Classification de messages échangés entre développeurs débutants dans le cadre d'une activité de pair programming distribué

Chloé Boireau-Devier[1], Sébastien Jolivet[2,3][0000−0003−3915−8465], José Colin[1][0009−0003−5330−848X], and Julien Broisin[1][0000−0001−8713−6282]

[1] Univ Toulouse, Toulouse INP, CNRS, IRIT, Toulouse, France,
julien.broisin@irit.fr
[2] IUFE & TECFA, Université de Genève, Genève, Suisse,
sebastien.jolivet@unige.ch
[3] Sorbonne Université, CNRS, LIP6, 4 place Jussieu, 75005 Paris, France

Résumé. Cet article propose une méthode de classification des messages textuels échangés entre des développeurs novices engagés dans une activité de pair programming distribué. En nous appuyant sur une revue de la littérature consacrée aux rétroactions dans le contexte de l'apprentissage (de la programmation), nous développons et affinons un système de classification utilisant trois attributs fondamentaux (le contenu, la forme et la nature du message) dont les valeurs associées sont extraites des éléments les plus prégnants des travaux étudiés. La recherche détaille un processus itératif d'accord inter-évaluateurs visant à valider et à stabiliser la classification finale, qui est conçue pour être applicable aux messages courts et instantanés typiques de ces environnements de programmation collaborative. À partir de cette grille de classification, le travail propose également une méthode d'annotation semi-automatique fondée sur un grand modèle de langue pour alléger la tâche fastidieuse d'étiquetage d'un vaste corpus de messages. Cette recherche représente la première brique pour, à terme, analyser l'impact des différents types de messages sur l'évolution du code produit par les apprenants.

Mots-clés : Pair programming distribué · Rétroaction · Classification

1 Introduction et questions de recherche

Le *pair programming* est une méthode de travail [8] qui implique deux acteurs, le pilote et le navigateur, qui doivent résoudre un même problème. Le premier produit le code tandis que le second ne peut pas agir directement sur le code mais peut donner des conseils au premier. Nous nous intéressons dans cet article au pair programming distribué pour l'apprentissage de la programmation, c'est-à-dire une situation où les deux acteurs réalisent une activité de pair programming à distance à l'aide d'une plateforme dédiée. Dans cet environnement, la communication entre les acteurs délocalisés est assurée au moyen de messages textuels [3].

Dans le contexte du pair programming distribué, un enjeu majeur lié à l'apprentissage réside dans la manière d'optimiser la collaboration entre les apprenants dans la plateforme. Cet enjeu peut être abordé sous différents angles (e.g., trouver les meilleurs appariements, fournir un étayage durant l'activité, guider les interventions de l'enseignant) et selon des temporalités différentes (e.g., avant le lancement du travail ou durant le travail).

Les travaux présentés dans cet article sont le fruit d'un stage de licence réalisé par la première auteure de cet article. Son objet était de proposer une classification des messages textuels échangés au sein des groupes d'apprenants lors d'une expérimentation menée à l'Université de La Réunion dans un cours de L1 informatique. L'objectif final, non développé dans cet article, est de pouvoir analyser les liens entre les types de messages échangés et l'évolution des codes produits par les apprenants. Dans cette contribution nous nous concentrons sur la classification que nous avons construite en nous appuyant sur une étude de la littérature. La section 4 permet d'esquisser l'exploitation d'un grand modèle de langue pour une classification (semi-) automatique des messages.

Le premier choix réalisé pour aborder ce travail a été de considérer les messages échangés entre pilote et navigateur comme des rétroactions mutuelles. Il s'agit d'une approche qui est généralement classée dans les pratiques d'évaluation par les pairs (*peer assessment*) et nommée rétroaction par les pairs (*peer feedbacks*) [17]. Cependant, dans la littérature, une rétroaction est le plus souvent un message délivré par un enseignant en direction d'un apprenant en réaction à une action de ce dernier. Dans le contexte du pair programming, l'asymétrie enseignant/apprenant habituelle est absente puisque pilote et navigateur sont tous deux apprenants. Il nous a cependant paru raisonnable de nous appuyer sur les travaux relatifs aux rétroactions, et en particulier aux rétroactions dans le cadre de l'apprentissage de la programmation, dans la mesure où notre objectif est de caractériser le contenu du message et non pas d'évaluer sa "véracité" ou son adéquation au contexte. Pour caractériser son contenu, il n'est ainsi pas problématique que le message contienne des éléments non adaptés à la situation (ce qui peut par ailleurs se produire même dans la relation asymétrique enseignant /apprenant). Par ailleurs, les messages font généralement suite à une action de l'autre acteur, et nous n'avons pas identifié de cadre spécifique d'analyse de ce type d'échanges. Ainsi, nous exploitons les nombreuses classifications des rétroactions ont été proposées dans la littérature [22, 4], avec des niveaux de précisions et des finalités diverses [11]. Certaines sont déjà en lien avec l'activité de programmation [12, 10, 6].

La section suivante présente un bref état de l'art des différentes classifications de rétroactions sur lequel nous nous appuyons pour proposer dans la section 3 les attributs et valeurs nous permettant de classifier les messages échangés entre pilote et navigateur dans la plateforme. Dans la section 4, nous présentons la démarche que nous avons adoptée pour réaliser une annotation automatique des messages à partir de notre classification, et pour évaluer la qualité des annotations ainsi obtenues. La conclusion est l'occasion de revenir sur les prolongements à venir de ce premier travail.

2 Quelques éléments sur les classifications des rétroactions

Dans cette section nous reprenons quelques éléments essentiels issus de l'état de l'art sur les classifications des rétroactions. Pour cela nous avons étudié quatorze articles, proposant tous des critères de catégorisation des rétroactions, certains étant généralistes [21, 23, 16, 18, 4, 13, 19, 14, 9, 20, 24], d'autres se focalisant sur le contexte de la programmation [12, 10, 6].

Chacun de ces articles a été abordé de manière à extraire les types et définitions des rétroactions qu'ils proposent, l'objectif étant d'établir un "catalogue" de "types" de rétroaction, mais aussi de pouvoir identifier ceux étant identiques ou proches au-delà de différences sémantiques. À titre d'illustration, le Tableau 1 donne le résultat de notre analyse pour les travaux de Leibold & Schwarz [16] portant sur la fourniture de rétroactions dans un contexte d'appren-tissage en ligne.

Table 1. Exemple de l'analyse réalisée sur les travaux de Leibold & Schwarz [16]

Type de rétroaction	**Description**
Corrective feedback	Rétroaction en rapport avec les exigences de la tâche à effectuer (ex : "on demandait x mais tu ne l'as pas utilisé")
Epistemic feedback	Guidage ou question pour une plus grande réflexion, explication ou clarification (ex : "explique comment ce concept se rapporte à ce que tu veux montrer")
Suggestive feedback	Conseil, idée, élargissement pour améliorer une idée (ex : "donner un exemple de x après sa définition le rendrait plus clair")
Epistemic and suggestive feedback	Guidage ou question pour développer ou suggérer une amélioration
Positive (negative) feedback tone	La rétroaction est présentée de manière positive (négative), encourageante (décourageante)
Specific/vague feedback	Une rétroaction spécifique est une rétroaction claire, détaillée, qui permet de transmettre des informations à un apprenant de manière précise, par opposition à une rétroaction vague
Balanced or sandwiched feedback	Commentaire positif, puis point à améliorer, puis commentaire positif

Ce travail d'analyse de chaque article a été synthétisé dans un tableau décrivant :

- le contexte dans lequel les rétroactions sont utilisées ;
- le registre du message (texte, vidéo, etc.) ;
- le type d'interaction mis en jeu (élève-professeur, élève-machine) ;
- les définitions des rétroactions prises en compte.

Cette grille de synthèse nous a permis de comparer les différentes approches de la littérature et de regrouper celles ayant une sémantique similaire mais

désignées par des termes différents. Ainsi, nous avons identifié huit grandes catégories de rétroaction qui sont présentées dans la Figure 1, et qui sont mises en relation avec différents articles dans lesquels elles sont présentes. Nous présentons dans la section suivante comme nous exploitons cette catégorisation pour produire notre grille de description des messages échangés dans un contexte de Pair Programming Distribué (Tableau 2).

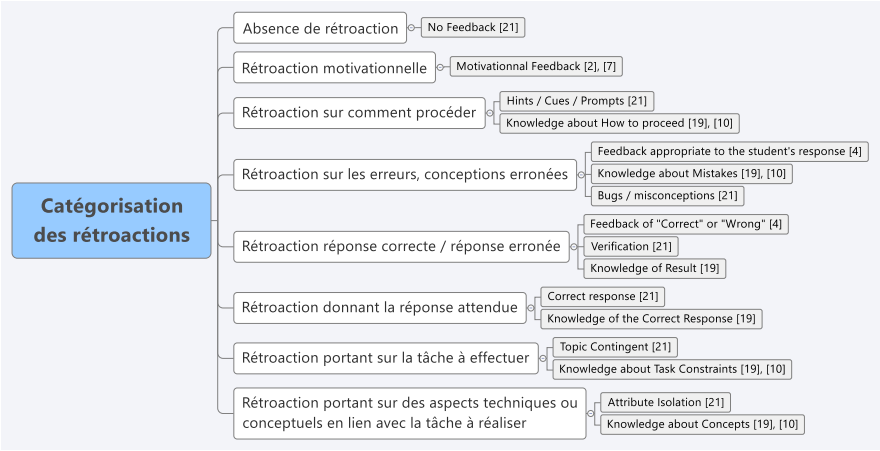

Fig. 1. Synthèse des analyses d'articles et catégorisation des rétroactions

3 Classification des rétroactions dans le cadre du pair programming

Dans cette section, nous présentons brièvement la plateforme de *distributed pair programming* (DPP) afin d'expliciter comment sont générés les messages étudiés. Puis, dans les sections suivantes, nous détaillons la classification des messages de DPP que nous avons définie. Pour cela nous explicitons notre méthode, de construction et de validation, et nous donnons les premiers résultats.

3.1 La plate-forme de pair programming

L'application de pair programming distribuée nous ayant permis de collecter des messages textuels est une application web dédiée à l'apprentissage du langage

Python [3]. L'interface proposée aux apprenants implémente toutes les fonctionnalités nécessaires à la mise en œuvre du pair programming distribué : un éditeur de code partagé, un système à base de rôle pour différencier le *Pilote* et le *Navigateur* et permettre les changements de rôle entre les apprenants, une console Python pour les tests et l'exécution du code, et une messagerie textuelle instantanée. Les changements de rôles ne sont pas contraints dans la plateforme, c'est-à-dire que les apprenants choisissent librement le moment où ils souhaitent inverser leur rôle. La plupart des interactions de l'utilisateur avec la plateforme sont collectées sous forme de traces horodatées, y compris les messages textuels échangés pendant les sessions de pair programming. Ce sont ces messages que nous souhaitons annoter.

3.2 Méthode

Ce travail de classification des messages échangés dans la plateforme entre les apprenants s'inscrit dans une problématique plus large d'analyse de l'effet de ces messages sur l'évolution du code produit. Il s'agit donc, dans un premier temps, de caractériser les messages selon certains critères, avant d'analyser les effets des différents types de messages (i.e., des différents types de rétroaction) sur l'évolution du code produit.

Notre objectif ici est donc de définir une liste d'attributs, et de valeurs associées, servant à annoter les messages. Cette liste doit nécessairement être accompagnée d'un *guide d'annotation* permettant son exploitation effective. Pour définir cette liste d'attributs et de valeurs, nous avons exploité à la fois les éléments de l'état de l'art présentés dans la section précédente, et les spécificités liées au contexte du pair programming distribué.

Aussi, afin de valider la robustesse de notre proposition de classification, nous avons adopté la méthode d'accords inter-juges suite à des annotations réalisées de manière indépendante par plusieurs annotateurs sur un sous-corpus de messages recueillis lors d'une expérimentation en contexte d'apprentissage réel. Ce travail a été mené en plusieurs itérations permettant d'améliorer à la fois les critères d'annotation (attributs et valeurs) et le guide associé. Les principaux résultats de cette démarche sont présentés ci-après.

3.3 Version initiale de la classification

Le travail réalisé dans notre revue de littérature a mis en avant l'absence de consensus pour exprimer la fonction ou le sens des rétroactions, des termes différents étant employés pour désigner la même intention. Nous avons aussi constaté qu'aucun des articles étudiés ne propose de réelle classification, ou ne décrit les rétroactions de manière exhaustive, surtout dans le contexte particulier du pair programming pour lequel les messages étudiés sont très courts, parfois segmentés, et délivrés instantanément par un pair.

Nous avons défini de premiers attributs : *source*, qui désigne l'émetteur du message, c'est-à-dire le *Pilote* ou le *Navigateur*, et *nombre de mots*, qui permet

de caractériser la longueur du message. Cependant, notre objectif est essentiellement de proposer des caractéristiques des contenus des messages, dont on peut faire l'hypothèse qu'elles peuvent impacter l'évolution du code produit. Nous définissons pour cela trois attributs :

1. le *contenu* permet de définir si le message est lié ou non à l'activité de programmation ;
2. la *forme* permet d'expliciter la fonction que l'émetteur semble associer à son message ;
3. la *nature* indique la catégorie (voir Figure 1) de rétroaction à laquelle appartient le message.

Ces attributs, les valeurs associées, et les éléments de la littérature dont ils sont issus, sont synthétisés dans le Tableau 2.

Concernant les valeurs de ces différents attributs, nous nous sommes appuyés dans la version initiale de la classification sur la majorité des valeurs identifiées dans la littérature afin de proposer une classification la plus fine et la plus exhaustive possible. À partir du guide d'annotation que nous avons élaboré sur la base de cette version initiale, quatre chercheurs ont alors annoté de manière indépendante un sous-corpus de 48 messages sélectionnés manuellement parmi le corpus entier pour leur intérêt, en particulier leur diversité apparente, et dont quelques exemples sont donnés dans le Tableau 3. Le degré d'accord entre les chercheurs a été déterminé à l'aide du Kappa de Fleiss [5], une version du Kappa de Cohen [2] pour plus de deux annotateurs, qui mesure la fidélité inter-évaluation en comparant les résultats des accords de classification par rapport au hasard. Le Kappa de Fleiss obtenu pour l'attribut *nature*, d'une valeur de $0,47$, témoigne d'un accord *modéré* selon [15]. Toutefois, les accords deux-à-deux ont atteint pour certaines pairs d'annotateurs des valeurs entre $0,52$ et $0,68$, témoignant d'un accord *substantiel*. Concernant les attributs *contenu* et *forme*, les Kappas de Fleiss ont respectivement atteint les valeurs de $0,30$ et $0,40$, reflétant un accord *équitable* et la difficulté pour les annotateurs de choisir entre une valeur et une autre pour ces deux attributs. Nous avons alors proposé différentes modifications à notre classification, tout en conservant suffisamment de précision au regard de nos objectifs.

3.4 Version finale de la classification

Dans la version finale de la classification, nous avons cherché à faciliter l'annotation des messages en effectuant quelques ajustements tels que la réduction du nombre de valeurs pour les attributs *contenu* et *forme* en ne retenant que les valeurs les plus fréquemment utilisées dans la littérature, ou encore le raffinement des règles d'annotation.

Table 2. Attributs fondamentaux et valeurs associées de la classification

Attribut	**Valeurs initiales**	**Valeurs finales**
Contenu	- Concepts clés de la tâche - Concepts clés de programmation - Demande de passage de la main - Mots clés Python - Pas de contenu utile au pair programming - Autre	- Relatif à la programmation - Non relatif à la programmation et relatif au rôle - Non relatif à la programmation, autre
Forme	- Instruction - Question - Correction - Explication - Encouragement - Description - Validation - Autre	- Instruction [21, 12, 16, 20, 23, 9, 6, 11] - Explication [21, 12, 16, 20, 23, 9, 6, 11] - Validation [21, 23, 6, 4, 14, 9, 25, 20] - Null
Nature	- Validité de la réponse - Réponse correcte - Connaissances sur la tâche - Connaissances sur les concepts - Informations sur comment procéder - Erreurs, idées fausses - Rétroaction motivationnelle - Demande d'aide - No feedback	- Validité de la réponse [21, 23, 6, 4, 14, 9, 25, 20] - Réponse correcte [21, 23, 25, 6] - Connaissances sur la tâche [21, 23, 9, 12, 16, 20] - Connaissances sur les concepts [21, 23, 20, 12, 14, 4, 25] - Informations sur comment procéder [21, 12, 23, 11, 16] - Erreurs, idées fausses [23, 21, 12, 25, 20, 6] - Rétroaction motivationnelle [4, 9] - Demande d'aide - No feedback [23]

Pour l'attribut *contenu*, nous avons regroupé les valeurs liées à la tâche à réaliser, à la programmation ou aux mots clés du langage Python dans l'unique valeur "Relatif à la programmation". En effet, il nous a semblé préférable de ne pas faire de distinctions car il est difficile, lorsqu'un message est analysé indépendamment des autres, d'affirmer qu'il est en lien direct plutôt avec la tâche à réaliser, la programmation en général, ou le langage Python en particulier. Les deux autres valeurs permettent de décrire les messages qui ne sont pas liés à la programmation, en distinguant ceux qui sont relatifs aux rôles de *Pilote* et *Navigateur* spécifiques au pair programming, et ceux qui sont sans rapport ("Allons manger" par exemple).

Table 3. Exemples de messages et leur classification

N°	Message	Classification
1	"tu pense si je fais un for i in range avec un for j in range dedans ça peut marcher?"	*Contenu* : relatif à la programmation *Forme* : explication *Nature* : demande d'aide
2	"Faut regardé si y'a un clé avec une valeur négative"	*Contenu* : relatif à la programmation *Forme* : instruction *Nature* : informations sur comment procéder
3	"j'essaye encore"	*Contenu* : non relatif à la programmation, autre *Forme* : null *Nature* : no feedback
4	"Tu peux prendre la main si tu veux"	*Contenu* : non relatif à la programmation et relatif au rôle *Forme* : null *Nature* : no feedback
5	"et tu as raison pour le tableau vide"	*Contenu* : relatif à la programmation *Forme* : validation *Nature* : validité de la réponse

Concernant l'attribut *forme*, nous n'avons conservé dans la classification finale que les valeurs largement utilisées dans une majorité des articles que nous avons étudiés. La valeur *instruction* permet de désigner un message contenant une directive pour réaliser une tâche de programmation (idée d'ordre). La valeur *explication* est utilisée pour les messages qui fournissent une justification, formulent un constat ou proposent une idée. La valeur *validation* correspond aux messages validant une proposition ou une idée formulée par un pair. Enfin, la valeur *null* est appliquée à tout autre message.

Finalement, nous avons complété notre guide d'annotation avec deux règles générales permettant de lever certaines ambiguïtés :

- Si le *contenu* du message n'est pas relatif à la programmation, sa *forme* prend la valeur *null*.
- Si le message correspond à du code source, son *contenu* est naturellement fixé à *relatif à la programmation*, et sa *forme* est fixée à *instruction*.

La première est motivée par notre intérêt, qui porte sur l'impact des messages sur le code et non pas, par exemple, sur les dynamiques de collaboration. La seconde est liée au fait qu'on interprète que si le navigateur fournit un code "complet" dans la conversation, il propose une instruction implicite qui est de l'intégrer au code produit par le pilote.

À partir de ces nouveaux ensembles de valeurs et du nouveau guide d'annotation, les quatre annotateurs ont à nouveau annoté le même sous-corpus de 48 messages. Le Kappa de Fleiss obtenu est de $0,71$ pour l'attribut *forme*, dénotant un accord *substantiel*, avec des Kappa de Cohen entre certaines paires d'annotateurs reflétant des accords *presque parfait* (voir l'histogramme à gauche

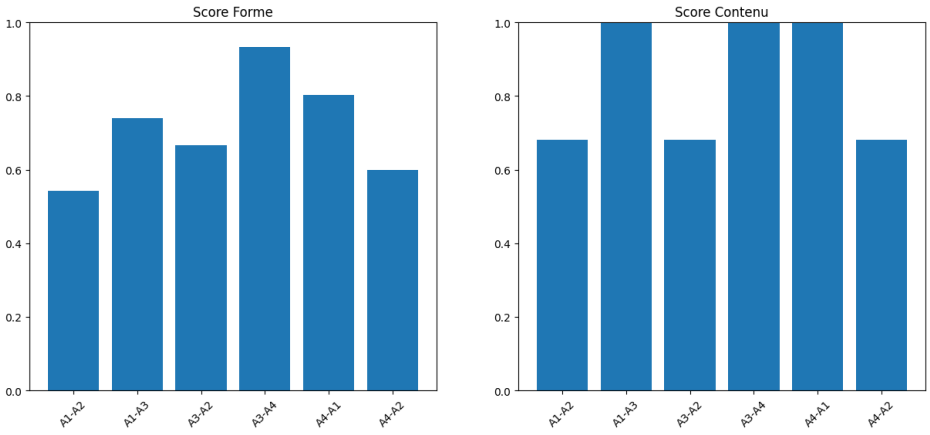

Fig. 2. Kappa de Cohen pour les attributs *forme* et *contenu* de la classification finale

de la Figure 2). Concernant l'attribut *contenu*, le Kappa de Fleiss obtenu est passé de $0,59$ pour quatre annotateurs à $0,82$, reflétant un accord *presque parfait*. L'histogramme à droite de la Figure 2 détaille les Kappa de Cohen obtenus deux-à-deux, où 3 paires d'annotateurs ont obtenu un accord parfait pour l'ensemble des 48 messages annotés. Ces différents scores nous ont ainsi permis de stabiliser et de valider notre classification des rétroactions qui prennent place entre deux apprenants lors de la réalisation d'une activité de pair programming distribué.

Il faut aussi noter que le contexte particulier dans lequel les rétroactions sont produites, avec des messages de type *rétroactions par les pairs*, augmente significativement la complexité du processus d'annotation. En effet, en complément de la richesse intrinsèque de la langue française, le respect des conventions grammaticales, orthographiques et syntaxiques (notamment l'utilisation de la ponctuation) de la langue n'est pas une priorité des auteurs des messages. Il y a donc parfois de réelles questions sur le sens et/ou l'intention à associer à certains messages, ne serait-ce que sur le fait de savoir s'il s'agit d'une injonction ou d'une interrogation. Ceci explique certainement certaines simplifications des valeurs des attributs que nous avons eu à réaliser, et augmente aussi le défi que représente une classification (semi)automatique des rétroactions que nous abordons dans la section suivante.

4 Aide à l'annotation des rétroactions

Le nombre de messages à classifier étant important (nous avons, à ce jour, environ 1700 messages résultant des différentes expériences menées), et la tâche d'annotation étant fastidieuse, l'équipe a initié une approche fondée sur une solution d'étiquetage semi-automatique.

L'attribut *source* est directement mentionné dans les traces collectées par l'application, alors que le *nombre de mots* peut être calculé par un simple programme en fonction du message passé en paramètre.

Pour les attributs *contenu*, *forme* et *nature*, qui sont bien plus complexes à inférer de manière automatique, notre approche repose sur l'utilisation d'un grand modèle de langue (*Large Language Model* - LLM). En effet, les travaux de [7] ont montré que les LLM entraînés par une approche Few-shot peuvent produire de bons résultats dans les tâches d'étiquetage, tout en décourageant une confiance totale dans les résultats qui sont produits. Nous avons donc décidé de tirer parti d'une pré-classification, réduisant la tâche fastidieuse d'annotation en une tâche de relecture et de correction, moins lourde. Nous avons jusqu'à présent conçu un script Python pour préétiqueter les messages à l'aide de l'API Mistral. Nous avons rédigé un prompt *few-shot* complet, fournissant non seulement les définitions de nos différents attributs mais proposant également plusieurs exemples pour chacune des valeurs associées aux différents attributs, et avons aussi testé et défini empiriquement les hyperparamètres. Des travaux en cours de réalisation qui impliquent deux annotateurs indépendants consistent à annoter les messages qui n'ont pas été étiquetés par le LLM, ainsi qu'à modifier les messages mal étiquetés par le LLM. Une fois ce travail réalisé, nous calculerons le Kappa de Cohen pour évaluer la concordance entre les deux évaluateurs. Nous sélectionnerons finalement la classification d'un des deux annotateurs afin de poursuivre ce travail à travers l'analyse des effets des différentes rétroactions sur le code produit par les apprenants lors d'une activité de pair programming distribué.

5 Conclusions et perspectives

Dans cette contribution nous proposons tout d'abord une analyse de nombreux travaux proposant des caractérisations des rétroactions. Cette analyse de la littérature nous permet de définir huit grandes catégories de rétroactions sur lesquelles nous nous appuyons pour proposer, de manière itérative, une grille d'annotation adaptée à des messages produits par des apprenants dans le cadre d'activités de pair programming distribué. La validation de cette grille, et du guide d'annotation associé, a été réalisée par un travail d'annotation à l'aveugle par quatre juges indépendants. Les différents degrés d'accord calculés ont abouti à des accords qualifiés de *substantiels* à *presque parfaits*. Aussi, pour réduire la charge humaine associée au travail d'annotation et pour pouvoir traiter un grand nombre de messages, nous avons proposé une approche d'étiquetage semi-automatique à l'aide d'un LLM. Cette approche est opérationnelle et l'étiquetage de quelques 1700 messages est actuellement en cours de finalisation. L'aboutissement de ce travail nous permettra de bénéficier d'une grande base de messages annotés, et ainsi de poursuivre d'autres objectifs de recherche.

En effet, les prolongements du travail présenté s'inscrivent dans deux directions. D'une part, en lien avec certaines évolutions envisagées dans notre plateforme de pair programming distribué, un objectif concerne la prise en compte en temps réel des messages échangés pour réguler et guider les apprenants pendant

la réalisation d'activités. Pour atteindre cet objectif, l'approche que nous privilégions consiste à s'appuyer sur des algorithmes tels que TF-IDF ou BM25 pour mesurer le degré de similarité entre un message émis par un apprenant et une large base de messages déjà annotés comme celle actuellement en cours de finalisation. Toutefois, les libertés linguistiques que prennent souvent les apprenants lors de la rédaction de messages textuels renforce la difficulté de l'exercice. L'autre direction de nos futurs travaux est en lien avec l'annotation de rétroactions, et concerne la généralisation de notre classification. Nous souhaitons en effet évaluer la pertinence et l'utilité de notre grille avec d'autres rétroactions en lien avec l'apprentissage de la programmation, mais produites dans d'autres contextes que celui du pair programming et avec des approches différentes (voir par exemple [10] et [1]).

Références

1. Branthôme, M., Lallé, S.: Impact of Adaptive Feedback on Learning Programming with a Serious Game in High Schools' Classes. In: Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization. pp. 104–113. ACM, New York City USA (Jun 2025). https://doi.org/10.1145/3699682.3728326
2. Cohen, J.: Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. Psychological bulletin **70**(4), 213 (1968)
3. Colin, J., Hoarau, S., Declercq, C., Broisin, J.: Design and evaluation of a web-based distributed pair programming tool for novice programmers. In: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1. p. 527–533. ITiCSE 2024, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3649217.3653571
4. Deeva, G., Bogdanova, D., Serral, E., Snoeck, M., De Weerdt, J.: A review of automated feedback systems for learners: Classification framework, challenges and opportunities. Computers & Education **162**, 104094 (Mar 2021). https://doi.org/10.1016/j.compedu.2020.104094
5. Fleiss, J.L.: Measuring nominal scale agreement among many raters. Psychological bulletin **76**(5), 378 (1971)
6. Gilman, D.A.: Comparison of several feedback methods for correcting errors by computer-assisted instruction. Journal of Educational Psychology **60**(6, Pt.1), 503–508 (Dec 1969). https://doi.org/10.1037/h0028501
7. Gunes, E., Florczak, C.K.: Replacing or enhancing the human coder? multiclass classification of policy documents with large language models. Journal of Computational Social Science **8**(2), 31 (Feb 2025). https://doi.org/10.1007/s42001-025-00362-2, https://doi.org/10.1007/s42001-025-00362-2
8. Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., Zander, C.: Pair programming in education: A literature review. Computer Science Education **21**(2), 135–173 (2011)
9. Hattie, J., Timperley, H.: The Power of Feedback. Review of Educational Research **77**(1), 81–112 (Mar 2007). https://doi.org/10.3102/003465430298487
10. Jolivet, S., Kirouchenassamy, B., Yessad, A., Luengo, V.: Modélisation et décision de rétroactions épistémiques dans l'environnement AlgoPython. In: Lavoué, E., Romero, M., Peter, Y. (eds.) Acte de la 12e conférence sur les Environnements Informatiques pour l'Apprentissage Humain, EIAH25. pp. 172–190. Lille (2025)

11. Jolivet, S., Yessad, A., Muratet, M., Lesnes, E., Grugeon-Allys, B., Luengo, V.: Rétroactions dans un environnement numérique d'apprentissage : modèle de description et décision. STICEF **29**(2), 87–123 (2022). https://doi.org/10.23709/STICEF.29.2.4
12. Keuning, H., Jeuring, J., Heeren, B.: A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. ACM Transactions on Computing Education **19**(1), 1–43 (2018). https://doi.org/10.1145/3231711
13. Van der Kleij, F.M., Feskens, R.C.W., Eggen, T.J.H.M.: Effects of Feedback in a Computer-Based Learning Environment on Students' Learning Outcomes: A Meta-Analysis. Review of Educational Research **85**(4), 475–511 (Dec 2015). https://doi.org/10.3102/0034654314564881
14. Kulhavy, R.W., Stock, W.A.: Feedback in written instruction: The place of response certitude. Educational Psychology Review **1**(4), 279–308 (Dec 1989). https://doi.org/10.1007/BF01320096
15. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. biometrics pp. 159–174 (1977)
16. Leibold, N., Schwarz, L.M.: The art of giving online feedback. Journal of Effective Teaching **15**(1), 34–46 (2015), publisher: ERIC
17. Liu, N.F., Carless, D.: Peer feedback: the learning element of peer assessment. Teaching in Higher Education **11**(3), 279–290 (Jul 2006). https://doi.org/10.1080/13562510600680582
18. Mason, B.J., Bruning, R.: Providing feedback in computer-based instruction: What the research tells us. CLASS Research Report 9, Center for Instructional Innovation, University of Nebraska-Lincoln (2001)
19. Mertens, U., Finn, B., Lindner, M.A.: Effects of computer-based feedback on lower- and higher-order learning outcomes: A network meta-analysis. Journal of Educational Psychology **114**(8), 1743–1772 (Nov 2022). https://doi.org/10.1037/edu0000764
20. Murray, J., Gasson, N.R., Smith, J.K.: Toward a taxonomy of written feedback messages. In: Lipnevich, A., Smith, J.K. (eds.) The Cambridge handbook of instructional feedback. pp. 79–96. Cambridge University Press (2018)
21. Narciss, S.: Feedback strategies for interactive learning tasks. In: Handbook of research on educational communications and technology, pp. 125–143. Routledge (2008)
22. Narciss, S.: Designing and Evaluating Tutoring Feedback Strategies for digital learning environments on the basis of the Interactive Tutoring Feedback Model. Digital Education Review **23** (2013)
23. Shute, V.J.: Focus on Formative Feedback. Review of Educational Research **78**(1), 153–189 (Mar 2008). https://doi.org/10.3102/0034654307313795
24. Small, M., Lin, A.: Instructional feedback in mathematics. In: Lipnevich, A., Smith, J. (eds.) The Cambridge handbook of instructional feedback, pp. 169–190. Cambridge University Press (2018)
25. VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., Wintersgill, M.: The andes physics tutoring system: Lessons learned. International Journal of Artificial Intelligence in Education **15**(3), 147–204 (2005)