `A`. Ajouter les classes intermédiaires $\mathtt{A}_i$ sous la classe `A` permet de réorganiser la hiérarchie comme dans la figure 5.6. Dans l'exemple de la figure 5.6, $\mathtt{B}_5$ est la sous-classe de `A` possédant le degré de typicalité le plus faible, ainsi $\mathtt{B}_5$ est directement subsumée par `A`. Les sous-classes $\mathtt{B}_1$ et $\mathtt{B}_2$ ont le même degré de typicalité pour `A`, ce qui explique qu'elles sont subsumées par la même classe $\mathtt{A}_3$. Si une sous-classe `B` a été classée dans le sous-ensemble i, alors $\mathtt{B} \sqsubseteq \mathtt{A}_{i-1} \in \mathtt{CD}$. On aura aussi pour chaque i, $\mathtt{A}_i \sqsubseteq \mathtt{A}i-1 \in \mathtt{CD}$. La classe $\mathtt{A}_0$ est équivalente à la classe `A`, et n'est donc pas ajoutée sous `A`

Application du raffinement de l'ontologie par la méthode des niveaux à eTaaable. Le tableau 5.4 présente la répartition des sous-classes des 3 classes `FruitÀNoyau`, `FruitÀPépins`

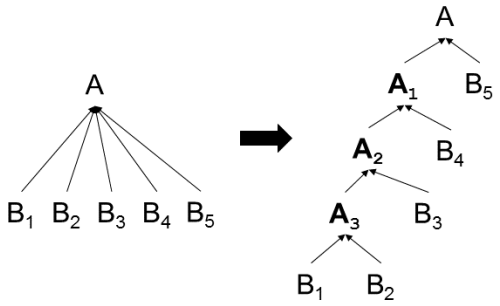


FIGURE 5.6 – Réorganisation des sous-classes de `A` dans la hiérarchie dont la racine est `A` par la méthode des niveaux.

et `Baie` par la méthode des niveaux à partir des degrés de typicalité présentés dans le tableau 5.1. Dans chaque tableau, S_i correspond au $i^{ème}$ sous-ensemble.

FruitÀNoyau		
S_7	Abricot	1,00
	Mirabelle	1,00
	Pêche	1,00
S_6	Cerise	0,92
S_5	Datte	0,42
S_4	Litchi	0,17
S_3	Mangue	0,08
S_2	Avocat	−0,17
S_1	Olive	−0,25

FruitÀPépins		
S_9	Pomme	0,67
S_8	Poire	0,50
S_7	Raisin	0,33
S_6	Citron	0,17
S_5	Melon	0,08
S_4	Orange	−0,17
S_3	Potiron	−0,58
	Tomate	−0,58
S_2	Poivron	−0,67
S_1	Concombre	−1,00

Baie		
S_5	Framboise	1,00
	Mûre	1,00
	Myrtille	1,00
S_4	Groseille	0,67
S_3	Fraise	0,17
S_2	Raisin	−0,33
S_1	Kiwi	−0,92

TABLEAU 5.4 – Répartition des sous-classes des classes `FruitÀNoyau`, `FruitÀPépins` et `Baie` par la méthode des niveaux.

En appliquant la procédure de réorganisation à la partie de la hiérarchie dont la racine est `FruitÀNoyau` par la méthode des niveaux, la nouvelle partie de hiérarchie produite est présentée dans la figure 5.7. Six nouvelles classes, $\texttt{FruitÀNoyau}_1$, $\texttt{FruitÀNoyau}_2$, ..., $\texttt{FruitÀNoyau}_6$, en gras, ont été ajoutées. `FruitÀNoyau` subsume la sous-classe de `FruitÀNoyau` la moins typique : `Olive` ; $\texttt{FruitÀNoyau}_1$ subsume la sous-classe de `FruitÀNoyau` qui a le $2^{ème}$ degré de typicalité le plus faible : `Avocat`. $\texttt{FruitÀNoyau}_6$ subsume les sous-classes de `FruitÀNoyau` les plus typiques, c'est-à-dire ayant le degré de typicalité le plus élevé : `Abricot`, `Mirabelle` et `Pêche`.

Conséquences du raffinement de l'ontologie par la méthode des niveaux sur la remémoration. La raffinement de l'ontologie selon la méthode de répartition des niveaux a plusieurs conséquences sur la phase de remémoration. La première étape de généralisation d'une sous-classe `B` de la classe `A` permet de remémorer les cas indexés par des sous-classes qui sont au moins aussi typiques que `B`. L'étape suivante de généralisation permet de retrouver les cas indexés par des sous-classes de `B` dont le degré de typicalité est plus faible que `B` mais pour lesquels la différence du degré de typicalité est minimale avec `B`. Chaque étape suivante de généralisation, jusqu'à la

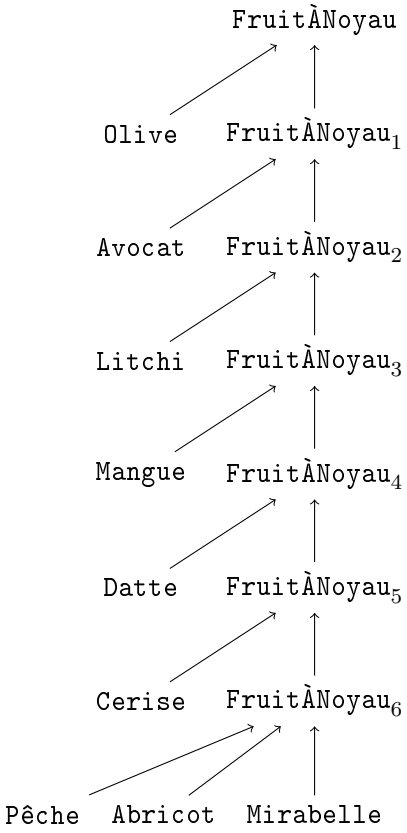


FIGURE 5.7 – Réorganisation de la hiérarchie des aliments dont la racine est `FruitÀNoyau` par la méthode des niveaux.

généralisation de `B` en `A`, permet de minimiser la différence de degré de typicalité entre `B` et les sous-classes indexant les cas remémorés.

Par l'exemple, d'après la figure 5.7, la première étape de généralisation de `Cerise` en `FruitÀNoyau`$_5$ permet de remémorer les cas indexés par les sous-classes `Abricot`, `Mirabelle` et `Pêche` qui ont un degré de typicalité plus élevé que `Cerise`. La deuxième étape de généralisation de `Cerise` en `FruitÀNoyau`$_5$ permet de remémorer les cas indexés par la sous-classe `Datte` qui a un degré de typicalité plus faible que `Cerise` mais qui est la classe ayant la plus petite différence de degré de typicalité avec `Cerise`.

Hypothèse sur l'apport du raffinement de l'ontologie par la méthode des niveaux. L'hypothèse sous-jacente à la réorganisation de l'ontologie grâce à la méthode des niveaux est la même que pour la méthode des k-moyennes.

5.3 Intégration de la typicalité dans le moteur de RÀPC

Dans cette section nous montrons comment la typicalité d'une sous-classe pour une classe peut être prise en compte sans modifier l'ontologie du domaine de RÀPC$_{\text{std}}$. Une solution est d'utiliser la typicalité dans le moteur de RÀPC à la fin de la phase de remémoration afin que le cas le plus typique soit sélectionné et adapté pour être présenté en premier à l'utilisateur.

5.3.1 Prise en compte de la typicalité dans le processus de remémoration

Utiliser la typicalité dans un système de RÀPC peut permettre de sélectionner le(s) cas remémoré(s) le(s) plus typique(s) par rapport au problème cible parmi l'ensemble des cas remémorés. La phase de sélection introduite en RÀPC par [Aamodt et Plaza, 1994] est décrite dans la section 1.6.1 du chapitre 1. La sélection du ou des cas le(s) plus typique(s) par rapport au problème cible permettra de retourner ce(s) cas en premier[30].

Soit $\mathtt{Q}_{ex}$ = Recette ⊓ Tarte ⊓ Cerise ⊓ Framboise une requête. Avec Cerise ⊑ FruitÀNoyau et Framboise ⊑ Baie des UC de l'ontologie du domaine, $\mathtt{Q}_{ex}$ peut être généralisée en $\Gamma(\mathtt{Q}_{ex})$ = Recette ⊓ Tarte ⊓ FruitÀNoyau ⊓ Baie. Avec Mirabelle ⊑ FruitÀNoyau et Fraise ⊑ Baie des UC de l'ontologie du domaine, $idx(\mathtt{R}_{ex})$ = Recette ⊓ Tarte ⊓ PâteÀTarte ⊓ Mirabelle ⊓ Fraise ⊓ Sucre est l'index de $\mathtt{R}_{ex}$, une des recettes remémorées grâce à $\Gamma(\mathtt{Q}_{ex})$. La sélection de $\mathtt{R}_{ex}$ comme faisant partie des cas remémorés les plus typiques par rapport à la requête $\mathtt{Q}_{ex}$ est une décision multi-critère [Giard et Roy, 1985] pour laquelle les critères sont : la sous-classe Mirabelle indexant $\mathtt{R}_{ex}$ doit être au moins aussi typique par rapport à FruitÀNoyau que la sous-classe Cerise et la sous-classe Fraise indexant $\mathtt{R}_{ex}$ doit être au moins aussi typique par rapport à Baie que la sous-classe Framboise.

Une façon de sélectionner les cas qui satisfont le plus les différents critères de typicalité par rapport à une requête Q est d'utiliser l'optimum de Pareto [Feldman et Serrano, 2006] :

- Cas_1 est au au moins aussi typique que Cas_2 par rapport à une requête Q, noté $Cas_1 \succeq_T^{\mathtt{Q}} Cas_2$, si pour tout critère, Cas_1 est au moins aussi bon que Cas_2 pour ce critère
- Cas_1 est plus typique que Cas_2 pour une requête Q, noté $Cas_1 \succ_T^{\mathtt{Q}} Cas_2$, si $Cas_1 \succeq_T^{\mathtt{Q}} Cas_2$ et $Cas_2 \not\succeq_T^{\mathtt{Q}} Cas_1$.

Si $Cas_1 \succ_T^{\mathtt{Q}} Cas_2$, alors on dit que Cas_2 est dominé par Cas_1 par rapport à Q. Le front de Pareto est défini par l'ensemble des cas qui ne sont pas dominés par rapport à Q : $\{Cas \in \mathtt{BaseC} \mid \nexists Cas' \in \mathtt{BaseC}, Cas' \succ Cas\}$, où BaseC correspond à la base de cas. Dans le cas où le front de Pareto contient un seul cas ou ne contient que des cas qui satisfont de manière égale tous les critères de typicalité par rapport à Q, alors ces cas sont les cas les plus typiques par rapport à la requête Q et sont sélectionnés pour être adaptés et retournés en premier. Si ce n'est pas le cas, alors les cas du front de Pareto doivent être différenciés afin de sélectionner les cas les plus typiques par rapport à la requête. Dans ce travail, nous établissons une méthode pour différencier les cas du front de Pareto en introduisant un nouveau critère pour diminuer la taille du front de Pareto.

Le critère disant que Mirabelle indexant la recette $\mathtt{R}_{ex}$ doit être au moins aussi typique par rapport à FruitÀNoyau que Cerise est évalué en calculant la différence de typicalité entre Cerise et Mirabelle par rapport à FruitÀNoyau. Comme Mirabelle doit être au moins aussi typique que Cerise par rapport à FruitÀNoyau, alors cette différence de typicalité sera considérée comme nulle si la typicalité de Cerise est strictement plus petite que Mirabelle par rapport à FruitÀNoyau. La différence de typicalité entre Cerise et Mirabelle par rapport à FruitÀNoyau, notée $\Delta_{typ}(\mathtt{Mirabelle} \overset{\mathtt{FruitÀNoyau}}{\rightsquigarrow} \mathtt{Cerise})$ est calculée comme suit :

$$\begin{aligned}
&\Delta_{typ}(\mathtt{Mirabelle} \overset{\mathtt{FruitÀNoyau}}{\rightsquigarrow} \mathtt{Cerise}) \\
&= \max(0; \mathtt{typicalité}(\mathtt{Cerise},\mathtt{FruitÀNoyau}) - \mathtt{typicalité}(\mathtt{Mirabelle},\mathtt{FruitÀNoyau})) \\
&= \max(0; 0{,}92 - 1{,}0) \\
&= 0
\end{aligned}$$

30. Nous utiliserons par la suite la forme plurielle pour faciliter la lecture.

De la même façon, la différence de typicalité entre `Framboise` et `Fraise` par rapport à `Baie`, notée $\Delta_{typ}(\texttt{Fraise} \overset{\texttt{Baie}}{\rightsquigarrow} \texttt{Framboise})$ est calculée comme suit :

$$\begin{aligned}
&\Delta_{typ}(\texttt{Fraise} \overset{\texttt{Baie}}{\rightsquigarrow} \texttt{Framboise}) \\
&= \max(0; \texttt{typicalité}(\texttt{Framboise},\texttt{Baie}) - \texttt{typicalité}(\texttt{Fraise},\texttt{Baie})) \\
&= \max(0; 1{,}0 - 0{,}17) \\
&= 0{,}83
\end{aligned}$$

D'une façon générale, la différence de typicalité entre une classe $\texttt{B}_i$ et une classe $\texttt{B}_j$ par rapport à une classe `A` avec $\texttt{B}_i \sqsubseteq \texttt{A}$ et $\texttt{B}_j \sqsubseteq \texttt{A}$) est défini par :

$$\Delta_{typ}(\texttt{B}_i \overset{\texttt{A}}{\rightsquigarrow} \texttt{B}_j) = \max(0; \texttt{typicalité}(\texttt{B}_j,\texttt{A}) - \texttt{typicalité}(\texttt{B}_i,\texttt{A})) \tag{5.1}$$

Pour $\texttt{R}_{ex}$, la différence de typicalité entre `Mirabelle` et `Cerise` par rapport à `FruitÀNoyau` et la différence de typicalité entre `Fraise` et `Framboise` par rapport à `Baie` sont agrégées afin de calculer la typicalité de $\texttt{R}_{ex}$ pour $\texttt{Q}_{ex}$, notée $\Delta_{casTyp}(\texttt{R}_{ex},\texttt{Q}_{ex})$, et est calculée comme suit :

$$\begin{aligned}
&\Delta_{casTyp}(\texttt{R}_{ex},\texttt{Q}_{ex}) \\
&= \Delta_{typ}(\texttt{Mirabelle} \overset{\texttt{FruitÀNoyau}}{\rightsquigarrow} \texttt{Cerise}) \\
&\quad + \ \Delta_{typ}(\texttt{Fraise} \overset{\texttt{Baie}}{\rightsquigarrow} \texttt{Framboise}) \\
&= 0 + 0{,}83 = 0{,}83
\end{aligned}$$

D'une façon générale, lorsqu'un cas $\mathit{Cas} \in \texttt{BaseC}$ a été remémoré à partir d'une requête `Q`, la typicalité de Cas par rapport à `Q` est calculée par la fonction notée Δ_{casTyp} :

$$\Delta_{casTyp}(\mathit{Cas},\texttt{Q}) = \sum_{i=1}^{n} \Delta_{typ}(\texttt{A}_i \overset{\texttt{C}_i}{\rightsquigarrow} \texttt{Q}_i) \tag{5.2}$$

où $\texttt{Q} = \texttt{Q}_1 \sqcap \texttt{Q}_2 \sqcap \ldots \sqcap \texttt{Q}_n$, $\texttt{Q}_i$ est une classe de `Q` qui a été généralisée en $\texttt{C}_i$ et $\texttt{A}_i$ est une sous-classe de $\texttt{C}_i$ qui indexe Cas.

La fonction Δ_{casTyp} est utilisée pour calculer la typicalité de chaque cas du front de Pareto afin de diminuer la taille du front de Pareto.

- $\mathit{Cas}_1 \succeq_T^{\texttt{Q}} \mathit{Cas}_2$, signifiant que Cas_1 est au moins aussi typique que Cas_2 par rapport à `Q`, si $\Delta_{casTyp}(\mathit{Cas}_2,\texttt{Q}) \geqslant \Delta_{casTyp}(\mathit{Cas}_1,\texttt{Q})$.
- $\mathit{Cas}_1 \succ_T^{\texttt{Q}} \mathit{Cas}_2$, signifiant que Cas_1 est plus typique que Cas_2 par rapport à `Q`, si :

$$\begin{aligned}
&\Delta_{casTyp}(\mathit{Cas}_2,\texttt{Q}) \geqslant \Delta_{casTyp}(\mathit{Cas}_1,\texttt{Q}) \text{ et} \\
&\Delta_{casTyp}(\mathit{Cas}_1,\texttt{Q}) < \Delta_{casTyp}(\mathit{Cas}_2,\texttt{Q})
\end{aligned}$$

Dans eTaaable par exemple, soit `Abricot` $\sqsubseteq$ `FruitÀNoyau`, `Pêche` $\sqsubseteq$ `FruitÀNoyau`, `Mangue` $\sqsubseteq$ `FruitÀNoyau`, `Framboise` $\sqsubseteq$ `Baie`, `Fraise` $\sqsubseteq$ `Baie`, `Kiwi` $\sqsubseteq$ `Baie` et `Myrtille` $\sqsubseteq$ `Baie` des formules de la base de connaissances constituant les connaissances du domaine. Pour la requête $\texttt{Q}_{ex}$ = `Recette` $\sqcap$ `Tarte` $\sqcap$ `Abricot` $\sqcap$ `Framboise`, la requête généralisée $\mathit{\Gamma}(\texttt{Q}_{ex})$ = `Recette` $\sqcap$ `Tarte` $\sqcap$ `FruitÀNoyau` $\sqcap$ `Baie` permet de remémorer la recette $\texttt{R}_1$ dont l'index est $idx(\texttt{R}_1)$ = `Recette` $\sqcap$ `Tarte` $\sqcap$ `PâteÀTarte` $\sqcap$ `Pêche` $\sqcap$ `Framboise` et la recette $\texttt{R}_2$ dont l'index est $idx(\texttt{R}_2)$ = `Recette` $\sqcap$ `Tarte` $\sqcap$ `PâteÀTarte` $\sqcap$ `Mangue` $\sqcap$ `Kiwi` $\sqcap$ `CrèmePâtissiere`. Les deux critères pour sélectionner le cas remémoré le plus typique par rapport à $\texttt{Q}_{ex}$ sont :

1. dans le cas remémoré, la classe subsumée par FruitÀNoyau qui indexe le cas est au moins aussi typique que Abricot par rapport à FruitÀNoyau ;
2. dans le cas remémoré la classe subsumée par Baie qui indexe le cas est au moins aussi typique que Framboise par rapport à Baie.

Avec typicalité(Mangue,FruitÀNoyau) $<$ typicalité(Pêche,FruitÀNoyau) $\leqslant$ typicalité(Abricot,FruitÀNoyau), $\mathtt{R}_1$ est meilleure pour le premier critère que $\mathtt{R}_2$. Avec typicalité(Kiwi,Baie) $<$ typicalité(Fraise,Baie) $<$ typicalité(Framboise,Baie), $\mathtt{R}_1$ est meilleure que $\mathtt{R}_2$ pour le deuxième critère. Comme $\mathtt{R}_1$ est meilleure que $\mathtt{R}_2$ pour les deux seuls critères, $\mathtt{R}_1$ domine $\mathtt{R}_2$. Soit $\mathtt{R}_3$ une autre recette remémorée dont l'index est $idx(\mathtt{R}_3) = \mathtt{Recette} \sqcap \mathtt{Tarte} \sqcap \mathtt{PâteÀTarte} \sqcap \mathtt{Mangue} \sqcap \mathtt{Myrtille} \sqcap \mathtt{Sucre}$. Avec typicalité(Mangue,FruitÀNoyau) $<$ typicalité(Pêche,FruitÀNoyau) $\leqslant$ typicalité(Abricot,FruitÀNoyau), $\mathtt{R}_1$ est meilleure que $\mathtt{R}_3$ pour le premier critère. Avec typicalité(Fraise,Baie) $<$ typicalité(Myrtille,Baie) $\leqslant$ typicalité(Framboise,Baie), $\mathtt{R}_3$ est meilleure pour le deuxième critère que $\mathtt{R}_1$. Comme $\mathtt{R}_1$ est meilleure que $\mathtt{R}_3$ pour le premier critère et $\mathtt{R}_3$ est meilleure que $\mathtt{R}_1$ pour le deuxième critère, alors aucune des recettes $\mathtt{R}_1$ et $\mathtt{R}_3$ n'est dominée par l'autre ; ces deux recettes forment le front de Pareto.

Si typicalité(Abricot,FruitÀNoyau) = 1,00, typicalité(Pêche,FruitÀNoyau) = 1,00 et typicalité(Mangue,FruitÀNoyau) = 0,08, alors

$$\begin{aligned}\Delta_{typ}(\mathtt{Pêche} \overset{\mathtt{FruitÀNoyau}}{\rightsquigarrow} \mathtt{Abricot}) &= \max(0; 1{,}00 - 1{,}00) = 0\\ \Delta_{typ}(\mathtt{Mangue} \overset{\mathtt{FruitÀNoyau}}{\rightsquigarrow} \mathtt{Abricot}) &= \max(0; 1{,}00 - 0{,}08) = 0{,}92\end{aligned}$$

Si typicalité(Framboise,FruitÀNoyau) = 1,00, typicalité(Myrtille,FruitÀNoyau) = 1,00 et typicalité(Fraise,FruitÀNoyau) = 0,17, alors

$$\begin{aligned}\Delta_{typ}(\mathtt{Myrtille} \overset{\mathtt{FruitÀNoyau}}{\rightsquigarrow} \mathtt{Framboise}) &= \max(0; 1{,}00 - 1{,}00) = 0\\ \Delta_{typ}(\mathtt{Fraise} \overset{\mathtt{FruitÀNoyau}}{\rightsquigarrow} \mathtt{Framboise}) &= \max(0; 1{,}00 - 0{,}17) = 0{,}83\end{aligned}$$

D'après les précédentes équations :

$$\begin{aligned}\Delta_{casTyp}(\mathtt{R}_1,\mathtt{Q}_{ex}) &= \Delta_{typ}(\mathtt{Pêche} \overset{\mathtt{FruitÀNoyau}}{\rightsquigarrow} \mathtt{Abricot})\\ &\quad + \Delta_{typ}(\mathtt{Fraise} \overset{\mathtt{Baie}}{\rightsquigarrow} \mathtt{Framboise})\\ &= 0{,}83\\ \Delta_{casTyp}(\mathtt{R}_3,\mathtt{Q}_{ex}) &= \Delta_{typ}(\mathtt{Mangue} \overset{\mathtt{FruitÀNoyau}}{\rightsquigarrow} \mathtt{Abricot})\\ &\quad + \Delta_{typ}(\mathtt{Myrtille} \overset{\mathtt{Baie}}{\rightsquigarrow} \mathtt{Framboise})\\ &= 0{,}92\end{aligned}$$

Comme $\Delta_{casTyp}(\mathtt{R}_3,\mathtt{Q}_{ex}) > \Delta_{casTyp}(\mathtt{R}_1,\mathtt{Q}_{ex})$, $\mathtt{R}_1 \succeq_T^{\mathtt{Q}_{ex}} \mathtt{R}_3$. $\mathtt{R}_1$ sera adaptée et retournée avant $\mathtt{R}_3$, qui sera elle-même adaptée et retournée avant $\mathtt{R}_2$ pour $\mathtt{Q}_{ex}$ dans TAAABLE.

5.3.2 Conclusion

Dans ce chapitre, nous avons montré comment une communauté en ligne peut aider à représenter la gradualité des classes de l'ontologie en acquérant différents avis à propos des relations

de subsomption de l'ontologie. Dans notre travail, représenter la gradualité d'une classe `A` revient à discriminer les sous-classes $\texttt{B}_i$ d'une classe `A` par rapport à `A` grâce à la typicalité des sous-classes $\texttt{B}_i$ de `A`. La typicalité d'une sous-classe $\texttt{B}_i$ par rapport à `A` est représentée par un degré de typicalité acquis grâce à des questionnaires en ligne.

Nous avons proposé dans un premier temps d'utiliser le degré de typicalité pour raffiner l'ontologie du domaine en réorganisant une hiérarchie grâce à différente méthodes : la méthode des seuils, la méthode des k-moyennes et la méthode des niveaux. Ces trois méthodes permettent d'ajouter des classes intermédiaires sous les classes de l'ontologie afin de détailler d'avantage le processus de remémoration du système de RÀPC.

Nous avons proposé dans un deuxième temps de prendre en compte la typicalité dans le moteur de RÀPC durant la phase de remémoration pour sélectionner les cas les plus typiques par rapport à la requête afin de les adapter et de les retourner en premiers.

Dans ce chapitre, nous avons représenté la gradualité des classes grâce à la typicalité. Une autre possibilité envisageable correspond à l'utilisation de la théorie des sous-ensembles flous. Au lieu de représenter une relation de subsomption entre une sous-classe et sa classe, en termes de tout ou rien, un degré de subsomption serait utilisé pour représenter la « force » de la relation de subsomption. Les différents degrés de subsomption pourraient alors être pris en compte lors de la généralisation du problème cible en pondérant le coût de généralisation afin de guider la remémoration des cas.

L'apport du raffinement de l'ontologie sera évaluée en section 6.2 du chapitre 6 en comparant le système eTaaable, notée $\text{eTaaable}_{\text{std}}$, qui n'utilise pas d'ontologie raffinée et trois différentes versions du système eTaaable utilisant une ontologie raffinée par l'une des 3 méthodes représentées : $\text{eTaaable}_{\text{raffS}}$ (« Raff » pour raffinée et « S » pour seuil), $\text{eTaaable}_{\text{raffK}}$ (« K » pour k-moyennes) et $\text{eTaaable}_{\text{raffN}}$ (« N » pour niveau). De plus, l'évaluation des trois différentes méthodes introduites pour raffiner l'ontologie sera aussi réalisée en comparant $\text{eTaaable}_{\text{raffS}}$, $\text{eTaaable}_{\text{raffK}}$ et $\text{eTaaable}_{\text{raffN}}$.

L'apport de la prise en compte de la typicalité dans le moteur d'un système de RÀPC sera évaluée en section 6.3 du chapitre 6 en comparant le système $\text{eTaaable}_{\text{std}}$ et le système eTaaable prenant en compte la typicalité dans son moteur, noté $\text{eTaaable}_{\text{TypM}}$ (« Typ » pour typicalité et « M » pour moteur).

Chapitre 6

Gestion de la typicalité dans eTaaable et évaluation de la prise en compte de la typicalité dans eTaaable

Sommaire

Ce chapitre illustre et évalue les apports, présentés dans chapitre 5, dans le système de RàPC eTaaable. La section 6.1 compare eTaaable avec et sans l'ontologie raffinée par la typicalité. La section 6.2 évalue l'apport du raffinement de l'ontologie par la typicalité. La section 6.3 compare eTaaable avec et sans prise en compte de la typicalité dans le moteur de eTaaable. La section 6.4 évalue l'apport de la prise en compte de la typicalité dans le moteur de Taaable.

6.1 Comparaison de eTaaable avec et sans raffinement de l'ontologie

Nous allons illustrer dans cette section les effets du raffinement de l'ontologie de aTaaable grâce à la typicalité à travers un exemple. Cet exemple compare $\text{eTaaable}_{\text{std}}$, la version standard

de eTaaable dans laquelle l'ontologie n'est pas raffinée grâce à la typicalité, avec $\text{eTaaable}_{\text{raffS}}$, une version de eTaaable dans laquelle l'ontologie de aTaaable est raffinée grâce à la typicalité par la méthode des seuils.

Soit $\texttt{Q}_{ex} = \texttt{Recette} \sqcap \texttt{Tarte} \sqcap \texttt{Pêche}$ une requête donnée en exemple. Le tableau 6.1 présente la base de cas de eTaaable réduite à 5 recettes avec leur index ; chaque recette est indexée par au moins un fruit à noyau [31].

$\texttt{R}_i$	index
$\texttt{R}_1$	Tarte ⊓ PâteÀTarte ⊓ Mangue ⊓ Caramel
$\texttt{R}_2$	Tarte ⊓ PâteÀTarte ⊓ Avocat ⊓ Crabe ⊓ Creme ⊓ CitronVert ⊓ Œuf
$\texttt{R}_3$	Tarte ⊓ PâteÀTarte ⊓ Abricot ⊓ Amande ⊓ Lait ⊓ Sucre ⊓ Œuf ⊓ Vanille
$\texttt{R}_4$	Tarte ⊓ PâteÀTarte ⊓ Cerise ⊓ Amande ⊓ SucreVanillé
$\texttt{R}_5$	Tarte ⊓ PâteÀTarte ⊓ Olive ⊓ Jambon ⊓ Œuf ⊓ CrèmeLiquide ⊓ Sel

Tableau 6.1 – 5 recettes de la base de cas de eTaaable contenant un ingrédient qui est un fruit à noyau.

$\texttt{Rép}_i$	généralisation	cas remémoré et adaptation
$\texttt{Rép}_1$	Γ_1 : Pêche ⇝ FruitÀNoyau	$\texttt{R}_1$: Mangue ⇝ Pêche
$\texttt{Rép}_2$	Γ_1 : Pêche ⇝ FruitÀNoyau	$\texttt{R}_2$: Avocat ⇝ Pêche
$\texttt{Rép}_3$	Γ_1 : Pêche ⇝ FruitÀNoyau	$\texttt{R}_3$: Abricot ⇝ Pêche
$\texttt{Rép}_4$	Γ_1 : Pêche ⇝ FruitÀNoyau	$\texttt{R}_4$: Cerise ⇝ Pêche
$\texttt{Rép}_5$	Γ_1 : Pêche ⇝ FruitÀNoyau	$\texttt{R}_5$: Olive ⇝ Pêche

Tableau 6.2 – Les réponses retournées par $\text{eTaaable}_{\text{std}}$, pour $\texttt{Q}_{ex}$, décrites chacune par leur identifiant $\texttt{Rép}_i$, la généralisation Γ_j par laquelle elle a été obtenue à la $j^{ème}$ étape de généralisation et le cas remémoré et son adaptation.

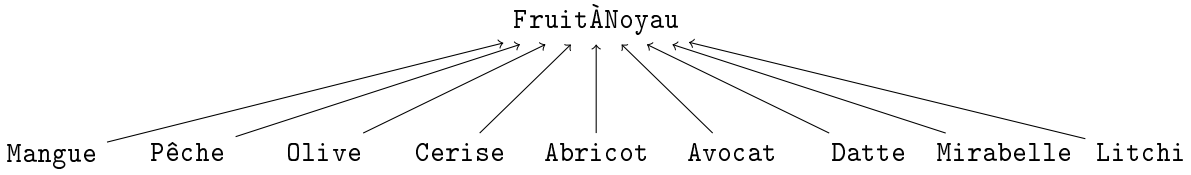


Figure 6.1 – Partie de la hiérarchie des aliments de eTaaable dont la racine est FruitÀNoyau.

Le tableau 6.2 montre les réponses retournées par $\text{eTaaable}_{\text{std}}$ pour $\texttt{Q}_{ex}$. $\text{eTaaable}_{\text{std}}$ utilise la partie non réorganisée de la hiérarchie des aliments dont la racine est FruitÀNoyau présentée dans la figure 6.1. Les 5 réponses sont retournées par $\text{eTaaable}_{\text{std}}$ à la première étape de généralisation où Pêche est généralisée en FruitÀNoyau. La dernière colonne du tableau 6.2 détaille pour chaque réponse le cas remémoré et son adaptation. Par exemple, $\texttt{Rép}_1$ consiste à remplacer Mangue par Pêche dans $\texttt{R}_1$, $\texttt{Rép}_3$ consiste à remplacer Abricot par Pêche dans $\texttt{R}_3$ et $\texttt{Rép}_5$ consiste à remplacer Olive par Pêche dans $\texttt{R}_5$.

31. Pour alléger le tableau, la classe Recette n'est pas spécifiée dans l'index de chaque recette.

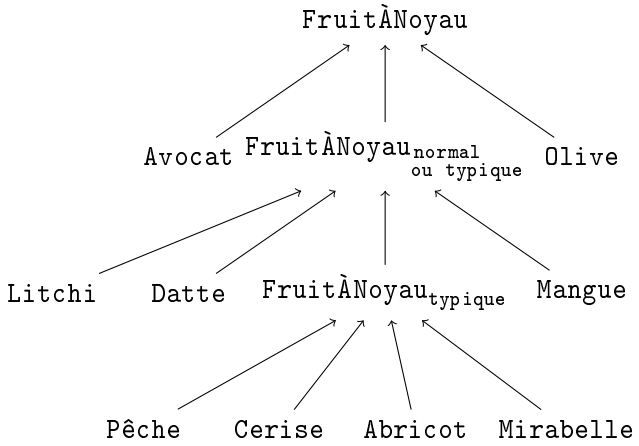


Figure 6.2 – Réorganisation de la hiérarchie des aliments dont la racine est `FruitÀNoyau` par la méthode des seuils.

La figure 6.2 correspond à la partie réorganisée de la hiérarchie des aliments dont la racine est `FruitÀNoyau` grâce à la typicalité par la méthode des seuils. D'après la figure 6.2, `Pêche` et `Abricot` sont des fruits à noyau typiques, `Mangue` est un fruit à noyau normal et `Olive` est un fruit à noyau atypique. Ainsi avec eTaaable$_{\text{std}}$:

- `Rép`$_1$ consiste à remplacer un fruit à noyau normal par un fruit à noyau typique ;
- `Rép`$_2$ consiste à remplacer un fruit atypique par un fruit typique ;
- `Rép`$_3$ consiste à remplacer un fruit à noyau typique par un fruit à noyau typique ;
- `Rép`$_4$ consiste à remplacer un fruit typique par un fruit typique ;
- `Rép`$_5$ consiste à remplacer un fruit à noyau atypique par un fruit à noyau atypique.

Comme les 5 réponses sont retournées grâce à la même étape de généralisation, ces 5 réponses sont retournées à l'utilisateur à classement égal.

`Rép`$_j$	généralisation	cas remémoré et adaptation
`Rép`$_3$	Γ_1 : `Pêche` $\rightsquigarrow$ `FruitÀNoyauTypique`	R_3 : `Abricot` $\rightsquigarrow$ `Pêche`
`Rép`$_4$	Γ_1 : `Pêche` $\rightsquigarrow$ `FruitÀNoyauTypique`	R_4 : `Cerise` $\rightsquigarrow$ `Pêche`
`Rép`$_1$	Γ_2 : `Pêche` $\rightsquigarrow$ `FruitÀNoyauNormal`	R_1 : `Mangue` $\rightsquigarrow$ `Pêche`
`Rép`$_2$	Γ_3 : `Pêche` $\rightsquigarrow$ `FruitÀNoyau`	R_2 : `Avocat` $\rightsquigarrow$ `Pêche`
`Rép`$_5$	Γ_3 : `Pêche` $\rightsquigarrow$ `FruitÀNoyau`	R_5 : `Olive` $\rightsquigarrow$ `Pêche`

Tableau 6.3 – Les réponses retournées par eTaaable$_{\text{raffS}}$, pour $\mathtt{Q}_{ex}$, décrites chacune par leur index `Rép`$_j$, la généralisation Γ_k par laquelle il a été obtenu à la $k^{ème}$ étape de généralisation et le cas remémoré et son adaptation.

Le tableau 6.3 montre les réponses retournées par eTaaable$_{\text{raffS}}$ pour $\mathtt{Q}_{ex}$. eTaaable$_{\text{raffS}}$ retourne les 5 mêmes réponses que eTaaable$_{\text{std}}$ mais à des étapes de généralisation différentes. Selon la figure 6.2, `Pêche` est un fruit à noyau typique qui est donc, en premier, généralisée en `FruitÀNoyauTypique` et permet de remémorer des recettes contenant des fruits à noyau typiques : R_3 indexée par `Abricot` est adaptée en remplaçant `Abricot` par `Pêche` et R_4 indexée par `Cerise` est adaptée en remplaçant `Cerise` par `Pêche`. À la seconde étape de généralisation, `Pêche` est généralisée en `FruitÀNoyauNormal`, permettant de retrouver R_1 qui est adaptée en remplaçant

Mangue par Pêche. Enfin, le système proposera R_2 adaptée en remplaçant Avocat par Pêche et R_5 adaptée en remplaçant Olive par Pêche à la troisième étape de généralisation où Pêche est généralisée en FruitÀNoyau. Ainsi, au lieu de retourner les réponses $Rép_1$, $Rép_2$, $Rép_3$, $Rép_4$ et $Rép_5$ à classement égal comme eTaaable$_{std}$, eTaaable$_{raffS}$ retourne en premier les réponses $Rép_3$ et $Rép_4$, puis la réponse $Rép_1$ et enfin les réponses $Rép_2$ et $Rép_5$.

6.2 Évaluation de la réorganisation de l'ontologie grâce à la typicalité

6.2.1 Hypothèses

Afin d'évaluer l'apport du raffinement de l'ontologie utilisée par un système de RàPC grâce à la typicalité, nous posons une hypothèse globale (H) : le raffinement de l'ontologie d'un système de RàPC grâce à la typicalité améliore les réponses du système.

Les trois méthodes pour raffiner l'ontologie d'un système de RàPC grâce à la typicalité, seront également évaluées, selon les hypothèses introduites dans la section 5.2.3 du chapitre 5 que nous rappelons ci-dessous.

Les hypothèses concernant le raffinement de l'ontologie grâce à la méthode des seuils sont :

(H1) avec une requête composée d'une sous-classe typique B_1 de A, les réponses pour lesquelles le cas remémoré est indexé par une sous-classe typique B_2 de A sont meilleures que les réponses pour lesquelles le cas remémoré est indexé par une sous-classe normale B_3 de A. Ainsi, les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe typique B_2 de A, indexant le cas remémoré, en $A_{typique}$ et spécialiser $A_{typique}$ en la sous-classe typique B_1 de A qui compose la requête sont meilleures que les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe normale B_3 de A, indexant le cas remémoré, en $A_{\substack{normal \\ ou\ typique}}$ et spécialiser $A_{\substack{normal \\ ou\ typique}}$ en la sous-classe typique B_1 de A qui compose la requête.

(H2) avec une requête composée d'une sous-classe typique B_1 de A, les réponses pour lesquelles le cas remémoré est indexé par une sous-classe typique ou normale B_2 de A sont meilleures que les réponses pour lesquelles le cas remémoré est indexé par une sous-classe atypique B_3 de A. Ainsi, les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe typique ou normale B_2 de A, indexant le cas remémoré, en $A_{\substack{normal \\ ou\ typique}}$ et spécialiser $A_{\substack{normal \\ ou\ typique}}$ en la sous-classe typique B_1 de A qui compose la requête sont meilleures que les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe atypique B_3 de A, indexant le cas remémoré, en A et spécialiser A en la sous-classe typique B_1 de A qui compose la requête.

(H3) avec une requête composée d'une sous-classe normale B_1 de A, les réponses pour lesquelles le cas remémoré est indexé par une sous-classe typique ou normale B_2 de A sont meilleures que les réponses pour lesquelles le cas remémoré est indexé par une sous-classe atypique B_3 de A. Ainsi, les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe typique ou normale B_2 de A, indexant le cas remémoré, en $A_{\substack{normal \\ ou\ typique}}$ et spécialiser $A_{\substack{normal \\ ou\ typique}}$ en la sous-classe normale B_1 de A qui compose la requête sont meilleures que les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe atypique B_3 de A, indexant le cas remémoré, en A et spécialiser A en la sous-classe normale B_1 de A qui compose la requête.

L'hypothèse concernant la réorganisation de l'ontologie grâce à la méthode des k-moyennes et selon la méthode des niveaux est :

(H4) avec une requête composée d'une sous-classe $\mathtt{B}_1$ subsumée par $\mathtt{A}_i$, les réponses pour lesquelles le cas remémoré est indexé par une autre sous-classe $\mathtt{B}_2$ de $\mathtt{A}_i$ (avec $\models_{BC} \mathtt{B}_2 \sqsubseteq \mathtt{A}_i$) sont meilleures que les réponses pour lesquelles le cas remémoré est indexé par une sous-classe $\mathtt{B}_3$ de $\mathtt{A}_{i-j}$, avec $i \geqslant 1$ et $0 < j < i$. Ainsi, les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe $\mathtt{B}_2$ de $\mathtt{A}$, indexant le cas remémoré, en $\mathtt{A}_i$ et spécialiser $\mathtt{A}_i$ en la sous-classe $\mathtt{B}_1$ de $\mathtt{A}_i$ qui compose la requête sont meilleures que les réponses dont l'adaptation du cas remémoré consiste à généraliser la sous-classe $\mathtt{B}_3$ de $\mathtt{A}$, indexant le cas remémoré, en $\mathtt{A}_{i-j}$ et spécialiser $\mathtt{A}_{i-j}$ en la sous-classe $\mathtt{B}_1$ de $\mathtt{A}_i$ qui compose la requête.

6.2.2 Méthode de comparaison

Pour valider toutes ces hypothèses, quatre versions de eTaaable sont comparées :

- eTaaable$_{\text{std}}$: la version standard de eTaaable qui utilise l'ontologie initiale du domaine contenue dans aTaaable.
- eTaaable$_{\text{raffS}}$: la version de eTaaable qui utilise l'ontologie initiale raffinée grâce à la typicalité par la méthode des seuils.
- eTaaable$_{\text{raffK}}$: la version de eTaaable qui utilise l'ontologie initiale raffinée grâce à la typicalité par la méthode des k-moyennes.
- eTaaable$_{\text{raffN}}$: la version de Taaable qui utilise l'ontologie initiale raffinée grâce à la typicalité par la méthode des niveaux.

(H) sera validée si eTaaable$_{\text{raffS}}$, eTaaable$_{\text{raffK}}$ et eTaaable$_{\text{raffN}}$ retourne de meilleures réponses que eTaaable$_{\text{std}}$.

(H1) sera validée si les réponses des quatre systèmes sont meilleures lorsqu'un ingrédient correspondant à une sous-classe typique est remplacé par un ingrédient correspondant à une sous-classe typique que lorsqu'un ingrédient correspondant à une sous-classe normale est remplacé par un ingrédient correspondant à une sous-classe typique.

(H2) sera validée si les réponses des quatre systèmes sont meilleures lorsqu'un ingrédient correspondant à une sous-classe typique ou normale est remplacé par un ingrédient correspondant à une sous-classe typique que lorsqu'un ingrédient correspondant à une sous-classe atypique est remplacé par un ingrédient correspondant à une sous-classe typique.

(H3) sera validée si les réponses des quatre systèmes sont meilleures lorsqu'un ingrédient correspondant à une sous-classe typique ou normale est remplacé par un ingrédient correspondant à une sous-classe normale que lorsqu'un ingrédient correspondant à une sous-classe atypique est remplacé par un ingrédient correspondant à une sous-classe normale.

(H4) sera validée si les réponses des quatre systèmes sont meilleures lorsqu'un ingrédient correspondant à une sous-classe de $\mathtt{A}_i$ est remplacé par un ingrédient correspondant à une sous-classe de $\mathtt{A}_i$ que lorsqu'un ingrédient correspondant à une autre sous-classe de $\mathtt{A}_{i-j}$, avec $i \geqslant 1$ et $j < i$, est remplacé par un ingrédient correspondant à une sous-classe de $\mathtt{A}_i$.

De plus, la comparaison de eTaaable$_{\text{raffS}}$, eTaaable$_{\text{raffK}}$ et eTaaable$_{\text{raffN}}$ permettra de sélectionner la méthode de raffinement de l'ontologie donnant les meilleures réponses.

	id	Requêtes
Requêtes sur les fruits à noyau	1	`Tarte` $\sqcap$ `Abricot`
	2	`Tarte` $\sqcap$ `Mirabelle`
	3	`Tarte` $\sqcap$ `Pêche`
	4	`Tarte` $\sqcap$ `Cerise`
	5	`Tarte` $\sqcap$ `Datte`
	6	`Tarte` $\sqcap$ `Mangue`
	7	`Tarte` $\sqcap$ `Litchi`
Requêtes sur les fruits à pépins	8	`Tarte` $\sqcap$ `Pomme`
	9	`Tarte` $\sqcap$ `Poire`
	10	`Tarte` $\sqcap$ `Raisin`
	11	`Tarte` $\sqcap$ `Citron`
	12	`Tarte` $\sqcap$ `Melon`
	13	`Tarte` $\sqcap$ `Orange`
	14	`Tarte` $\sqcap$ `Potiron`
	15	`Tarte` $\sqcap$ `Tomate`
Requêtes sur les baies	16	`Tarte` $\sqcap$ `Framboise`
	17	`Tarte` $\sqcap$ `Mûre`
	18	`Tarte` $\sqcap$ `Myrtille`
	19	`Tarte` $\sqcap$ `Groseille`
	20	`Tarte` $\sqcap$ `Fraise`

Tableau 6.4 – Ensemble des requêtes du plan de test. Les requêtes 1 à 7 concernent des requêtes sur des fruits à noyau, les requêtes 8 à 15 concernent des requêtes sur des fruits à pépins et les requêtes 16 à 20 concernent des requêtes sur des baies.

6.2.3 Construction du plan de test pour l'évaluation

Comme pour une partie du plan de test du chapitre 4, nous nous focalisons sur les tartes dans cette évaluation. Chaque requête est composée de la classe `Tarte` et d'une sous-classe d'une des classes `FruitÀPépins`, `FruitÀNoyau` ou `Baie` qui sont les classes qui ont été réorganisées dans la hiérarchie des aliments de aTaaable. Dans le but de valider les hypothèses (H1), (H2), (H3), et (H4), les sous-classes de `FruitÀPépins`, `FruitÀNoyau` et `Baie` qui composent les requêtes sont :

- toutes les sous-classes de `FruitÀNoyau`, `FruitÀPépins` et `Baie` classées comme typiques ou normales selon la méthode des seuils ;
- toutes les sous-classes $\mathtt{B}_i$ de `FruitÀNoyau`, `FruitÀPépins` et `Baie` telles que $\mathtt{B}_i \sqsubseteq \mathtt{FruitÀNoyau} \notin \mathtt{CD}$, $\mathtt{B}_i \sqsubseteq \mathtt{FruitÀPépins} \notin \mathtt{CD}$ et $\mathtt{B}_i \sqsubseteq \mathtt{Baie} \notin \mathtt{CD}$ dans la hiérarchie des aliments réorganisée grâce à la méthode des k-moyennes ou des niveaux.

21 sous-classes de `FruitÀNoyau`, `FruitÀPépins` et `Baie` correspondent à ces critères, cependant, nous rejetons la sous-classe `Poivron` pour les raisons qui vont suivre. D'après la figure 6.5 qui présente la partie de hiérarchie des aliments dont la racine est `FruitÀPépins`, réorganisée par la méthode des niveaux, comme $\mathtt{Poivron} \sqsubseteq \mathtt{FruitÀPépins} \notin \mathtt{CD}$, la sous-classe `Poivron` fait partie des 21 sous-classes. De plus, d'après cette figure, $\mathtt{Poivron} \sqsubseteq \mathtt{FruitÀPépins}_1 \in \mathtt{CD}$, $\mathtt{Concombre} \sqsubseteq \mathtt{FruitÀPépins} \in \mathtt{CD}$ et il n'existe pas de classes `C` telle que $\mathtt{C} \sqsubseteq \mathtt{FruitÀPépins} \in \mathtt{CD}$ et $\mathtt{C} \not\equiv \mathtt{Concombre}$. Ainsi, la sous-classe `Poivron` devrait être utilisée pour tester si les réponses dont l'adaptation consiste à remplacer un ingrédient qui n'est pas un concombre par du poivron

sont meilleures que les réponses dont l'adaptation consiste à remplacer du concombre par du poivron. Cependant, comme aucune recette n'est indexée par `Concombre`, il n'est pas possible de le tester et ainsi la sous-classe `Poivron` n'est pas utilisée pour la construction des requêtes. Par conséquent, seulement 20 sous-classes de `FruitÀNoyau`, `FruitÀPépins` et `Baie` sont utilisées pour construire les requêtes. 20 requêtes composées chacune d'une de ces 20 sous-classes ont été construites. Le tableau 6.4 [32] contient l'ensemble des requêtes de tartes du plan de test.

6.2.4 Préparation des données

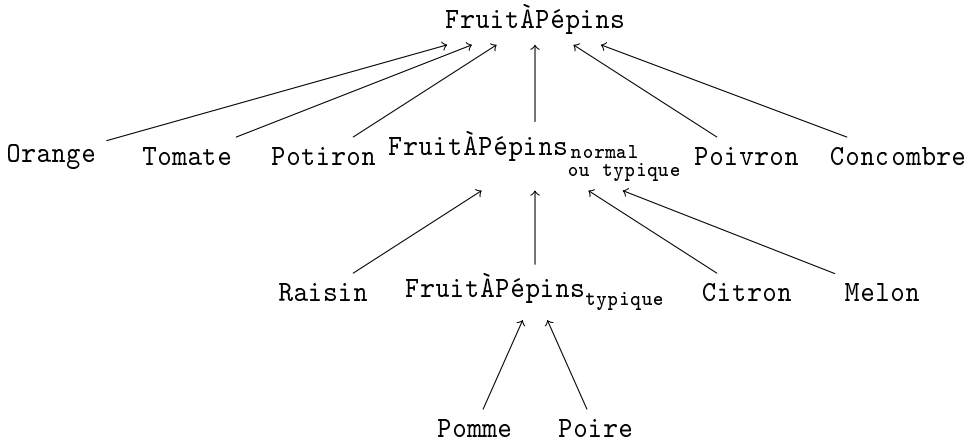


FIGURE 6.3 – Réorganisation de la hiérarchie des aliments dont la racine est `FruitÀPépins` par la méthode des seuils.

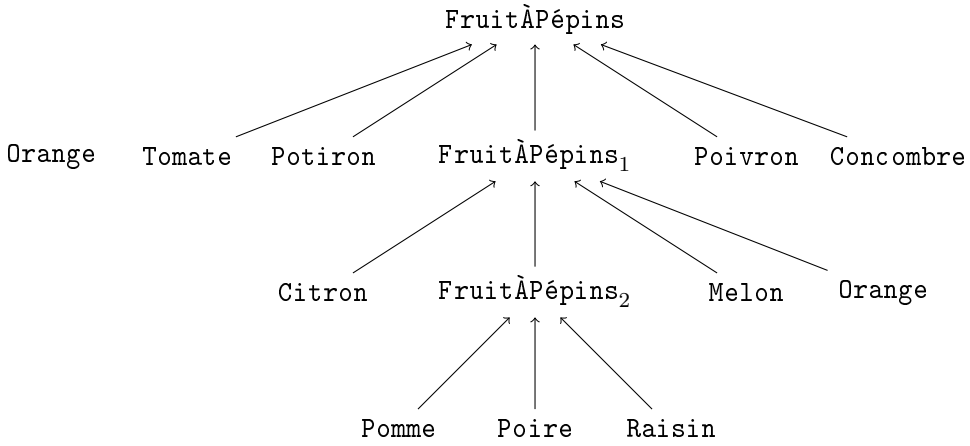


FIGURE 6.4 – Réorganisation de la hiérarchie des aliments dont la racine est `FruitÀPépins` par la méthode des k-moyennes.

32. Pour alléger les différents tableaux, la classe `Recette` n'est pas spécifiée pour chaque requête.

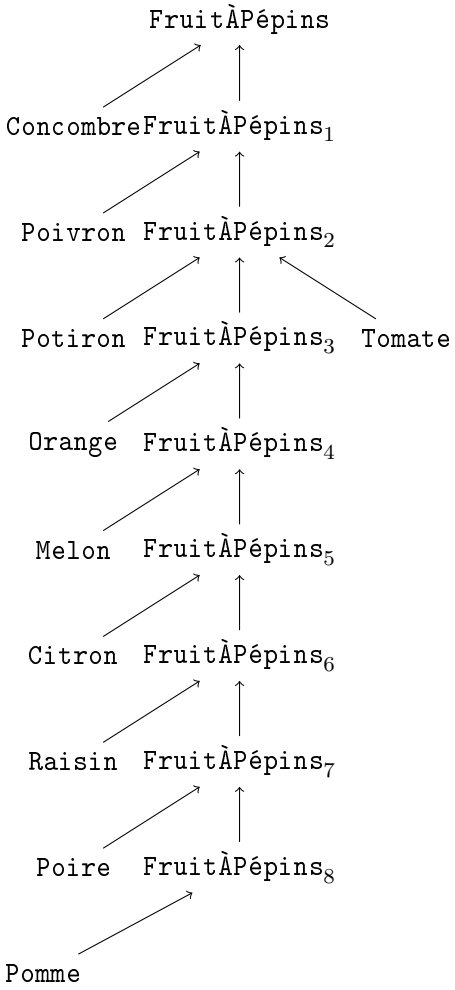


Figure 6.5 – Réorganisation de la hiérarchie des aliments dont la racine est `FruitÀPépins` par la méthode des niveaux.

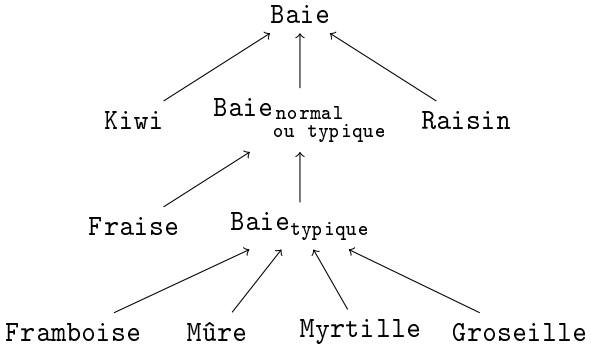


Figure 6.6 – Réorganisation de la hiérarchie des aliments dont la racine est `Baie` par la méthode des seuils.

Comme dans la section 5.2.3 du chapitre 5, les classes de `FruitÀNoyau`, `FruitÀPépins` et `Baie` de eTaaable$_{\text{raffS}}$, eTaaable$_{\text{raffK}}$ et eTaaable$_{\text{raffN}}$ ont été réorganisées grâce à l'acquisition des degrés de typicalité présentés en section 5.2.3. Pour eTaaable$_{\text{raffS}}$, 6 classes ont été ajoutées, pour eTaaable$_{\text{raffK}}$, 5 classes ont été ajoutées et pour eTaaable$_{\text{raffN}}$, 18 classes ont été ajoutées.

La partie de hiérarchie des aliments de aTaaable dont la racine est `FruitÀPépins` a été réorganisée par la méthode des seuils, par la méthode des k-moyennes et par la méthode des niveaux, respectivement représentées dans la figure 6.3, la figure 6.4 et la figure 6.5.

La partie de hiérarchie des aliments de aTaaable dont la racine est `Baie` a été réorganisée par la méthode des seuils, par la méthode des k-moyennes et par la méthode des niveaux, respectivement représentées dans la figure 6.6, la figure 6.7 et la figure 6.8.

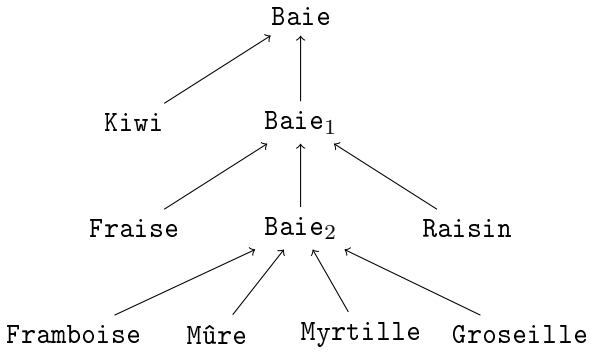


Figure 6.7 – Réorganisation de la hiérarchie des aliments dont la racine est `Baie` par la méthode des k-moyennes.

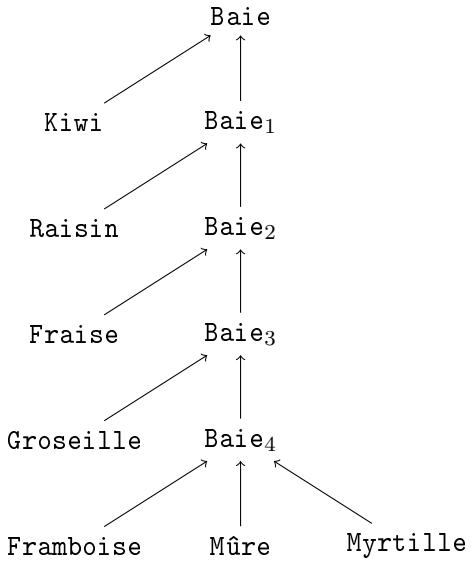


Figure 6.8 – Réorganisation de la hiérarchie des aliments dont la racine est `Baie` par la méthode des niveaux.

6.2.5 Analyse brute des résultats retournés par les systèmes

Les 5 premières réponses retournées par chacun des quatre systèmes ont été calculées pour les 20 requêtes. Chaque système a retourné $20 \times 5 = 100$ réponses. Au total, les quatre systèmes ont retourné 193 réponses différentes, chacune ayant été évaluée par au moins 3 utilisateurs, conformément à la méthode d'évaluation présentée en section 2.5 du chapitre 2.

Le tableau 6.5 présente le nombre de réponses communes entre les quatre systèmes. Les systèmes eTaaable$_{\text{raffS}}$, eTaaable$_{\text{raffK}}$ et eTaaable$_{\text{raffN}}$ ont retourné des réponses similaires tandis que eTaaable$_{\text{std}}$ a retourné plus de la moitié de réponses différentes par rapport aux autres systèmes.

Les réponses pour les requêtes 5, 13 et 14 sont identiques pour tous les systèmes car la généralisation de ces deux requêtes ne fait pas intervenir une des trois classes `FruitÀNoyau`, `FruitÀPépins` et `Baie`. Pour la requête 5, `Datte` est généralisée en `FruitSec`, la généralisation la moins coûteuse. De même, pour la requête 13, `Orange` est généralisée en `Agrume` et pour la requête 14, `Potiron` est généralisée en `Courge`

	eTaaable$_{\text{std}}$	eTaaable$_{\text{raffS}}$	eTaaable$_{\text{raffK}}$	eTaaable$_{\text{raffN}}$
eTaaable$_{\text{std}}$	100	46	45	41
eTaaable$_{\text{raffS}}$	×	100	89	91
eTaaable$_{\text{raffK}}$	×	×	100	80
eTaaable$_{\text{raffN}}$	×	×	×	100

Tableau 6.5 – Nombre de réponses communes entre les quatre systèmes où chaque système a retourné 100 réponses pour les requêtes de tartes.

6.2.6 Analyse des résultats de l'expérimentation

La moyenne et la médiane des scores de satisfaction des 5 premières réponses retournées pour les 20 requêtes de tartes par les quatre systèmes ont été calculées et sont présentées dans le tableau 6.6.

Système	Moyenne	Médiane
eTaaable$_{\text{std}}$	0,33	0,42
eTaaable$_{\text{raffS}}$	0,70	1,03
eTaaable$_{\text{raffK}}$	0,66	0,99
eTaaable$_{\text{raffN}}$	0,72	1,03

Tableau 6.6 – Moyennes et médianes des scores de satisfaction pour les réponses retournées par les quatre systèmes.

Pour l'ensemble des requêtes, nous observons que c'est eTaaable$_{\text{raffN}}$ qui satisfait le mieux les utilisateurs avec une moyenne de 0,72. Cependant, la moyenne de eTaaable$_{\text{raffN}}$ est proche de eTaaable$_{\text{raffS}}$ qui a une moyenne de 0,70 et de eTaaable$_{\text{raffK}}$ qui a une moyenne de 0,66. Avec une différence entre 0,33 et 0,39 avec les autres systèmes, la moyenne des scores de satisfaction de eTaaable$_{\text{std}}$ est de 0,33. Les systèmes eTaaable$_{\text{raffS}}$ et eTaaable$_{\text{raffN}}$ ont la meilleure médiane des scores de satisfaction, 1,05, très similaire à la médiane des scores de satisfaction de eTaaable$_{\text{raffK}}$ qui est de 0,99. Avec une différence entre 0,57 et 0,61 avec les autres systèmes, la médiane des scores de satisfaction est de 0,42.

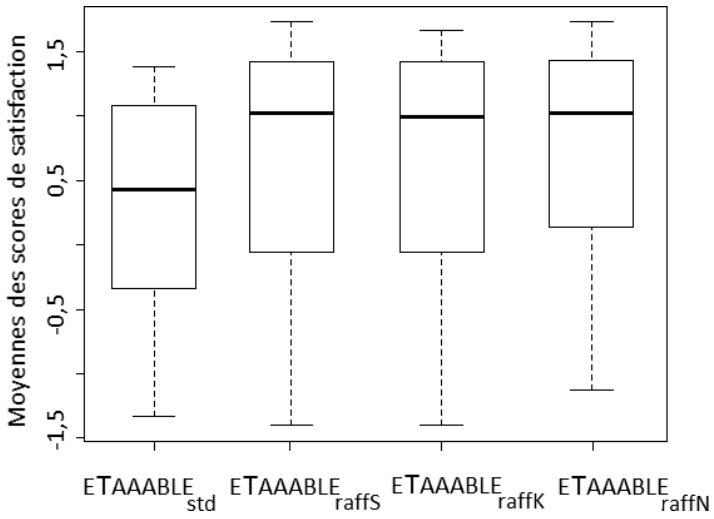


FIGURE 6.9 – Diagrammes en boîtes des moyennes des scores de satisfaction obtenus par les 5 premières réponses retournées par les quatre systèmes pour l'ensemble des 20 requêtes.

id	Requête	ET_{std}	ET_{raffS}	ET_{raffK}	ET_{raffN}
1	Tarte ⊓ Abricot	1,30	**1,46**	**1,46**	**1,46**
2	Tarte ⊓ Mirabelle	1,36	**1,39**	**1,39**	**1,39**
3	Tarte ⊓ Pêche	0,93	**1,53**	**1,53**	1,47
4	Tarte ⊓ Cerise	0,45	**1,43**	**1,43**	**1,43**
5	Tarte ⊓ Datte	0,41	0,41	0,41	0,41
6	Tarte ⊓ Mangue	−0,27	−0,20	−0,93	**−0,13**
7	Tarte ⊓ Litchi	−0,85	**−0,82**	−0,90	−0,89
8	Tarte ⊓ Pomme	0,08	**1,56**	1,51	**1,56**
9	Tarte ⊓ Poire	−1,00	**1,73**	1,67	**1,73**
10	Tarte ⊓ Raisin	−0,67	**−0,07**	**−0,07**	0,53
11	Tarte ⊓ Citron	0,74	0,64	**0,84**	0,64
12	Tarte ⊓ Melon	−1,33	−1,40	−1,40	**−1,13**
13	Tarte ⊓ Orange	1,27	1,27	1,27	1,27
14	Tarte ⊓ Potiron	−0.40	−0.40	−0.40	−0.40
15	Tarte ⊓ Tomate	−0.20	−0.05	−0.05	−0.45
16	Tarte ⊓ Framboise	1,38	1,35	1,35	**1,43**
17	Tarte ⊓ Mûre	**1,15**	1,08	1,08	1,08
18	Tarte ⊓ Myrtille	1,02	**1,41**	**1,41**	**1,41**
19	Tarte ⊓ Groseille	0,27	**0,67**	**0,67**	**0,67**
20	Tarte ⊓ Fraise	**0,97**	**0,97**	0,90	**0,97**
		0,33	0,70	0,66	**0,72**

TABLEAU 6.7 – Moyennes des scores de satisfactions pour ET_{std} =ETAAABLE$_{\text{std}}$, ET_{raffS} =ETAAABLE$_{\text{raffS}}$, ET_{raffK} =ETAAABLE$_{\text{raffK}}$ et ET_{raffN} =ETAAABLE$_{\text{raffN}}$. Les scores entre gras indiquent les scores les plus élevés pour une requête.

La figure 6.9 présente les diagrammes en boîte des moyennes des scores de satisfaction pour les réponses retournées par les quatre systèmes, pour l'ensemble des requêtes. En outre des différences de médiane, nous pouvons observer que seulement 33% des moyennes des scores de satisfaction de eTaaable$_{\text{std}}$ sont égales ou supérieures à 1,0 alors que 52% des moyennes des scores de satisfaction de eTaaable$_{\text{raffS}}$, eTaaable$_{\text{raffK}}$ et eTaaable$_{\text{raffN}}$ sont au-dessus de 1,0.

Le tableau 6.7 présente les moyennes des scores de satisfaction pour les réponses retournées par les quatre systèmes pour chacune des 20 requêtes. Le tableau contient les requêtes et leur identifiant, et la moyenne des scores de satisfaction des réponses retournées par chaque système.

Pour 18 des 20 requêtes, c'est un des trois systèmes eTaaable$_{\text{raffS}}$, eTaaable$_{\text{raffK}}$ et eTaaable$_{\text{raffN}}$ qui a la meilleure moyenne des scores de satisfaction. Dans un seul cas (la requête 17), eTaaable$_{\text{std}}$ a la meilleure moyenne des scores de satisfaction. Cependant, la différence de la moyenne des scores de satisfaction entre les trois autres systèmes est négligeable (0,07).

eTaaable$_{\text{raffS}}$ retourne les meilleures réponses pour 11 requêtes et pour 10 requêtes composées d'une sous-classe typique, eTaaable$_{\text{raffS}}$ retourne les meilleures réponses pour 8 requêtes.

eTaaable$_{\text{raffK}}$ retourne les meilleures réponses pour 8 requêtes et pour 11 requêtes composées d'une sous-classe de l'ensemble des sous-classes les plus typiques, eTaaable$_{\text{raffK}}$ retourne les meilleures réponses pour 7 requêtes.

eTaaable$_{\text{raffN}}$ retourne les meilleures réponses pour 11 requêtes et pour 7 requêtes composées d'une sous-classe de l'ensemble des sous-classes les plus typiques, eTaaable$_{\text{raffN}}$ retourne les meilleures réponses pour 5 requêtes.

Requête	$T \rightsquigarrow T$	$T\,ou\,N \rightsquigarrow T$	$A \rightsquigarrow T$
`Tarte ⊓ Abricot`	**1,46**	1,33	−1.83
`Tarte ⊓ Mirabelle`	1,25	**1,28**	−1,67
`Tarte ⊓ Pêche`	**1,47**	1.46	−2
`Tarte ⊓ Cerise`	**1,23**	1,25	−1,83
`Tarte ⊓ Pomme`	**1,56**	1,11	−1,92
`Tarte ⊓ Poire`	**1,62**	1,03	−1,87
`Tarte ⊓ Framboise`	1,39	**1,45**	1,08
`Tarte ⊓ Mûre`	**1,58**	1,12	−0,92
`Tarte ⊓ Myrtille`	**1,53**	1,13	0,00
`Tarte ⊓ Groseille`	**0,75**	0,43	0,00
	1,38	1,06	−1,10

Tableau 6.8 – Moyennes des scores de satisfaction pour les réponses des requêtes composées d'un ingrédient correspondant à une classe typique liée au type d'adaptation : remplacer un ingrédient correspondant à une sous-classe typique par un autre ingrédient correspondant à une sous-classe typique de la requête (colonne « $T \rightsquigarrow T$ »), remplacer un ingrédient correspondant à une sous-classe typique ou normale par un ingrédient correspondant à une sous-classe typique de la requête (colonne « T *ou* $N \rightsquigarrow T$ »), et remplacer un ingrédient correspondant à une sous-classe atypique par un ingrédient correspondant à une sous-classe typique de la requête (colonne « $A \rightsquigarrow T$ »).

Le tableau 6.8 donne la moyenne des scores de satisfaction pour les réponses des requêtes composées d'une sous-classe typique, selon la méthode des seuils, liée au type d'adaptation :

- La $2^{ème}$ colonne (« $T \rightsquigarrow T$ ») donne la moyenne des scores de satisfaction des réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe typique avec un ingrédient correspondant à une autre sous-classe typique.
- La $3^{ème}$ colonne (« T *ou* $N \rightsquigarrow T$ ») donne la moyenne des scores de satisfaction des réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe normale ou typique avec un ingrédient correspondant à une sous-classe typique.
- La $4^{ème}$ colonne (« $A \rightsquigarrow T$ ») donne la moyenne des scores de satisfaction des réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe atypique avec un ingrédient correspondant à une sous-classe typique.

Pour chaque requête, la moyenne en gras indique le type d'adaptation pour lequel la moyenne est la plus élevée.

Les réponses qui consistent à remplacer un ingrédient correspondant à une sous-classe typique avec un ingrédient correspondant à une autre sous-classe typique donnent de meilleures moyennes de scores de satisfaction pour 8 requêtes sur 10. De plus, ces types de réponses obtiennent la meilleure moyenne des scores de satisfaction, qui est de 1,38. Les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe typique ou normal avec un ingrédient typique sont meilleures ou égales pour les 10 requêtes par rapport aux réponses adaptant un ingrédient correspondant à une sous-classe atypique avec un ingrédient correspondant à une sous-classe typique.

Requête	T *ou* $N \rightsquigarrow N$	$A \rightsquigarrow N$
`Tarte ⊓ Mangue`	**0,98**	0,83
`Tarte ⊓ Litchi`	**−0,53**	−1,23
`Tarte ⊓ Citron`	**1,06**	−2
`Tarte ⊓ Melon`	**−0,87**	−1,17
`Tarte ⊓ Fraise`	**−1,10**	−1,23
	−0,09	−1,21

Tableau 6.9 – Moyennes des scores de satisfaction pour les réponses des requêtes composées d'un ingrédient correspondant à une classe normale liée au type d'adaptation : remplacer un ingrédient correspondant à une sous-classe typique ou normale par un ingrédient correspondant à une sous-classe normale de la requête (colonne « T *ou* $N \rightsquigarrow N$ »), et remplacer un ingrédient correspondant à une sous-classe atypique par un ingrédient correspondant à une sous-classe normale de la requête (colonne « $A \rightsquigarrow N$ »).

Le tableau 6.9 donne la moyenne des scores de satisfaction pour les réponses des requêtes composées d'une sous-classe normale, selon la méthode des seuils, liée au type d'adaptation :

- La $2^{ème}$ colonne (« T *ou* $N \rightsquigarrow N$ ») donne la moyenne des scores de satisfaction des réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe normale ou typique avec un ingrédient correspondant à une sous-classe normale.
- La $3^{ème}$ colonne (« $A \rightsquigarrow N$ ») donne la moyenne des scores de satisfaction des réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe atypique avec un ingrédient correspondant à une sous-classe normale.

Pour chaque requête, la moyenne en gras indique le type d'adaptation pour lequel la moyenne est la plus élevée. Les requêtes composées des sous-classes normales `Datte` et `Raisin` ne sont pas présentées car la requête `Tarte ⊓ Datte` est généralisée en `Tarte ⊓ FruitSec` (la classe

`FruitSec` n'a pas été réorganisée) et la requête `Tarte` ⊓ `Raisin` est généralisée en `Tarte` ⊓ `Baie` (`Raisin` est une sous-classe atypique et non pas normale de `Baie`). Les réponses qui consistent à remplacer un ingrédient correspondant à une sous-classe normale ou typique avec un ingrédient correspondant à une sous-classe normale donnent de meilleures moyennes des scores de satisfaction pour 5 requêtes sur 5. De plus, ces types de réponses obtiennent la meilleure moyenne des scores de satisfaction, qui est de −0.,09.

	Réorganisation par la méthode des k-moyennes			Réorganisation par la méthode des niveaux	
	$\mathtt{A}_i \rightsquigarrow \mathtt{A}_i$	$\mathtt{A}_{i-j} \rightsquigarrow \mathtt{A}_i$		$\mathtt{A}_i \rightsquigarrow \mathtt{A}_i$	$\mathtt{A}_{i-j} \rightsquigarrow \mathtt{A}_i$
`Tarte` ⊓ `Abricot`	**1,43**	1,00		**1,60**	1,17
`Tarte` ⊓ `Mirabelle`	**1,39**	1,17		**1,49**	1,21
`Tarte` ⊓ `Pêche`	**1,08**	0,56		**1,50**	0,87
`Tarte` ⊓ `Cerise`	**1,43**	−0,17		**0,43**	−0,17
`Tarte` ⊓ `Mangue`	−0,29	×		**−0,53**	−1,23
`Tarte` ⊓ `Litchi`	0,93	×		**−0,87**	−1,17
`Tarte` ⊓ `Pomme`	**1,53**	−0,89		×	0,72
`Tarte` ⊓ `Poire`	**1,66**	−1,00		**1,70**	0,62
`Tarte` ⊓ `Raisin`	−0,44	**−0,33**		−0,44	**−0,33**
`Tarte` ⊓ `Citron`	−0,53	×		−0,75	**1,18**
`Tarte` ⊓ `Melon`	−1,06	×		−1,06	**−1,00**
`Tarte` ⊓ `Orange`	1,33	×		1,33	×
`Tarte` ⊓ `Tomate`	−0,05	×		−0,05	**1,25**
`Tarte` ⊓ `Framboise`	**1,34**	1,33		**1,35**	1,33
`Tarte` ⊓ `Mûre`	**1,58**	0,86		**1,58**	0,86
`Tarte` ⊓ `Myrtille`	**1,53**	1,02		**1,53**	1,02
`Tarte` ⊓ `Groseille`	**0,67**	0,17		**0,67**	0,17
`Tarte` ⊓ `Fraise`	0,97	×		0,98	**1,00**
	0,87	0,34		**0,64**	0,42

Tableau 6.10 – Moyennes des scores de satisfaction des réponses des quatre systèmes, liée à la méthode de réorganisation de la hiérarchie des aliments et au type d'adaptation. `A` correspond à `FruitÀNoyau`, `FruitÀPépins` ou `Baie`. Les $2^{ème}$ et $4^{ème}$ colonnes $\mathtt{A}_i \rightsquigarrow \mathtt{A}_i$ montre pour chaque requête, la moyenne des scores de satisfaction des réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous classe $\mathtt{A}_i$ par un ingrédient correspondant à une autre sous-classe de $\mathtt{A}_i$. Les $3^{ème}$ et $5^{ème}$ colonnes $\mathtt{A}_{i-j} \rightsquigarrow \mathtt{A}_i$, avec $i \geqslant 1$ et $0 < j < i$, montre pour chaque requête, la moyenne des scores de satisfaction des réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous classe $\mathtt{A}_{i-j}$ par un ingrédient correspondant à une autre sous-classe de $\mathtt{A}_i$. Pour la $2^{ème}$ et la $3^{ème}$ colonne, $\mathtt{A}_i$ et $\mathtt{A}_{i-j}$ sont les classes de la hiérarchie des aliments réorganisée par la méthode des k-moyennes. Pour la $4^{ème}$ et la $5^{ème}$ colonne, $\mathtt{A}_i$ et $\mathtt{A}_{i-j}$ sont les classes de la hiérarchie des aliments réorganisée par la méthode des k-moyennes. Le signe × signifie qu'aucune réponse correspondant au critère de la colonne n'a été retournée par les quatre systèmes.

Le tableau 6.10 donne la moyenne des scores de satisfaction pour les réponses des quatre systèmes, liée à la méthode de réorganisation de la hiérarchie des aliments et au type d'adaptation. Pour la réorganisation par la méthode des k-moyennes et la réorganisation par la méthode des niveaux :

- la ligne $\mathtt{A}_i \rightsquigarrow \mathtt{A}_i$ correspond à la moyenne des scores de satisfaction dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe de $\mathtt{A}_i$ par un ingrédient correspondant à une autre sous-classe de $\mathtt{A}_i$.
- la ligne $\mathtt{A}_{i-j} \rightsquigarrow \mathtt{A}_i$ correspond à la moyenne des scores de satisfaction dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe de $\mathtt{A}_{i-j}$ par un ingrédient correspondant à une sous-classe de $\mathtt{A}_i$, avec $i \geqslant 1$ et $0 < j < i$.

`A` correspond à `FruitÀNoyau`, `FruitÀPépins` ou `Baie`. Pour la $2^{ème}$ et la $3^{ème}$ colonne, $\mathtt{A}_i$ et $\mathtt{A}_{i-j}$ sont les classes de la hiérarchie des aliments réorganisée par la méthode des k-moyennes. Pour la $4^{ème}$ et la $5^{ème}$ colonne, $\mathtt{A}_i$ et $\mathtt{A}_{i-j}$ sont les classes de la hiérarchie des aliments réorganisée par la méthode des k-moyennes.

Pour la réorganisation par la méthode des k-moyennes, seules 11 requêtes permettent de comparer les deux types d'adaptation. 7 requêtes ne donnent aucune réponse pour le type d'adaptation $\mathtt{A}_{i-j} \rightsquigarrow \mathtt{A}_i$. Pour la réorganisation par la méthode des niveaux, seules 16 requêtes permettent de comparer les deux types d'adaptation. 1 requête ne donne aucune réponse pour le type d'adaptation $\mathtt{A}_i \rightsquigarrow \mathtt{A}_i$ et 1 requête ne donne aucune réponse pour le type d'adaptation $\mathtt{A}_{i-j} \rightsquigarrow \mathtt{A}_i$.

Nous constatons que pour 10 requêtes sur 11, avec la hiérarchie des aliments réorganisée par la méthode des k-moyennes, les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous classe $\mathtt{A}_i$ par un ingrédient correspondant à une autre sous-classe de $\mathtt{A}_i$ sont meilleures. De plus, les réponses impliquant ce type d'adaptation pour la hiérarchie des aliments réorganisée par la méthode des k-moyennes obtiennent la meilleure moyenne qui est de 0,87, contre 0,34 pour les réponses de l'autre type d'adaptation. Pour 11 requêtes sur 16, avec la hiérarchie des aliments réorganisée par la méthode des niveaux, les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous classe $\mathtt{A}_i$ par un ingrédient correspondant à une autre sous-classe de $\mathtt{A}_i$ sont meilleures. De plus, les réponses impliquant ce type d'adaptation pour la hiérarchie des aliments réorganisée par la méthode des niveaux obtiennent la meilleure moyenne qui est de 0,64, contre 0,42 pour les réponses de l'autre type d'adaptation.

6.2.7 Validation des résultats

Grâce au test des rangs signés de Wilcoxson, nous pouvons conclure que les scores de satisfaction pour les réponses de eTaaable$_{\text{raffS}}$, eTaaable$_{\text{raffK}}$ et eTaaable$_{\text{raffN}}$ sont significativement meilleures que les scores de satisfaction pour les réponses de eTaaable$_{\text{std}}$:

- avec une p-valeur de $4{,}851.10^{-3}$ (inférieure à 0,05, disant que l'hypothèse nulle est rejetée), la différence entre la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{raffS}}$ et la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{std}}$ est significative ; eTaaable$_{\text{raffS}}$ retourne significativement de meilleures réponses que eTaaable$_{\text{std}}$.
- avec une p-valeur de $2{,}209.10^{-2}$ (inférieure à 0,05), la différence entre la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{raffK}}$ et la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{std}}$ est significative ; eTaaable$_{\text{raffK}}$ retourne significativement de meilleures réponses que eTaaable$_{\text{std}}$.
- avec une p-valeur de $7{,}021.10^{-3}$ (inférieure à 0,05), la différence entre la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{raffN}}$ et la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{std}}$ est significative ; eTaaable$_{\text{raffN}}$ retourne significativement de meilleures réponses que eTaaable$_{\text{std}}$.

Comme les réponses de eTaaable$_{\text{raffS}}$, eTaaable$_{\text{raffK}}$ et eTaaable$_{\text{raffN}}$ sont significativement meilleures que les réponses de eTaaable$_{\text{std}}$, l'hypothèse globale (H) est validée.

D'après le tableau 6.8, les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe typique par un ingrédient correspondant à une autre sous-classe typique sont meilleures que les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe normale par un ingrédient correspondant à une autre sous-classe typique. Cependant, avec une p-valeur de $1{,}54.10^{-1}$ (supérieure à 0,05), cette différence n'est pas significative : (H1) n'est pas validée.

Les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe typique ou normale avec un ingrédient correspondant à une sous-classe typique sont significativement meilleures que les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe atypique avec un ingrédient correspondant à une sous-classe typique avec une p-valeur de $2{,}961.10^{-3}$ (inférieure à 0,05) : (H2) est validée.

D'après le tableau 6.9, les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe typique ou normale par un ingrédient correspondant à une sous-classe normale sont meilleures que les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe atypique par un ingrédient correspondant à une sous-classe normale, avec une p-valeur de $2{,}9.10^{-2}$ (inférieure à 0,05) : (H3) est validée.

D'après le tableau 6.10 et la réorganisation de la hiérarchie des aliments par la méthode des k-moyennes, les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe de $\mathtt{A}_i$ par un ingrédient correspondant à une autre sous-classe de $\mathtt{A}_i$ sont significativement meilleures que les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe de $\mathtt{A}_i$ par un ingrédient correspondant à une sous-classe de $\mathtt{A}_j$, pour $j > i$, avec une p-valeur de $3{,}3346.10^{-3}$ (inférieure à 0,05) : (H4) est validée pour la méthode des k-moyennes. D'après le tableau 6.10 et la réorganisation de la hiérarchie des aliments par la méthode des niveaux, les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe de $\mathtt{A}_i$ par un ingrédient correspondant à une autre sous-classe de $\mathtt{A}_i$ sont significativement meilleures que les réponses dont l'adaptation consiste à remplacer un ingrédient correspondant à une sous-classe de $\mathtt{A}_i$ par un ingrédient correspondant à une sous-classe de $\mathtt{A}_j$, pour $j > i$, avec une p-valeur de $2{,}469.10^{-2}$ (inférieure à 0,05) : (H4) est validée pour la méthode des niveaux.

Par ailleurs, nous pouvons conclure qu'il n'a pas de différence significative entre les réponses des trois systèmes eTaaable$_{\text{raffS}}$, eTaaable$_{\text{raffK}}$ et eTaaable$_{\text{raffN}}$:

- Avec une p-valeur de $1{,}473.10^{-1}$ (supérieure à 0,05), la différence entre la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{raffS}}$ et la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{raffK}}$ n'est pas significative.
- Avec une p-valeur de $2{,}771.10^{-1}$ (supérieure à 0,05), la différence entre la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{raffN}}$ et la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{raffS}}$ n'est pas significative.
- Avec une p-valeur de $1{,}238.10^{-1}$ (supérieure à 0,05), la différence entre la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{raffN}}$ et la moyenne des scores de satisfaction des réponses de eTaaable$_{\text{raffK}}$ n'est pas significative.

6.3 Comparaison de eTaaable avec et sans prise en compte de la typicalité dans le moteur de eTaaable

Nous allons illustrer dans cette section les effets de la prise en compte de la typicalité dans le moteur d'un système de RàPC durant la phase de remémoration. Cet exemple compare les réponses de eTaaable$_{\text{std}}$, la version standard de Taaable dans laquelle la typicalité n'est pas prise en compte, avec les réponses de eTaaable$_{\text{TypM}}$, la version de eTaaable dans laquelle la typicalité est prise en compte dans le moteur.

Soit $\texttt{Q}_{ex}$ = `Recette` $\sqcap$ `Tarte` $\sqcap$ `Cerise` $\sqcap$ `Poire` une requête donnée en exemple. Dans cet exemple, nous nous intéressons à l'étape de généralisation où la fonction de généralisation correspond à Γ = `Cerise` $\rightsquigarrow$ `FruitÀNoyau` $\circ$ `Poire` $\rightsquigarrow$ `FruitÀPépins` et la requête généralisée correspond à $\Gamma(\texttt{Q}_{ex})$ = `Recette` $\sqcap$ `Tarte` $\sqcap$ `FruitÀNoyau` $\sqcap$ `FruitÀPépins`. Le tableau 6.11 présente 5 recettes remémorées pour $\Gamma(\texttt{Q}_{ex})$ et indexées par au moins un fruit à noyau et au moins un fruit à pépins.

Le tableau 6.12 montre les réponses retournées par eTaaable$_{\text{std}}$ pour $\texttt{Q}_{ex}$ avec la base de cas composée des 5 recettes présentées dans le tableau 6.11 [33]. Les 5 réponses sont retournées par eTaaable$_{\text{std}}$ à la première étape de généralisation où `Cerise` est généralisée en `FruitÀNoyau` et `Poire` est généralisée en `FruitÀPépins`. La dernière colonne du tableau 6.12 donne pour chaque réponse le cas remémoré, parmi les 5 cas du tableau 6.12, et son adaptation. Par exemple, dans le tableau 6.12, $\texttt{Rép}_1$ consiste à adapter $\texttt{R}_1$ en remplaçant `Abricot` par `Cerise` et `Pomme` par `Poire`.

$\texttt{R}_i$	index
$\texttt{R}_1$	`Tarte` $\sqcap$ `PâteÀTarte` $\sqcap$ `Abricot` $\sqcap$ `Pomme` $\sqcap$ `Creme` $\sqcap$ `Œuf` $\sqcap$ `Sucre`
$\texttt{R}_2$	`Tarte` $\sqcap$ `PâteÀTarte` $\sqcap$ `Avocat` $\sqcap$ `Citron` $\sqcap$ `CrèmeFraîche` $\sqcap$ `Tomate` $\sqcap$ `Œuf`
$\texttt{R}_3$	`Tarte` $\sqcap$ `PâteÀTarte` $\sqcap$ `Olive` $\sqcap$ `Citron` $\sqcap$ `Lardons` $\sqcap$ `CrèmeLiquide` $\sqcap$ `Œuf`
$\texttt{R}_4$	`Tarte` $\sqcap$ `PâteÀTarte` $\sqcap$ `Pêche` $\sqcap$ `Raisin` $\sqcap$ `Gélatine` $\sqcap$ `Sucre`
$\texttt{R}_5$	`Tarte` $\sqcap$ `PâteÀTarte` $\sqcap$ `Mangue` $\sqcap$ `Pomme` $\sqcap$ `Caramel`

Tableau 6.11 – 5 recettes de la base de cas de eTaaable contenant au moins un ingrédient qui est un fruit à noyau et au moins un ingrédient qui est un fruit à pépins.

$\texttt{Rép}_i$	cas remémoré et adaptation
$\texttt{Rép}_1$	$\texttt{R}_1$: `Abricot` $\rightsquigarrow$ `Cerise` $\circ$ `Pomme` $\rightsquigarrow$ `Poire`
$\texttt{Rép}_2$	$\texttt{R}_2$: `Avocat` $\rightsquigarrow$ `Cerise` $\circ$ `Citron` $\rightsquigarrow$ `Poire`
$\texttt{Rép}_3$	$\texttt{R}_3$: `Olive` $\rightsquigarrow$ `Cerise` $\circ$ `Citron` $\rightsquigarrow$ `Poire`
$\texttt{Rép}_4$	$\texttt{R}_4$: `Pêche` $\rightsquigarrow$ `Cerise` $\circ$ `Raisin` $\rightsquigarrow$ `Poire`
$\texttt{Rép}_5$	$\texttt{R}_5$: `Mangue` $\rightsquigarrow$ `Cerise` $\circ$ `Pomme` $\rightsquigarrow$ `Poire`

Tableau 6.12 – Réponses retournées par eTaaable$_{\text{std}}$ pour $\texttt{Q}_{ex}$. Chaque réponse est décrite par son identifiant, l'identifiant du cas remémoré, l'index du cas remémoré et l'adaptation du cas.

33. Pour alléger le tableau, la classe `Recette` n'est pas spécifiée dans l'index de chaque recette.

D'après le tableau 6.12 et le tableau 6.13 indiquant les degrés de typicalité des sous-classes des classes `FruitÀNoyau` et `FruitÀPépins`, avec eTaaable$_{\text{std}}$:

- `Rép`$_1$ consiste à remplacer, `Abricot`, un fruit au moins aussi typique que `Cerise` par rapport à `FruitÀNoyau` par `Cerise` et `Pomme`, un fruit au moins aussi typique que `Poire` par rapport à `FruitÀPépins` par `Poire` ;
- `Rép`$_2$ consiste à remplacer, `Avocat`, un fruit moins typique que `Cerise` par rapport à `FruitÀNoyau` par `Cerise` et `Citron`, un fruit moins typique que `Poire` par rapport à `FruitÀPépins` par `Poire` ;
- `Rép`$_3$ consiste à remplacer, `Olive`, un fruit moins typique que `Cerise` par rapport à `FruitÀNoyau` par `Cerise` et `Citron`, un fruit moins typique que `Poire` par rapport à `FruitÀPépins` par `Poire` ;
- `Rép`$_4$ consiste à remplacer, `Pêche`, un fruit au moins aussi typique que `Cerise` par rapport à `FruitÀNoyau` par `Cerise` et `Raisin`, un fruit moins typique que `Poire` par rapport à `FruitÀPépins` par `Poire` ;
- `Rép`$_5$ consiste à remplacer, `Mangue`, un fruit moins typique que `Cerise` par rapport à `FruitÀNoyau` par `Cerise` et `Pomme`, un fruit au moins aussi typique que `Poire` par rapport à `FruitÀPépins` par `Poire`.

Comme les 5 réponses sont retournées grâce à la même étape de généralisation, ces 5 réponses sont retournées à l'utilisateur à classement égal.

`FruitÀNoyau`	
`Abricot`	1,00
`Mirabelle`	1,00
`Pêche`	1,00
`Cerise`	0,92
`Datte`	0,42
`Litchi`	0,17
`Mangue`	0,08
`Avocat`	−0,17
`Olive`	−0,25

`FruitÀPépins`	
`Pomme`	0,67
`Poire`	0,50
`Raisin`	0,33
`Citron`	0,17
`Melon`	0,08
`Orange`	−0,17
`Potiron`	−0,58
`Tomate`	−0,58
`Poivron`	−0,67
`Concombre`	−1,00

Tableau 6.13 – Degrés de typicalité pour les sous-classes de la classe `FruitÀNoyau` et `FruitÀPépins`.

Pour eTaaable$_{\text{TypM}}$, les 5 recettes du tableau 6.11 sont, comme pour eTaaable$_{\text{std}}$, toutes remémorées en même temps, cependant la phase de sélection des recettes les plus typiques par rapport à $\mathtt{Q}_{ex}$ permet de classer les 5 recettes avant de les adapter dans le même ordre.

Le classement de ces 5 recettes se fait grâce à plusieurs critères :

critère 1 : `Cerise` doit remplacer une sous-classe de `FruitÀNoyau` au moins aussi typique que `Cerise` pour `FruitÀNoyau`.

critère 2 : `Poire` doit remplacer une sous-classe de `FruitÀPépins` au moins aussi typique que `Poire` pour `FruitÀPépins`.

$\mathtt{R}_1$, la recette remémorée dans `Rép`$_1$, est la seule recette à satisfaire les deux critères : $\mathtt{R}_1$ domine toutes les autres recettes. En effet, `Abricot`, indexant $\mathtt{R}_1$, est plus typique que `Cerise` pour `FruitÀNoyau` et `Pomme`, une autre sous-classe indexant $\mathtt{R}_1$, est plus typique que `Poire` pour

FruitÀNoyau. Comme R_1 domine tous les autres recettes, alors $Rép_1$ est adaptée et retournée en premier à l'utilisateur par eTaaable$_{TypM}$.

R_4, la recette remémorée dans la réponse $Rép_4$, satisfait mieux les deux critères que R_2 (respectivement R_3), la recette remémorée dans la réponse $Rép_2$ (respectivement $Rép_3$). En effet, Pêche, indexant R_4 est plus typique que Cerise pour FruitÀNoyau alors que Avocat indexant R_2 et Olive indexant R_3 sont moins typiques que Cerise pour FruitÀNoyau. De plus, Raisin, indexant R_4 est moins typique que Poire pour FruitÀPépins mais est plus typique que Citron indexant R_2 et R_3.

R_5, la recette remémorée dans la réponse $Rép_5$, satisfait mieux les deux critères que R_2 et R_3. En effet, Mangue, indexant R_5, est moins typique que Cerise pour FruitÀNoyau mais est plus typique que Avocat indexant R_2 et Olive indexant R_3. De plus, Pomme, indexant R_5, est plus typique que Poire pour FruitÀPépins alors que Citron, indexant R_2 et R_3, est moins typique que Poire pour FruitÀPépins.

R_4 satisfait mieux le critère 1 que R_5 et R_5 satisfait mieux le critère 2 que R_4. En effet, Pêche, indexant R_4, est plus typique que Cerise pour FruitÀNoyau alors que Mangue, indexant R_5 est moins typique que Cerise pour FruitÀNoyau. Pomme, indexant R_5 est plus typique que que Poire pour FruitÀPépins alors que Raisin, indexant R_4 est moins typique que Poire pour FruitÀPépins. Ainsi, ni R_4, ni R_5 ne domine l'autre. Pour ordonner ces deux recettes, $\Delta_{casTyp}(R_4,Q_{ex})$ et $\Delta_{casTyp}(R_5,Q_{ex})$ doivent être calculées en prenant en compte les degrés de typicalité du tableau 6.13 et selon les équations (5.2) et (5.1) :

$$\begin{aligned}\Delta_{casTyp}(R_4,Q_{ex}) &= \Delta_{typ}(\text{Pêche} \overset{\text{FruitÀNoyau}}{\rightsquigarrow} \text{Cerise}) \\ &+ \Delta_{typ}(\text{Raisin} \overset{\text{FruitÀPépins}}{\rightsquigarrow} \text{Poire}) \\ &= \max(0; 0{,}92 - 1{,}00) + \max(0; 0{,}50 - 0{,}33) \\ &= 0{,}17\end{aligned}$$

$$\begin{aligned}\Delta_{casTyp}(R_5,Q_{ex}) &= \Delta_{typ}(\text{Mangue} \overset{\text{FruitÀNoyau}}{\rightsquigarrow} \text{Cerise}) \\ &+ \Delta_{typ}(\text{Pomme} \overset{\text{FruitÀPépins}}{\rightsquigarrow} \text{Poire}) \\ &= \max(0; 0{,}92 - 0{,}08) + \max(0; 0{,}50 - 0{,}67) \\ &= 0{,}84\end{aligned}$$

Comme $\Delta_{casTyp}(R_4,Q_{ex}) < \Delta_{casTyp}(R_5,Q)$ alors $R_4 \succeq_T^{Q_{ex}} R_5$. Ainsi, R_4 est retournée avant R_5.

R_2 satisfait mieux le critère 1 et est a égalité pour le critère 2 par rapport à R_3. En effet, Avocat, indexant R_2 est moins typique que Cerise pour FruitÀNoyau mais est plus typique que Olive pour FruitÀNoyau et Citron, une sous-classe de FruitÀPépins indexe R_2 et R_3. Ainsi, R_2 domine R_3 et $Rép_2$ est retournée avant $Rép_3$.

Finalement, eTaaable$_{TypM}$ retourne les réponses dans cet ordre : $Rép_1$, $Rép_4$, $Rép_5$, $Rép_2$ et $Rép_3$.

6.4 Évaluation de l'intégration de la typicalité dans le moteur de RàPC

6.4.1 Hypothèses

Afin de valider l'apport de la prise en compte de la typicalité dans le moteur d'un système de RàPC, nous posons les deux hypothèses suivantes :

(H1) La prise en compte de la typicalité dans le moteur d'un système de RàPC améliore les réponses par rapport à un système ne prenant pas en compte la typicalité.

(H2) La prise en compte de la typicalité dans le moteur d'un système de RàPC ne diminue pas la satisfaction des réponses par rapport à un système prenant en compte la typicalité grâce au raffinement de l'ontologie utilisée.

6.4.2 Méthode de comparaison

Trois versions de eTaaable sont comparées :

- eTaaable$_{\text{std}}$: la version standard de eTaaable qui ne prend pas en compte la typicalité.
- eTaaable$_{\text{raffN}}$: la version de eTaaable qui utilise l'ontologie initiale de aTaaable raffinée grâce à la méthode des niveaux, qui est la méthode ayant permis de retourner les meilleurs réponses (cf. section 6.2).
- eTaaable$_{\text{TypM}}$: la version de eTaaable qui prend en compte la typicalité dans son moteur.

eTaaable$_{\text{TypM}}$ utilise les degrés de typicalité de la section 5.2.3 du chapitre 5.

(H1) sera validée si eTaaable$_{\text{TypM}}$ retourne significativement de meilleures réponses que eTaaable$_{\text{std}}$. (H2) sera validée si eTaaable$_{\text{raffK}}$ ne retourne par significativement de meilleures réponses que eTaaable$_{\text{TypM}}$.

6.4.3 Plan de test pour l'évaluation

Le plan de test utilisé est le plan de test de la section 6.2.3 qui contient 20 requêtes portant sur des tartes. Nous utilisons des requêtes composées d'une sous-classes de la hiérarchie des aliments, des trois classes suivantes : `FruitÀNoyau`, `FruitÀPépins` et `Baie`, car notre base de cas n'est pas assez importante pour remémorer des recettes indexées par plus d'une sous-classe de ces trois classes.

6.4.4 Analyse brute des résultats retournés par les systèmes

Comme nous avons réutilisé le plan de test de la section 6.2.3 et que ces requêtes ne sont composées que d'une seule classe de la hiérarchie des aliments, les réponses de eTaaable$_{\text{TypM}}$ sont identiques à eTaaable$_{\text{raffN}}$. En effet, avec la réorganisation de la hiérarchie des aliments avec la méthode des niveaux, la première généralisation d'une sous-classes $\texttt{B}_i$ de `A` consiste à remémorer des cas contenant des sous-classe au moins aussi typique que $\texttt{B}_i$. Puis les prochaines étapes de généralisation permettent de minimiser l'écart de typicalité. Ainsi, eTaaable$_{\text{raffN}}$ et eTaaable$_{\text{TypM}}$ retournent les mêmes réponses pour des requêtes composées d'une seule classe de la hiérarchie des aliments, qui a été réorganisée pour eTaaable$_{\text{raffN}}$.

6.4.5 Validation des résultats

Grâce au test des rangs signés de Wilcoxson, nous avons conclu dans la section 6.2.7 que les réponses de eTaaable$_{\text{raffN}}$ étaient significativement meilleures que eTaaable$_{\text{std}}$. Comme les réponses de eTaaable$_{\text{TypM}}$ sont identiques à eTaaable$_{\text{raffN}}$, alors les réponses de eTaaable$_{\text{TypM}}$ sont significativement meilleures que eTaaable$_{\text{std}}$: (H1) est validée. De plus, pour les mêmes raisons, eTaaable$_{\text{raffN}}$ ne retourne pas de meilleures réponses que eTaaable$_{\text{TypM}}$: (H2) est validée.

6.5 Conclusion

Nous avons montré expérimentalement dans ce chapitre que raffiner l'ontologie grâce à la typicalité améliore les réponses retournées par eTaaable de façon significative quelque soit la méthode utilisée parmi la méthode des seuils, la méthode des k-moyennes et la méthode des niveaux. Un seul des apports proposé ne permet pas d'améliorer significativement les réponses retournées par eTaaable : avec la méthode des seuils, séparer les sous-classes normales et les sous-classes typiques d'une classe dans la hiérarchie des aliments ne permet pas d'améliorer significativement les réponses. Ainsi, pour certaines classes, un seul seuil pourrait être utilisé dans la méthode des seuils, afin de discriminer les sous-classes atypiques et les sous-classes non atypiques de la classe, grâce à la typicalité ; par exemple, pour la classe des `Baie` qui ne contient que 7 sous-classes.

Nous avons également montré expérimentalement que la prise en compte de la typicalité dans le moteur de eTaaable améliore les réponses de façon significative par rapport à eTaaable qui ne prend pas en compte la typicalité. De plus, il n'y a pas de différence au niveau de la satisfaction des réponses entre la version d'eTaaable prenant en compte la typicalité dans son moteur et les versions d'eTaaable utilisant une ontologie raffinée grâce à la typicalité.

Il serait envisageable de réaliser un nouveau plan de tests avec des requêtes composées d'autres sous-classes que les sous-classes de `FruitÀNoyau`, `FruitÀPépins` et `Baie`. De plus, les requêtes pourraient être composées de plus d'une classe de la hiérarchie des aliments.

Par ailleurs, il serait intéressant d'appliquer les différentes approches concernant la typicalité dans des systèmes de RÀPC différents de eTaaable.

Conclusion

Cette thèse propose deux approches pour améliorer la qualité des réponses d'un système de raisonnement à partir de cas (RÀPC) qui utilise des connaissances produites par une communauté en ligne.

La première approche est liée au fait que les connaissances produites par une communauté en ligne peuvent être d'une fiabilité variable à cause de la présence d'utilisateurs « non experts » ou encore malveillants. Afin de pouvoir contrôler la fiabilité des connaissances, nous avons construit un modèle de métaconnaissances nommé MKM (*Meta-Knowledge Model*). MKM est un modèle générique permettant de représenter la fiabilité des unités de connaissances (UC) produites par une communauté en ligne, indépendamment de l'utilisation qui est faite des UC produites. À partir d'interactions spécifiques produites par les utilisateurs de la communauté en ligne, différents scores sont calculés pour représenter des métaconnaissances relatives à la croyance d'une UC, à la qualité d'une UC, à la confiance d'un utilisateur envers un autre utilisateur, à la réputation d'un utilisateur et finalement à la fiabilité d'une UC. Dans le cadre de cette thèse qui a pour cadre d'application le RÀPC, le modèle MKM permet d'étendre un système de RÀPC utilisant des connaissances produites par une communauté en ligne. Cette extension est possible grâce à l'ajout de deux fonctions au système de RÀPC : une fonction de filtre dont le but est de filtrer les connaissances non fiables afin qu'elles ne soient pas utilisées par le système de RÀPC et une fonction de classement dont le but est de classer les réponses retournées par le système selon un score de fiabilité.

La deuxième approche est liée au fait qu'utiliser une communauté en ligne pour acquérir des connaissances peut conduire à l'acquisition de différents avis à propos d'une même UC. Dans cette thèse, nous nous sommes intéressés aux relations de subsomption des connaissances du domaine. L'utilisation de différents avis à propos des relations de subsomption entre les sous-classes $\texttt{B}_i$ d'une classe `A`, nous permet de représenter une certaine gradualité dans les relations de subsomption en représentant la typicalité d'une sous-classe `B` par rapport à sa classe `A`. Le but est de discriminer entre elles les $\texttt{B}_i$ par rapport à `A` dans l'ontologie du domaine grâce à la typicalité de chaque $\texttt{B}_i$ par rapport à `A` représentée par un degré de typicalité. Dans un premier temps, une acquisition des degrés de typicalité nous a permis de raffiner l'ontologie du domaine utilisée par le système de RÀPC dans le but de rendre plus progressive la phase de remémoration du RÀPC. Dans un deuxième temps, l'acquisition des degrés de typicalité nous a permis de prendre en compte la typicalité dans le moteur du RÀPC.

L'apport de ces deux approches, à savoir l'extension d'un système de RÀPC avec MKM et l'utilisation de la typicalité dans un système de RÀPC, a été évalué dans le cadre de eTaaable, un système de RÀPC du domaine culinaire qui adapte des recettes de cuisine en utilisant des connaissances produites par la communauté en ligne du site Web collaboratif aTaaable. Une base de tests a été construite afin de comparer différentes versions du système eTaaable. La construction de la base de tests est un apport complémentaire de cette thèse puisque l'objectif est de la distribuer afin de comparer d'autres systèmes que eTaaable, notamment dans le

cadre du CCC (*Computer Cooking Contest*), une compétition organisée chaque année dans le cadre des congrès ICCBR (*International Conference on Case-Based Reasoning*). Les évaluations ont permis de conclure que la fonction de filtre ainsi que la fonction de classement ajoutées à eTaaable étendu avec MKM améliorent les réponses par rapport à eTaaable sans cette extension. Les évaluations ont permis de conclure que raffiner l'ontologie du domaine utilisée par eTaaable grâce à la typicalité améliore également les réponses par rapport à eTaaable n'utilisant pas d'ontologie raffinée. Enfin, la prise en compte de la typicalité dans le moteur de RàPC du système eTaaable permet là encore d'améliorer les réponses de eTaaable par rapport à eTaaable qui n'utilise pas la typicalité dans son moteur.

Nous avons validé l'apport de chaque approche dans le cadre expérimental du système eTaaable. Les poids des scores de qualité, de confiance et de réputation ainsi que les seuils utilisés pour filtrer les UC non fiables pourraient être fixés expérimentalement grâce à de nouvelles évaluations. Il serait intéressant de valider l'apport des différentes approches dans des systèmes de RàPC différents de eTaaable utilisant des connaissances produites par une communauté en ligne. De plus, d'autres systèmes à base de connaissances — n'utilisant par le RàPC— étendus avec MKM pourront être évalués afin de valider l'aspect générique de MKM. Concernant la typicalité, il serait également intéressant de réaliser des évaluations avec une plus grande variété de requêtes et de cas.

Chacune des approches a fait émerger différentes problématiques déjà présentées en conclusion des différents chapitres de cette thèse et que nous rappelons brièvement ci-dessous.

Pour MKM, une première problématique vise à modéliser la confiance par transitivité afin de calculer un score de fiabilité personnalisé. La confiance d'un utilisateur u envers un utilisateur v pourrait être calculée grâce aux recommandations d'autres utilisateurs. MKM devrait alors faire la différence entre un score de confiance et un score de recommandation. De plus, des fonctions devraient être mises en place pour prendre en compte les différentes recommandations dans le réseau de confiance. Ainsi, étendre un système de RàPC avec un tel modèle pourrait permettre de construire un système de RàPC personnalisé.

Une autre problématique porte sur les fonctions d'agrégation des différents scores. Alors que nous utilisons la moyenne arithmétique pour prendre en compte les différents scores de croyance à propos d'une même UC dans le calcul du score de la qualité et pour prendre en compte les différents scores de confiance envers un utilisateur dans le calcul du score de réputation, d'autres opérateurs d'agrégation tels que le minimum, le maximum ou la médiane peuvent être utilisés afin de les comparer avec l'utilisation de la moyenne arithmétique. De plus, l'utilisation de scores de recommandation d'une UC, dans le calcul du score de fiabilité de cette UC, pourrait être étudié.

Une autre perspective concerne l'intégration de la fiabilité directement dans la phase de remémoration du système de RàPC. Une approche potentielle consisterait à pondérer le coût de généralisation par la fiabilité directement lors de la remémoration, mais aussi à prendre en compte la fiabilité durant la spécialisation du problème cible.

Enfin, l'utilisation de la théorie des sous-ensembles flous pour représenter la gradualité des classes de l'ontologie du domaine utilisées par un système de RàPC, à la place de la typicalité, est une problématique à étudier. Une relation de subsomption entre une sous-classe et une classe ne serait plus représentée en termes de tout ou rien mais par un degré de subsomption. Ces degrés de subsomption pourraient, là encore, être utilisés dans le moteur du système de RàPC durant la remémoration.

Annexe A

La logique de descriptions $\mathcal{ALC}$

Les logiques de descriptions [Baader, 2003] forment une famille de formalismes utilisés pour la représentation d'un domaine de connaissances. $\mathcal{ALC}$ est une logique de descriptions que nous utilisons pour définir les notions utiles à notre application. Dans $\mathcal{ALC}$, les éléments manipulés sont les classes qui représentent des ensembles d'éléments, les propriétés qui représentent des relations binaires et les instances qui représentent les éléments d'un ensemble. Une classe peut correspondre à :

- la classe la plus spécifique, notée $\bot$;
- la classe la plus générale, notée $\top$;
- à une classe atomique $\mathtt{A}$ qui est un nom de classe ;
- à la classe, qui avec $\mathtt{C}$ et $\mathtt{D}$ deux classes et $\mathtt{p}$ une propriété, peut être de la forme : $\neg\mathtt{C}$ (le complémentaire de $\mathtt{C}$), $\mathtt{C}\sqcup\mathtt{D}$ (la disjonction de $\mathtt{C}$ et $\mathtt{D}$), $\mathtt{C}\sqcap\mathtt{D}$ (la conjonction de $\mathtt{C}$ et $\mathtt{D}$), $\exists\ \mathtt{p}.\mathtt{C}$ (le quantificateur existentiel sur $\mathtt{C}$) et $\forall\ \mathtt{p}.\mathtt{C}$ (le quantificateur universel sur $\mathtt{C}$).

Une base de connaissances BC en $\mathcal{ALC}$ est composée de :

- Une TBox, aussi appelée niveau terminologique, qui décrit les connaissances générales du domaine. La TBox est un ensemble de formules qui sont des axiomes terminologiques et peuvent être de la forme $\mathtt{C}\sqsubseteq\mathtt{D}$ (la subsomption) ou $\mathtt{C}\equiv\mathtt{D}$ (l'équivalence), où $\mathtt{C}$ et $\mathtt{D}$ sont deux classes.
- Une ABox, aussi appelée niveau factuel, qui décrit les connaissances sur des individus spécifiques. L'ABox est un ensemble de formules qui sont des assertions et qui peuvent être la forme $\mathtt{C}(a)$ ou $\mathtt{p}(a,b)$, où $\mathtt{C}$ est une classe, $\mathtt{p}$ est une propriété et, a et b sont deux instances.

En $\mathcal{ALC}$, une interprétation est un couple $\mathcal{I}=(\Delta_\mathcal{I},\cdot^\mathcal{I})$ où $\Delta_\mathcal{I}$ est un ensemble non vide appelé le domaine d'interprétation et $\cdot^\mathcal{I}$ est une fonction d'interprétation. La fonction d'interprétation associe à chaque classe atomique $\mathtt{A}$ un sous-ensemble $\mathtt{A}^\mathcal{I}$ de $\Delta_\mathcal{I}$, à chaque propriété $\mathtt{p}$ un sous-ensemble $\mathtt{p}^\mathcal{I}$ de $\Delta_\mathcal{I}\times\Delta_\mathcal{I}$, et à chaque instance a, un élément $a^\mathcal{I}$ de $\Delta_\mathcal{I}$. De plus, la fonction d'interprétation $\cdot^\mathcal{I}$ est étendue sur l'ensemble de toutes les classes de la façon suivante :

- $\top^\mathcal{I}=\Delta_\mathcal{I}$;
- $\bot^\mathcal{I}=\emptyset$;
- $(\neg\mathtt{C})^\mathcal{I}=\Delta_\mathcal{I}\backslash\mathtt{C}^\mathcal{I}$;
- $(\mathtt{C}\sqcap\mathtt{D})^\mathcal{I}=\mathtt{C}^\mathcal{I}\cap\mathtt{D}^\mathcal{I}$;
- $(\mathtt{C}\sqcup\mathtt{D})^\mathcal{I}=\mathtt{C}^\mathcal{I}\cup\mathtt{D}^\mathcal{I}$;
- $\{(\exists\ \mathtt{p}.\mathtt{C})^\mathcal{I}=x\in\Delta_\mathcal{I}\mid\exists y\in\mathtt{C}^\mathcal{I},(x,y)\in\mathtt{p}^\mathcal{I}\}$;
- $\{(\forall\ \mathtt{p}.\mathtt{C})^\mathcal{I}=x\in\Delta_\mathcal{I}\mid\forall y\in\Delta_\mathcal{I},(x,y)\in\mathtt{p}^\mathcal{I}\Rightarrow y\in\mathtt{C}^\mathcal{I}\}$.

Étant donné une interprétation $\mathcal{I}$, la satisfaction par $\mathcal{I}$ d'une formule f (axiome terminologique ou assertion) est notée $\mathcal{I} \models f$, signifiant « $\mathcal{I}$ satisfait f », et est définie comme suit :

- $\mathcal{I} \models \mathtt{C} \sqsubseteq \mathtt{D}$ si $\mathtt{C}^{\mathcal{I}} \subseteq \mathtt{D}^{\mathcal{I}}$;
- $\mathcal{I} \models \mathtt{C} \equiv \mathtt{D}$ si $\mathtt{C}^{\mathcal{I}} = \mathtt{D}^{\mathcal{I}}$;
- $\mathcal{I} \models \mathtt{C}(a)$ si $a^{\mathcal{I}} \in \mathtt{C}^{\mathcal{I}}$;
- $\mathcal{I} \models \mathtt{p}(a,b)$ si $(a,b) \in \mathtt{p}^{\mathcal{I}}$.

De plus, $\mathcal{I} \models BC$, signifiant que $\mathcal{I}$ est un modèle de BC, si $\mathcal{I} \models f$, pour tout $f \in BC$. Finalement, $BC \models f$, signifiant que BC satisfait f, si pour toute interprétation $\mathcal{I}$ on a : si $\mathcal{I} \models BC$ alors $\mathcal{I} \models f$.

Annexe B

Le langage RDF

B.1 RDF

RDF [34] est un standard du W3C (*World Wide Web Consortium*) pour le Web Sémantique. Le langage RDF permet de faire des annotations sémantiques des ressources sur le Web. L'élément de base du langage RDF est un triplet de la forme ⟨sujet prédicat objet⟩. Le sujet est une ressource, le prédicat est une propriété et l'objet est une ressource ou un littéral (par exemple, une valeur numérique). Une ressource peut être soit une URI (*Uniform Resource Identifier*) qui correspond à l'adresse unique de la ressource, soit une ressource anonyme. Par exemple, ⟨`Paul aPourEnfant Marie`⟩ est un triplet qui encode que « Paul a pour enfant Marie » et ⟨`Paul aPourFemme Thérèse`⟩ est un triplet qui encode que « Paul a pour femme Thérèse ».

B.2 Graphe RDF

Un ensemble de triplets forme un graphe RDF. Dans un graphe RDF, les sujets et objets sont représentés par des nœuds et les propriétés étiquettent les arcs reliant sujets à objets. Les deux triplets donnés précédemment en exemple sont représentés dans le graphe RDF de la figure B.1.

FIGURE B.1 – Graphe RDF représentant les deux triplets : ⟨`Paul aPourEnfant Marie`⟩ et ⟨`Paul aPourFemme Thérèse`⟩.

B.3 Vocabulaire RDF

Un vocabulaire RDF correspond à une collection d'URI qui vont être utilisées dans un graphe RDF. Dans un vocabulaire RDF, les URI commencent toute par un intitulé commun nommé espace de noms. Un vocabulaire RDF est associé à l'espace de noms que partagent les URI de la collection.

Pour alléger l'écriture, les espaces de noms peuvent être associés à un préfixe. Par exemple :

34. `https://www.w3.org/RDF/`

- le préfixe `rdf` est associé à l'espace de nom `http://www/.w3.org/1999/02/22-rdf-syntax-ns#` qui correspond au vocabulaire construit autour de RDF.
- le préfixe `rdfs` est associé à l'espace de nom `http://www/.w3.org/2000/01/rdf-schema#` qui correspond au vocabulaire qui a été construit pour organiser les données RDF.

Dans un même graphe RDF, plusieurs vocabulaires peuvent être utilisés. Par exemple, les deux triplets suivant peuvent être représentés dans un même graphe : le triplet ⟨`Enfant rdfs:SubClassOf Parent`⟩ signifie que la classe `Enfant` est une sous-classe de la classe `Parent` et le triplet ⟨`Enfant rdf:type Marie`⟩ signifie que la ressource Marie est une instance de la classe `Enfant`.

B.4 *Triplestore* et SPARQL

Un *triplestore* est une base de données destinée au stockage et à la gestion de données RDF. Un triplestore stocke seulement des triplets RDF et permet leur mise à jour et leur interrogation grâce à un langage de requête : le langage SPARQL [35] [Prud'hommeaux et Seaborne, 2006].

35. `https://www.w3.org/TR/rdf-sparql-query/`

Annexe C

Notions et notations mathématiques utilisées

C.1 Sous-ensemble

Étant donné un ensemble E, on note 2^E, l'ensemble des sous-ensembles de E.

C.2 Multi-ensemble

Étant donné un ensemble E, un multi-ensemble M sur E est une application de E dans $\mathbb{N}$: $E \to \mathbb{N}$. Intuitivement, un multi-ensemble M sur E est un sous-ensemble de E dans lequel on autorise un élément à « apparaître plusieurs fois ». On note $\mathcal{M}(E)$ l'ensemble des multi-ensembles sur E. Par exemple, pour l'ensemble [0,1], $\mathcal{M}([0,1])$ est l'ensemble des multi-ensembles sur [0,1].

C.3 Relation d'ordre

Soit E un ensemble et $\leqslant$ une relation binaire sur E. Sur E, $\leqslant$ est une relation d'ordre si :
– $\leqslant$ est réflexive (i.e. pour tout $x \in E$, $x \leqslant x$) ;
– $\leqslant$ est antisymétrique (i.e. pour $x \in E$ et $y \in E$, si $x \leqslant y$ et $y \leqslant x$, alors x = y) ;
– $\leqslant$ est transitive (i.e. pour $x \in E$, $y \in E$ et $z \in E$, si $x \leqslant y$ et $y \leqslant z$, alors $x \leqslant z$).

Un ensemble ordonné est un couple $(E, \leqslant)$ où E est un ensemble et $\leqslant$ est une relation d'ordre sur E. $\leqslant$ est une relation d'ordre total si tous les éléments de E sont deux à deux comparables (i.e. pour tout $x \in E$ et $y \in E$, on a $x \leqslant y$ ou $y \leqslant x$) : on dit alors que $(E, \leqslant)$ est un ensemble totalement ordonné. Si tous les éléments de E ne sont pas deux à deux comparables, alors $\leqslant$ est une relation d'ordre partiel et $(E, \leqslant)$ est un ensemble partiellement ordonné.

C.4 Diagramme de Hasse

Soit E un ensemble et $\to$ une relation binaire sur E. On appelle fermeture réflexive-transitive de $\to$ la relation $\to^*$ définie par :

$$x \to^* y \text{ si } x = y \text{ ou s'il existe } z \in E \text{ tel que } x \to z \text{ et } z \to^* y.$$

Si $(E, \leqslant)$ est un ensemble ordonné fini, on appelle diagramme de Hasse de E, le graphe $(E, \to)$ tel que $\to^* = \leqslant$ et $\to$ est la plus petite relation binaire r vérifiant $r^* = \leqslant$ (i.e. si $r^* = \leqslant$,

alors $x \to y$ entraîne $x\ r\ y$ pour tout $(x,y) \in E^2$). Pour construire le diagramme de Hasse de $(E, \leqslant)$, on supprime du graphe $(E, \leqslant)$ tous les arcs (x,x) (arcs de réflexivité) et tous les arcs (x,y) tels qu'il existe $z \in E$ avec $x \leqslant z$, $z \leqslant y$, $z \neq x$ et $z \neq y$ (arcs de transitivité).

Bibliographie

[Aamodt et Nygård, 1995] Aamodt, A. et Nygård, M. (1995). Different roles and mutual dependencies of data, information, and knowledge—an ai perspective on their integration. *Data & Knowledge Engineering*, 16(3):191–222.

[Aamodt et Plaza, 1994] Aamodt, A. et Plaza, E. (1994). Case-based reasoning; foundational issues, methodological variations, and system approaches. *AI COMMUNICATIONS*, 7(1):39–59.

[Abdul-Rahman et Hailes, 1996] Abdul-Rahman, A. et Hailes, S. (1996). A Distributed Trust Model. *In Proceedings of the 1997 workshop on New security paradigms NSPW 97*, pages 48–60. ACM Press.

[Abdul-Rahman et Hailes, 2000a] Abdul-Rahman, A. et Hailes, S. (2000a). Supporting Trust in Virtual Communities. IEEE Computer Society.

[Abdul-Rahman et Hailes, 2000b] Abdul-Rahman, A. et Hailes, S. (2000b). Supporting Trust in Virtual Communities. *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences-Volume 6*.

[Adler *et al.*, 2008] Adler, B., Benterou, J., Chatterjee, K., De Alfaro, L., Pye, I. et Raman, V. (2008). Assigning trust to wikipedia content. *In Proceedings of the 4th International Symposium on Wikis*. ACM.

[Agichtein *et al.*, 2008] Agichtein, E., Castillo, C., Donato, D., Gionis, A. et Mishne, G. (2008). Finding high-quality content in social media. *In Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 183–194.

[Aha et Gupta, 2002] Aha, D. et Gupta, K. (2002). Causal query elaboration in conversational case-based reasoning. *In FLAIRS Conference*, pages 95–100.

[Aha *et al.*, 1991] Aha, D., Kibler, D. et Albert, M. (1991). Instance-based learning algorithms. *Machine learning*, 6(1):37–66.

[Alchiekh Haydar, 2014] Alchiekh Haydar, C. (2014). *Les systèmes de recommandation à base de confiance*. Thèse de doctorat. Thèse de doctorat dirigée par Boyer, A. et Roussanaly, A. Informatique Université de Lorraine 2014.

[Artin, 2015] Artin, E. (2015). *The gamma function*. Courier Dover Publications.

[Artz et Gil, 2007] Artz, D. et Gil, Y. (2007). A survey of trust in computer science and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71.

[Baader, 2003] Baader, F. (2003). *The description logic handbook: Theory, implementation and applications*. Cambridge university press.

[Badra, 2009] Badra, F. (2009). *Extraction de connaissances d'adaptation en raisonnement à partir de cas*. Thèse de doctorat. Thèse de doctorat dirigée par Napoli, A. et Lieber, J. Informatique Nancy 1 2009.

[Badra *et al.*, 2008] Badra, F., Bendaoud, R., Bentebibel, R., Champin, P., Cojan, J., Cordier, A., Després, S., Jean-Daubias, S., Lieber, J. et Meilender, T. (2008). Taaable: Text mining, ontology engineering, and hierarchical classification for textual case-based cooking. *In ECCBR Workshops*, pages 219–228.

[Badra *et al.*, 2009] Badra, F., Cordier, A. et Lieber, J. (2009). Opportunistic Adaptation Knowledge Discovery. *In 8th International Conference on Case-Based Reasoning (ICCBR-09)*, pages 60–74, Seattle, United States. Springer.

[Bakshy *et al.*, 2011] Bakshy, E., Hofman, J., Mason, W. A. et Watts, D. (2011). Everyone's an influencer: quantifying influence on twitter. *In Proceedings of the fourth ACM international conference on Web search and data mining*, pages 65–74. ACM.

[Barber, 1983] Barber, B. (1983). *The logic and limits of trust*. Rutgers University Press.

[Bareiss *et al.*, 1990] Bareiss, E. R., Porter, B. E. et Wier, C. C. (1990). Protos: An exemplar-based learning apprentice. *In* Gaines, B. R. et Boose, J. H., éditeurs : *Machine Learning and Uncertain Reasoning*, pages 1–13. Academic Press Ltd., London, UK.

[Barsalou et Sewell, 1984] Barsalou, L. et Sewell, D. (1984). Constructing representations of categories from different points of view. Technical report, Atlanta. Emory Cognition Project Technical Report #2.

[Barsalou et Sewell, 1985] Barsalou, L. W. et Sewell, D. R. (1985). Contrasting the representation of scripts and categories. *Journal of Memory and Language*, 24(6):646–665.

[Beckner, 1959] Beckner, M. (1959). *The biological way of thought*. Columbia University Press.

[Bellazzi *et al.*, 1998] Bellazzi, R., Montani, S. et Portinale, L. (1998). Retrieval in a prototype-based case library: a case study in diabetes therapy revision. *In Advances in Case-Based Reasoning*, pages 64–75. Springer.

[Bergmann, 2002] Bergmann, R. (2002). *Experience management: foundations, development methodology, and internet-based applications*.

[Bergmann *et al.*, 2005] Bergmann, R., Kolodner, J. et Plaza, E. (2005). Representation in case-based reasoning. *The Knowledge Engineering Review*, 20(03):209–213.

[Berners-Lee, 1997] Berners-Lee, T. (1997). Metadata architecture. http://www.w3.org/DesignIssues/Metadata.html.

[Bertino et Lim, 2010] Bertino, E. et Lim, H. (2010). Assuring data trustworthiness-concepts and research challenges. *In Secure Data Management*, pages 1–12. Springer.

[Boon et Holmes, 1991] Boon, S. et Holmes, J. (1991). The dynamics of interpersonal trust: Resolving uncertainty in the face of risk. *In* Hinde, R. et Groebel, J., éditeurs : *Cooperation and prosocial behaviour*, chapitre 11, pages 190–211. Cambridge University Press.

[Boyd et Richerson, 2009] Boyd, R. et Richerson, P. (2009). Culture and the evolution of human cooperation. *Philosophical Transactions of the Royal Society of London - Series B: Biological Sciences*, 364(1533):3281–3288.

[Boyle et Bonacich, 1970] Boyle, R. et Bonacich, P. (1970). The development of trust and mistrust in mixed-motive games. *Sociometry*, 33(2):123–139.

[Bubenko et Orci, 1989] Bubenko, J. et Orci, I. (1989). Knowledge base management systems: a database view. *In Foundations of knowledge base management*, pages 373–378. Springer.

[Buneman *et al.*, 2001] Buneman, P., Khanna, S. et Wang-Chiew, T. (2001). Why and where: A characterization of data provenance. *In Database Theory (ICDT 2001)*, pages 316–330. Springer.

[Caliński et Harabasz, 1974] Caliński, T. et Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27.

[Castelfranchi et Falcone, 1998] Castelfranchi, C. et Falcone, R. (1998). Principles of trust for mas: Cognitive anatomy, social importance, and quantification. *In Proceedings of the International Conference on Multi Agent Systems (ICMAS-98)*, pages 72–79. IEEE.

[Castillo *et al.*, 2011] Castillo, C., Mendoza, M. et Poblete, B. (2011). Information credibility on twitter. *In Proceedings of the 20th international conference on World wide web*, pages 675–684.

[Ceausu et Desprès, 2005] Ceausu, V. et Desprès, S. (2005). Fouille de textes pour orienter la construction d'une ressource terminologique. *In EGC*, pages 239–244.

[Christianson et Harbison, 1996] Christianson, B. et Harbison, W. (1996). Why isn't trust transitive? *In Security protocols*, pages 171–176. Springer Berlin Heidelberg.

[Cleverdon, 1997] Cleverdon, C. (1997). Readings in information retrieval. chapitre The Cranfield Tests on Index Language Devices, pages 47–59. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Cleverdon *et al.*, 1966] Cleverdon, C., Mills, J. et Keen, M. (1966). Factors determining the performance of indexing systems volume 1: Design. *Technical report*.

[Cobden, 2014] Cobden, M. (2014). *Engineering a semantic web trust infrastructure*. Thèse de doctorat, University of Southampton.

[Cojan, 2011] Cojan, J. (2011). *Application de la théorie de la révision des connaissances au raisonnement à partir de cas*. Thèse de doctorat. Thèse de doctorat dirigée par Lieber, J. Informatique Nancy 1 2011.

[Cordier, 2008] Cordier, A. (2008). *Interactive and opportunistic knowledge acquisition in case-based reasoning*. Thèse de doctorat. Thèse de doctorat dirigée par Mille, A. et Fuchs, B. Informatique Lyon 1 2008.

[Cordier *et al.*, 2014] Cordier, A., Dufour-Lussier, V., Lieber, J., Nauer, E., Badra, F., Cojan, J., Gaillard, E., Infante-Blanco, L., Molli, P., Napoli, A. et Skaf-Molli, H. (2014). Taaable: a Case-Based System for personalized Cooking. *In* Montani, S. et Jain, L. C., éditeurs : *Successful Case-based Reasoning Applications-2*, volume 494 de *Studies in Computational Intelligence*, pages 121–162. Springer.

[Cordier *et al.*, 2008] Cordier, A., Fuchs, B., Lana De Carvalho, L., Lieber, J. et Mille, A. (2008). Opportunistic Acquisition of Adaptation Knowledge and Cases - The IakA Approach. *In 9th European Conference on Case-Based Reasoning (ECCBR-08*, pages 150–164, Trier, Germany. Springer.

[Cordier *et al.*, 2009] Cordier, A., Lieber, J., Molli, P., Nauer, E., Skaf-Molli, H. et Toussaint, Y. (2009). Wiki-taaable: A semantic wiki as a blackboard for a textual case-based reasoning system. *In 4th Workshop on Semantic Wikis (SemWiki 2009). 6th European Semantic Web Conference, Heraklion*, pages 88–101.

[Crosby, 1980] Crosby, P. (1980). *Quality is free: The art of making quality certain*. Signet.

[Cruse, 1990] Cruse, D. (1990). Prototype theory and lexical semantics. *In* Tsohatzidis, S. L., éditeur : *Meanings and prototypes: Studies in linguistic categorization*, pages 382–402. London: Routledge.

[Dai *et al.*, 2008] Dai, C., Lin, D., Bertino, E. et Kantarcioglu, M. (2008). An approach to evaluate data trustworthiness based on data provenance. *In Secure Data Management*, pages 82–98. Springer.

[Das et Islam, 2012] Das, A. et Islam, M. (2012). Securedtrust: a dynamic trust computation model for secured communication in multiagent systems. *Dependable and Secure Computing, IEEE Transactions on*, 9(2):261–274.

[Davenport et Prusak, 1998] Davenport, T. et Prusak, L. (1998). *Working knowledge: How organizations manage what they know*. Harvard Business Press.

[Davies et Bouldin, 1979] Davies, D. et Bouldin, D. (1979). A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):224–227.

[DeLone et McLean, 1992] DeLone, W. et McLean, E. (1992). Information systems success: The quest for the dependent variable. *Information systems research*, 3(1):60–95.

[Deutsch, 1962] Deutsch, M. (1962). Cooperation and trust: Some theoretical notes. *In* Jones, M. R., éditeur : *Nebraska Symposium On Motivation*, volume 10, pages 275–320. University of Nebraska Press.

[Ding *et al.*, 2004] Ding, L., Kolari, P., Ganjugunte, S., Finin, T. et Joshi, A. (2004). Modeling and evaluating trust network inference. *In Proceedings of The Workshop on Deception, Fraud and Trust in Agent Societies at The Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2004)*, pages 21–32.

[Dividino *et al.*, 2009] Dividino, R., Sizov, S., Staab, S. et Schueler, B. (2009). Querying for provenance, trust, uncertainty and other meta knowledge in RDF. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):204–219.

[Dreyer *et al.*, 2010] Dreyer, L., Grant, M. et Cae, M. (2010). *Open Community: A Little Book of Big Ideas for Associations Navigating the Social Web*. SocialFish.

[Dubois *et al.*, 1991] Dubois, D., Prade, H. et Rossazza, J. P. (1991). Vagueness, typicality, and uncertainty in class hierarchies. *International Journal of Intelligent Systems*, 6(2):167–183.

[Dufour-Lussier, 2014] Dufour-Lussier, V. (2014). *Reasoning with qualitative spatial and temporal textual cases*. Thèse de doctorat. Thèse de doctorat dirigée par Lieber, J. et Le Ber, F. Informatique Université de Lorraine 2014.

[Feigenbaum, 1984] Feigenbaum, E. (1984). Knowledge engineering. *Annals of the New York Academy of Sciences*, 426(1):91–107.

[Feldman et Serrano, 2006] Feldman, A. et Serrano, R. (2006). *Welfare economics and social choice theory*. Springer Science & Business Media.

[Friedman *et al.*, 1995] Friedman, M., Ming, M. et Kandel, A. (1995). On the theory of typicality. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 03(02):127–142.

[Gaillard, 2011] Gaillard, E. (2011). Les systèmes informatiques fondés sur la confiance: un état de l'art. *Rapport interne*.

[Gaillard *et al.*, 2014a] Gaillard, E., Infante-Blanco, L., Lieber, J. et Nauer, E. (2014a). Tuuurbine: A generic cbr engine over rdfs. *In Case-Based Reasoning Research and Development*, pages 140–154. Springer International Publishing.

[Gaillard *et al.*, 2013a] Gaillard, E., Lieber, J., Naudet, Y. et Nauer (2013a). Case-based reasoning on e-community knowledge. *In 21st International Conference on Case-Based Reasoning (ICCBR-13)*, pages 114–128, Saragota Springs, NY, USA.

[Gaillard *et al.*, 2013b] Gaillard, E., Lieber, J., Naudet, Y. et Nauer, E. (2013b). Raisonner sur des connaissances provenant d'une e-communauté. *In Ingénierie des connaissances 2013 (IC-13)*, Lille, France.

[Gaillard *et al.*, 2011] Gaillard, E., Lieber, J. et Nauer, E. (2011). Adaptation knowledge discovery for cooking using closed itemset extraction. *In The Eighth International Conference on Concept Lattices and their Applications (CLA-11)*, pages 87–99.

[Gaillard *et al.*, 2014b] Gaillard, E., Lieber, J., Nauer, E. et Cordier, A. (2014b). How case-based reasoning on e-community knowledge can be improved thanks to knowledge reliability. *In 22nd International Conference on Case-Based Reasoning (ICCBR-14)*, pages 155–169, Cork, Ireland.

[Gaillard *et al.*, 2015a] Gaillard, E., Lieber, J., Nauer, E. et Cordier, A. (2015a). How Improving Case Retrieval Thanks to Typicality. *In 23rd International Conference on Case-Based Reasoning (ICCBR-15)*, pages 114–128, Frankfort, Germany.

[Gaillard *et al.*, 2015b] Gaillard, E., Lieber, J., Nauer, E. et Cordier, A. (2015b). How Managing the Knowledge Reliability Improves the Results of a Reasoning Process. *In 16th European Conferences on Knowledge Managements (ECKM-15)*, pages 114–128, Udine, Italy.

[Gambetta, 2000] Gambetta, D. (2000). Can we trust trust. *In* Gambetta, D., éditeur : *Trust Making and Breaking Cooperative Relations*, chapitre 13, pages 213–237. Basil Blackwell, Oxford.

[Giard et Roy, 1985] Giard, V. et Roy, B. (1985). *Méthodologie multicritère d'aide à la décision*. Editions Economica.

[Gil et Ratnakar, 2002] Gil, Y. et Ratnakar, V. (2002). Trusting information sources one citizen at a time. *In The Semantic Web—ISWC 2002*, pages 162–176. Springer.

[Gisselbrecht *et al.*, 2015] Gisselbrecht, T., Denoyer, L., Gallinari, P. et Lamprier, S. (2015). Apprentissage en temps réel pour la collecte d'information dans les réseaux sociaux. *In CORIA*, pages 7–22.

[Gleverdon et Cleverdon, 1962] Gleverdon, C. et Cleverdon, C. (1962). Report on the testing and analysis of an investigation into the comparative efficiency of indexing systems. *Technical report*.

[Golbeck, 2005] Golbeck, J. (2005). *Computing and applying trust in web-based social networks*. Thèse de doctorat, University of Maryland.

[Golbeck et Mannes, 2006] Golbeck, J. et Mannes, A. (2006). Using trust and provenance for content filtering on the semantic web. *In MTW*.

[Grandison et Sloman, 2000] Grandison, T. et Sloman, M. (2000). A survey of trust in internet applications. *Communications Surveys & Tutorials, IEEE*, 3(4):2–16.

[Groth *et al.*, 2006] Groth, P., Jiang, S., Miles, S., Munroe, S., Tan, V., Tsasakou, S. et Moreau, L. (2006). An architecture for provenance systems. *Technical report, University of Southampton*.

[Hagel et Armstrong, 1997] Hagel, J. et Armstrong, A. (1997). *Net gain: Expanding markets through virtual communities*. Harvard Business Press.

[Halpern et Moses, 1985] Halpern, J. et Moses, Y. (1985). Towards a theory of knowledge and ignorance: Preliminary report. *In Logics and models of concurrent systems*, pages 459–476. Springer.

[Hammond, 1990] Hammond, K. (1990). Explaining and repairing plans that fail. *Artificial intelligence*, 45(1):173–228.

[Hampton, 1979] Hampton, J. A. (1979). Polymorphous concepts in semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 18(4):441–461.

[Harmon, 1996] Harmon, D. (1996). *Overview of the third text retrieval conference (Trec-3)*. DIANE Publishing Company.

[Hartig et Zhao, 2009] Hartig, O. et Zhao, J. (2009). Using web data provenance for quality assessment. *In CEUR Workshop Proceedings*.

[Haydar *et al.*, 2014] Haydar, C., R., A. et Boyer, A. (2014). Comparing Local, Collective, and Global Trust Models. *International Journal On Advances in Life Sciences*, 6(1&2):10.

[Hume, 2012] Hume, D. (2012). *A treatise of human nature*. Courier Corporation.

[Jøsang, 2001] Jøsang, A. (2001). A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(03):279–311.

[Jøsang et Ismail, 2002] Jøsang, A. et Ismail, R. (2002). The beta reputation system. pages 324–337.

[Jøsang *et al.*, 2006] Jøsang, A., Marsh, S. et Pope, S. (2006). Exploring different types of trust propagation. *In Trust management*, pages 179–192. Springer.

[Jøsang et Pope, 2005] Jøsang, A. et Pope, S. (2005). Semantic constraints for trust transitivity. *In Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43*, pages 59–68. Australian Computer Society, Inc.

[Jøsang *et al.*, 2007] Jøsang, A., Ismail, R. et Boyd, C. (2007). A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644.

[Juran, 1988] Juran, J. (1988). *Juran on planning for quality*. Free Press New York.

[Kamvar *et al.*, 2003] Kamvar, S., Schlosser, M. et Garcia-Molina, H. (2003). The Eigentrust algorithm for reputation management in P2P networks. WWW '03, page 640, Budapest, Hungary. ACM Press.

[Kant, 1869] Kant, I. (1869). *Critique de la raison pure*, volume 1. Germer-Baillière.

[Knap et Mlynková, 2011] Knap, T. et Mlynková, I. (2011). Revealing beliefs influencing trust between members of the czech informatics community. *In Proceedings of the Third international conference on Social informatics*, pages 226–239.

[Lawrence, 1995] Lawrence, T. (1995). Power and resources in an organizational community. *In Academy of Management Best Papers Proceedings*, pages 251–255.

[Leake et Whitehead, 2007] Leake, D. et Whitehead, M. (2007). Case provenance: The value of remembering case sources. *In 7th international conference on Case-Based Reasoning: Case-Based Reasoning Research and Development (ICCBR-07)*, pages 194–208, Belfast, Northern Ireland, UK.

[Lenz et Burkhard, 1996] Lenz, M. et Burkhard, H. (1996). Case retrieval nets: Basic ideas and extensions. *In KI-96: Advances in Artificial Intelligence*, pages 227–239. Springer.

[Lesani et Bagheri, 2006] Lesani, M. et Bagheri, S. (2006). *Canadian Semantic Web*, chapitre Applying and Inferring Fuzzy Trust in Semantic Web Social Networks, pages 23–43. Springer US, Boston, MA.

[Levien, 2009] Levien, R. (2009). Attack-resistant trust metrics. *In Computing with Social Trust*, pages 121–132. Springer.

[Lewis *et al.*, 2004] Lewis, D., Yang, Y., Rose, T. et Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397.

[Likert, 1932] Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):1–55.

[Luhmann, 1979] Luhmann, N. (1979). *Trust and power: two works*. UMI Books on Demand. Wiley.

[Luhmann, 1990] Luhmann, N. (1990). Familiarity, confidence, trust: Problems and alternatives. *In Trust*, volume Chapter 6, pages 94–107. Gambetta, D. Basil Blackwell, Oxford.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[Marek et Truszczyński, 1991] Marek, W. et Truszczyński, M. (1991). Autoepistemic logic. *Journal of the ACM (JACM)*, 38(3):587–618.

[Marsh, 1994] Marsh, S. (1994). *Formalising Trust as a Computational Concept*. Thèse de doctorat, University of Stirling.

[Massa et Avesani, 2004] Massa, P. et Avesani, P. (2004). Trust-aware collaborative filtering for recommender systems. *In On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pages 492–508. Springer.

[McBride, 2004] McBride, B. (2004). The resource description framework (rdf) and its vocabulary description language rdfs. *In Handbook on ontologies*, pages 51–65. Springer Berlin Heidelberg.

[McKnight et Chervany, 1996] McKnight, D. et Chervany, N. (Management Information Systems Research Center, 1996). The meanings of trust. *Technical Report MISRC Working Paper Series 96-04, University of Minnesota*.

[Medin et Schaffer, 1978] Medin, D. et Schaffer, M. (1978). Context theory of classification learning. *Psychological Review*, 85(3):207–238.

[Mille, 1998] Mille, A. (1998). Associer expertise et expérience pour assister les tâches de l'utilisateur. *Habilitation à diriger des recherches, Université Claude Bernard, Lyon1*.

[Moturu et Liu, 2011] Moturu, S. et Liu, H. (2011). Quantifying the trustworthiness of social media content. *Distributed and Parallel Databases*, 29(3):239–260.

[Mui *et al.*, 2002] Mui, L., Mohtashemi, M. et Halberstadt, A. (2002). A computational model of trust and reputation. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 00(c):2431–2439.

[Naudet *et al.*, 2010] Naudet, Y., Latour, T., Vidou, G. et Djaghloul, Y. (2010). Towards a novel approach for high-stake decision support system based on community contributed knowledge base. *In 10th International Conference on Intelligent Systems Design and Applications (ISDA-10)*, pages 730–736.

[Newell, 1982] Newell, A. (1982). The knowledge level. *Artificial intelligence*, 18(1):87–127.

[Nietzsche, 1878] Nietzsche, F. (1968 [1878]). Humain trop humain. *Paris : Galimard, vol 1*.

[Nute, 2001] Nute, D. (2001). Defeasible logic. *In Web knowledge management and decision support*, pages 151–169. Springer.

[Pal et Counts, 2011] Pal, A. et Counts, S. (2011). Identifying topical authorities in microblogs. *In Proceedings of the fourth ACM international conference on Web search and data mining*, pages 45–54.

[Pinheiro da Silva *et al.*, 2003] Pinheiro da Silva, P., McGuinness, D. et Mccool, R. (2003). Knowledge provenance infrastructure. *IEEE Data Engineering Bulletin*, 26(4):26–32.

[Pitrat, 1990] Pitrat, J. (1990). *Métaconnaissance: futur de l'intelligence artificielle*. Hermes Paris.

[Plaza, 1995] Plaza, E. (1995). Cases as terms: A feature term approach to the structured representation of cases. *In Case-Based Reasoning Research and Development*, pages 265–276. Springer.

[Prud'hommeaux et Seaborne, 2006] Prud'hommeaux, E. et Seaborne, A. (2006). SPARQL Query Language for RDF. Rapport technique.

[Putnam *et al.*, 1994] Putnam, R., Leonardi, R. et Nanetti, R. Y. (1994). *Making democracy work: Civic traditions in modern Italy*. Princeton university press.

[Qin et Cunningham, 2012] Qin, X. et Cunningham, P. (2012). Assessing the quality of wikipedia pages using edit longevity and contributor centrality. *arXiv preprint arXiv:1206.2517*.

[Quijano-Sánchez *et al.*, 2012] Quijano-Sánchez, L., Bridge, D., Díaz-Agudo, B. et Recio-García, J. (2012). Case-based aggregation of preferences for group recommenders. *In Case-Based Reasoning Research and Development*, pages 327–341. Springer Berlin Heidelberg.

[Quijano-Sánchez *et al.*, 2011] Quijano-Sánchez, L., Recio Garcia, J. et Díaz-Agudo, B. (2011). Using personality to create alliances in group recommender systems. *In* Ram, A. et Wiratunga, N., éditeurs : *Case-Based Reasoning Research and Development*, volume 6880 de *Lecture Notes in Computer Science*, pages 226–240. Springer Berlin Heidelberg.

[Ramamoorthy et Wah, 1989] Ramamoorthy, C. et Wah, B. (1989). Knowledge and data engineering. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):9–16.

[Ramchurn *et al.*, 2004] Ramchurn, S., Jennings, N., Sierra, C. et Godo, L. (2004). Devising a trust model for multi-agent interactions using confidence and reputation. *Applied Artificial Intelligence*, 18(9-10):833–852.

[Rao *et al.*, 1995] Rao, A., Georgeff, M. *et al.* (1995). BDI agents: From theory to practice. *In ICMAS 95*, pages 312–319.

[Redman, 1997] Redman, T. (1997). *Data Quality for the Information Age*. Artech House, Inc.

[Rees, 1965] Rees, A. (1965). The aslib-cranfield test of the western reserve university indexing system for metallurgical literature: A review of the final report. *American Documentation*, 16(2):73–76.

[Rempel et Holmes, 1986] Rempel, J. et Holmes, J. (1986). How do I trust thee? *Psychology Today*, 20(2):28–34.

[Resnick et Zeckhauser, 2002] Resnick, P. et Zeckhauser, R. (2002). Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. *The Economics of the Internet and E-commerce*, 11(2):23–25.

[Ricci *et al.*, 2010] Ricci, F., Rokach, L., Shapira, B. et Kantor, P. (2010). *Recommender Systems Handbook*. Springer.

[Richards, 2009] Richards, D. (2009). A social software/web 2.0 approach to collaborative knowledge engineering. *Information Sciences*, 179(15):2515–2523.

[Richardson *et al.*, 2003] Richardson, M., Agrawal, R. et Domingos, P. (2003). Trust management for the semantic web. *In The Semantic Web-ISWC 2003*, pages 351–368. Springer.

[Richter, 1995] Richter, M. (1995). The knowledge contained in similarity measures. Invited talk at the first International Conference on Case-Based Reasoning.

[Riesbeck et Schank, 1889] Riesbeck, C. et Schank, R. (1889). *Inside Case-Based Reasoning*. Artificial Intelligence Series.

[Romero *et al.*, 2011] Romero, D., Galuba, W., Asur, S. et Huberman, B. (2011). Influence and passivity in social media. *In Machine learning and knowledge discovery in databases*, pages 18–33. Springer.

[Rosch et Mervis, 1975a] Rosch, E. et Mervis, C. (1975a). Family resemblances: Studies in the internal structure of categories. *Cognitive psychology*, 7(4):573–605.

[Rosch et Mervis, 1975b] Rosch, E. et Mervis, C. B. (1975b). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, 7(4):573–605.

[Rosch, 1973] Rosch, E. H. (1973). On the internal structure of perceptual and semantic categories. *In* Moore, T. E., éditeur : *Cognitive Development and the Acquisition of Language*, pages 111–144. Academic.

[Rose *et al.*, 2002] Rose, T., Stevenson, M. et Whitehead, M. (2002). The reuters corpus volume 1-from yesterday's news to tomorrow's language resources. *In Third International Conference on Language Resources and Evaluation (LREC-02)*, pages 827–832.

[Roth et Shoben, 1983] Roth, E. et Shoben, E. (1983). The effect of context on the structure of categories. *Cognitive psychology*, 15(3):346–378.

[Rothaermel et Sugiyama, 2001] Rothaermel, F. et Sugiyama, S. (2001). Virtual internet communities and commercial success: individual and community-level theory grounded in the atypical case of timezone. com. *Journal of management*, 27(3):297–312.

[Saaya *et al.*, 2011] Saaya, Z., Smyth, B., Coyle, M. et Briggs, P. (2011). Recommending case bases: Applications in social web search. *In Case-Based Reasoning Research and Development*, pages 274–288. Springer Berlin Heidelberg.

[Sabater et Sierra, 2002] Sabater, J. et Sierra, C. (2002). Reputation and social network analysis in multi-agent systems. *In Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 475–482.

[Sanchez, 1984] Sanchez, E. (1984). Solution of fuzzy equations with extended operations. *Fuzzy sets and Systems*, 12(3):237–248.

[Schank et Abelson, 1977] Schank, R. et Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures*. The Artificial Intelligence Series. Lawrence Erlbaum Associates.

[Schmidt et Gierl, 1996] Schmidt, R. et Gierl, L. (1996). The roles of prototypes in medical case-based reasoning systems. *In 4th German Workshop on CBR–System Development and Evaluation, Humbolt University, Informatik-Berichte, Berlin*, pages 207–216.

[Shankar et Watts, 2003] Shankar, G. et Watts, S. (2003). A relevant, believable approach for data quality assessment. *In IQ*, pages 178–189.

[Shankaranarayan *et al.*, 2003] Shankaranarayan, G., Ziad, M. et Wang, R. (2003). Managing data quality in dynamic decision environments: An information product approach. *Journal of Database Management*, 14(4):14–32.

[Smets et Kennes, 1994] Smets, P. et Kennes, R. (1994). The transferable belief model. *Artificial intelligence*, 66(2):191–234.

[Smith et Medin, 1981] Smith, E. et Medin, D. (1981). *Categories and concepts*. Cognitive science series. Harvard University Press.

[Smith *et al.*, 1988] Smith, E., Osherson, D., Rips, L. et Keane, M. (1988). Combining prototypes: A selective modification model. *Cognitive science*, 12(4):485–527.

[Smyth, 1996] Smyth, B. (1996). *Case-Based Design*. Thèse de doctorat, Trinity College, University of Dublin.

[Smyth et Keane, 1995] Smyth, B. et Keane, M. (1995). Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. *In Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95) - Volume 1*, pages 377–382, San Francisco, CA, USA.

[Sztompka, 1999] Sztompka, P. (1999). *Trust: A sociological theory*. Cambridge University Press.

[Tan, 2004] Tan, W. (2004). Research problems in data provenance. *IEEE Data Eng. Bull.*, 27(4):45–52.

[Tversky, 1977] Tversky, A. (1977). Features of similarity. *Psychological review*, 84(4):327–352.

[Wang, 1998] Wang, R. (1998). A product perspective on total data quality management. *Communications of the ACM*, 41(2):58–65.

[Wang et Strong, 1996] Wang, R. et Strong, D. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33.

[Weber *et al.*, 2005] Weber, R., Ashley, K. et Brüninghaus, S. (2005). Textual case-based reasoning. *The Knowledge Engineering Review*, 20(03):255–260.

[Weber-Lee *et al.*, 1996] Weber-Lee, R., Barcia, R. M., Martins, A. et Pacheco, R. C. (1996). Using typicality theory to select the best match. *In* Smith, I. et Faltings, B., éditeurs : *Advances in Case-Based Reasoning*, volume 1168 de *Lecture Notes in Computer Science*, pages 445–459. Springer Berlin Heidelberg.

[Wiederhold, 1984] Wiederhold, G. (1984). Knowledge and database management. *IEEE Software*, 1(1):63.

[Wilcoxon et Wilcox, 1964] Wilcoxon, F. et Wilcox, R. (1964). *Some rapid approximate statistical procedures*. Lederle Laboratories.

[Williams, 1988] Williams, B. (1988). Formal Structures and Social Reality. *In* Gambetta, D., éditeur : *Making Sense of Humanity*, pages 3–13. Cambridge University Press.

[Yeung et Leung, 2006] Yeung, C. et Leung, H. (2006). Ontology with likeliness and typicality of objects in concepts. *In* Embley, D. W., Olivé, A. et Ram, S., éditeurs : *Conceptual Modeling - ER 2006*, volume 4215 de *Lecture Notes in Computer Science*, pages 98–111. Springer Berlin Heidelberg.

[Zacharia et Maes, 2000] Zacharia, G. et Maes, P. (2000). Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14(9):881–907.

[Zadeh, 1982] Zadeh, L. (1982). A note on prototype theory and fuzzy sets. *Cognition*, 12(3): 291–297.

[Zaveri *et al.*, 2015] Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J. et Auer, S. (2015). Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93.

[Zeng *et al.*, 2006] Zeng, H., Alhossaini, M. A., Ding, L., Fikes, R. et McGuinness, D. (2006). Computing trust from revision history. *In Proceedings of the 2006 International Conference on Privacy Security and Trust Bridge the Gap Between PST Technologies and Business Services PST 06*.

[Ziegler et Lausen, 2004] Ziegler, C. et Lausen, G. (2004). Spreading activation models for trust propagation. *In International Conference on e-Technology, e-Commerce and e-Service (EEE-04)*, pages 83–97. IEEE.

Résumé

Cette thèse propose deux approches pour améliorer la qualité des réponses d'un système de raisonnement à partir de cas (RÀPC) utilisant des connaissances produites par une communauté en ligne. La première approche concerne la mise en œuvre d'un modèle permettant de gérer la fiabilité des connaissances produites par la communauté sous la forme d'un score. Ce score de fiabilité est utilisé d'une part pour filtrer les connaissances non fiables afin qu'elles ne soient pas utilisées par le système de RÀPC et d'autre part pour classer les réponses retournées par le système. La deuxième approche concerne la représentation de la typicalité entre sous-classes et classes dans une organisation hiérarchique. La typicalité est alors utilisée pour réorganiser les connaissances hiérarchiques utilisées par le système de RÀPC. L'apport de ces deux approches a été évalué dans le cadre de eTaaable, un système de RÀPC qui adapte des recettes de cuisine en utilisant des connaissances produites par une communauté en ligne. L'évaluation montre que la gestion de la fiabilité des connaissances produites par la communauté améliore la qualité des réponses retournées par eTaaable. De même, l'évaluation montre que l'utilisation par eTaaable des hiérarchies des connaissances réorganisées en exploitant la typicalité améliore également la qualité des réponses.

Mots-clés: raisonnement à partir de cas, communauté en ligne, fiabilité, typicalité.

Abstract

This research work presents two approaches to improve the quality of the results returned by a case-based reasoning system (CBR) exploiting knowledge produced by an e-community. The first approach relies on a new model to manage the trustworthiness of the knowledge produced by the e community. In this model, the trustworthiness is represented through a score which is used to filter untrustworthy knowledge so that the CBR system will not use it anymore. Moreover, the trustworthiness score is also used to rank the CBR results. The second approach addresses the issue of representing the typicality between subclasses and classes in a hierarchy. The typicality is used to change the hierarchical organization used by the CBR system. Both approaches have been evaluated in the framework of eTaaable, a CBR system which adapts cooking recipes using knowledge coming from an e-community. The evaluations show that managing the trustworthiness of the knowledge produced by an e-community improves the quality of the results returned by eTaaable. The evaluations also shows that eTaaable returns also better results when using knowledge reorganized according to typicality.

Keywords: case-based reasoning, e-community, trustworthiness, typicality.