Univsertité de Toulouse
Licence 3 MIDL

Rapport de Stage

Analyse des messages textuels pour l'apprentissage de la programmation : le cas du pair programming distribué

BOIREAU-DEVIER Chloé
22204638
chloe.boireau-devier@univ-tlse3.fr

Laboratoire d'accueil : Institut de Recherche en Informatique de Toulouse (IRIT)
Responsable : BROISIN Julien

Janvier-Juin 2025

Table des matières

1 Introduction

Le pair programming est une technique de développement dans laquelle deux programmeurs travaillent ensemble sur le même poste. Les deux participants occupent deux rôles différents. L'un, nommé "Driver" écrit le code tandis que l'autre, le "Navigator", examine et vérifie le code produit. Les deux acteurs peuvent changer de rôle au cours d'une même session, et peuvent éxécuter independamment le code produit par le Driver, possédant chacun une console.

Dans notre cas, nous étudions précisemment le pair programming ditribué (distributed pair programming ou DPP), qui consiste en la même chose, à ceci près que les deux programmeurs travaillent ensemble à distance, en communiquant par exemple en utilisant une visioconférence ou une messagerie en ligne.

La tâche qui m'a été confiée lors de ce stage est de classifier relativement à la littérature les messages échangés sur une plateforme de pair programming distribué, en les considérant comme du feedback sur le travail produit, et d'étudier l'impact des différents types de feedback sur la qualité du code produit afin d'assister le navigator dans le choix de ses messages pour avoir l'impact le plus positif possible sur le code.

Ce travail s'inscrit dans un contexte plus large, la thèse de José Colin, doctorant à l'Institut de Recherche en Informatique de Toulouse (IRIT), sous la direction de Julien Broisin, maître de conférences à l'IRIT, et en collaboration avec Sébastien Jolivet, docteur en didactique des mathématiques.

L'objectif du travail de José est d'apporter de nouvelles connaissances dans le domaine de l'ingénierie d'Environnements Informatiques pour l'Apprentissage Humain (EIAH) dédiés au distributed pair programming pour des programmeurs novices, dans des situations d'apprentissage à distance ou hybrides. Il doit notamment concevoir, implanter et évaluer "un nouveau système de distributed pair programming pour les débutants, favorisant le gain d'apprentissage et la collaboration entre apprenants, et permettant à l'enseignant d'entreprendre des actions de remédiations individuelles et/ou collectives", d'après son sujet de thèse.

L'application sur laquelle ont été conduites les expériences permettant la récupération de messages à annoter est celle qu'il développe dans le cadre de son travail.

José fait donc partie de l'équipe TALENT (Teaching And Learning Enhanced by Technologies) de l'IRIT, qui travaille principalement sur trois axes scientifiques dans le domaine des Environnements Informatiques pour l'Apprentissage Humain : EIAH pour soutenir l'apprentissage actif, analyse de l'apprentissage pour un apprentissage personnalisé et autogéré, et compétences numériques et développement pour l'avenir. J'ai eu le plaisir de temporairement rejoindre cette équipe dans le cadre de ce stage.

L'aboutissement de mon travail permettrait, grâce aux résultats obtenus, d'implanter directement dans l'application de José un moyen de guider le Navigator sur le choix de ses messages afin de soutenir au mieux la tâche de programmation.

2 État de l'art

La classification des messages se basant sur la littérature, la première partie de mon travail a consisté à réaliser un état de l'art des classifications déjà existantes, en se concentrant sur la didactique de l'informatique. En partant des travaux et documents proposés dans ma proposition de stage (annexe A), j'ai cherché de nouveaux articles à consulter en me basant sur les travaux cités par les articles que je consultais.

Étant donné la précision du sujet sur lequel j'ai travaillé, il y avait peu de littérature qui correspondait exactement au cas du feedback dans le cas du pair programming distribué, à part le travail de Mr. Jolivet [1]. En parcourant les références, je me suis donc concentrée sur des articles traitant du feedback généré par ordinateur [3], [6], [9], [10], [12], de systèmes de tuteurs intelligents [7], [8], de feedback pendant une tâche d'apprentissage sur ordinateur [2], [11] et du feedback entre élève et professeur [4], [5], [13], [14], [15], [16].

Dans la suite de mon travail, je me suis demandé si je ne m'étais parfois pas trop éloignée de mon sujet à cause de la structure arborescente des citations, et si j'avais aujourd'hui à réitérer l'exercice, je m'assurerais de l'intérêt de l'article dans le cadre de mes recherches avant de l'inclure, plutôt que de "m'enfoncer" dans un sujet sans me rendre compte que cela n'est pas le plus pertinent vis à vis de mon travail. Par exemple, les articles sur le feedback entre élève et professeur ne s'inscrit pas réellement dans le cadre du pair programming distribué ; bien que les résultats obtenus nous serviront plus tard.

Cet état de l'art a tout d'abord pris la forme d'un tableau (figure 1, annexe B) qui, pour chaque article étudié, donne le contexte, le type de message (texte, video, ...) et l'intéraction (élève-professeur, élève-machine) étudiée, ainsi que la classification proposée ou utilisée et la définition de chaque catégorie selon le ou les auteur(s).

Article : Jolivet, S., Yessad, A., Muratet, M., Lesnes-Cuisiniez, E., Grugeon-Allys, B., & Luengo, V. Rétroactions dans un environnement numérique d'apprentissage: modèle de description et décision. Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation, 29(2), 87-123

Contexte : Présentation d'un modèle de description des rétroactions épistémiques et d'un modèle informatique de décision de ces rétroactions.

Type de message : Message textuel, produit par un système informatique. L'apprenant ne peut pas répondre directement à une rétroaction, mais son travail impacte la rétroaction qu'il reçoit.

Composante	**Définition**
Contenu Rétroaction	Une ou plusieurs procédures de résolution et/ou une ou plusieurs justifications ex : « 2y+1 = 5y – 5 2y+1-2y = 5y – 5 -2y 1+5 = 3y – 5 + 5 6 = 3y y = 2 »
Annotation	Éléments facultatifs enrichissant la rétroaction : détaille un aspect du contenu, intègre des éléments relatifs à une erreur, expliciter une fonction de la rétroaction Peut concerner : niveau global de la rétroaction, élément du contenu Cibles : niveau global, procédure, justification ex : « Nous te proposons un exemple de résolution d'une équation dont tu peux t'inspirer »
Erreur	Décrit un contenu facultatif relatif à une erreur, par une annotation Peut présenter une erreur faite ou un élément souvent source d'erreur ex :« Attention, avant d'utiliser le théorème de Pythagore, il faut bien vérifier que ton triangle est rectangle »
Fonction de la rétroaction (fonction explicitée ou fonction(s) potentielle(s)) 1. Fonction de guidage 2. Fonction d'aide à la reprise de tâche 3. Fonction de contrôle	1. Aide pour débuter l'activité (rappels de cours, procédure) ex : « Pour commencer n'hésite pas à réaliser un schéma à partir des données de l'énoncé » 2. Justification(s), procédure(s) et éventuellement éléments relatifs à une erreur ; utilisé quand l'apprenant a réalisé au moins partiellement la tâche, avec une ou plusieurs erreurs. ex : « Voici un exemple détaillé qui devrait t'aider à corriger ta production [...]» 3. Destiné aux apprenants ayant fini la tâche avec au moins une erreur, moins fort que (2). But : emmener l'apprenant à contrôler de manière autonome son résultat ex : « Nous te rappelons que [...]. Ta réponse est-elle bien une solution de l'équation ? »
Données	Type de données utilisées pour la formulation de la rétroaction et son/leur rapport

FIGURE 1 – Exemple d'une synthèse de classification issue de la littérature [1]

A la vue du nombre de catégories trouvées, et étant donné que certaines semblaient faire écho à d'autres, nous avons comparé les différentes approches des auteurs dans un tableur (annexe C), pour voir quelles étaient les catégories qui nous seraient utiles dans notre construction de classification. Comme certaines catégories étaient des sous catégories d'attributs déjà présents, nous avons résumé les recherches en utilisant une carte mentale (figure 2, annexe D), afin de voir les catégories mères, qui seraient plus générales et donc, qui seraient celles qui nous intéresseraient le plus.

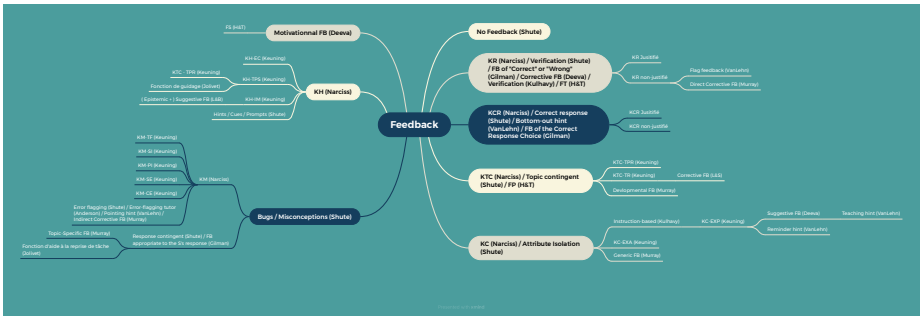

FIGURE 2 – Arborescence des Classifications (D)

Les attributs se divisaient donc en 8 grandes catégories :

— l'absence de retour ou feedback ;

— le feedback "motivationnel", qui vise à encourager l'apprenant ;
— les feedbacks concernant le savoir sur comment procéder ;
— les feedbacks à propos des erreurs ou des fausses croyances de l'apprennant ;
— les feedbacks donnant de l'information à propos de la justesse ou non du travail rendu ;
— les feedback donnant la réponse attendue ;
— les feedbacks concernant la tâche à effectuer ;
— enfin, les feedbacks concernant les aspects techniques ou conceptuels en lien avec la tâche à effectuer.

Étant donné la taille et la nature des messages étudiés (très courts, parfois ségmentés et délivrés instantanément par quelqu'un du même statut que l'apprennant, et non pas un professeur), ainsi que la volonté d'automatiser le processus d'étiquetage, il n'est pas possible d'utiliser directement l'une des classifications trouvées lors de l'état de l'art, ni même le regroupement fait précedemment. Nous avons donc sélectionné des attributs et des valeurs justifiées par la littérature, mais plus en lien avec notre sujet. La recherche a abouti aux attributs et aux valeurs suivantes :

- Attribut "Forme grammaticale du Feedback" : décrit la forme grammaticale du message. Les valeurs possibles sont :
  - "Phrase affirmative" ;
  - "Phrase négative" ;
  - "Phrase interrogative" ;
  - "Phrase impérative".
- Attribut "Source" : permet de définir la source du message, s'il a été envoyé par un acteur ayant le rôle de Driver ou le rôle de Navigator ;
- Attribut "Taille du message" : le nombre de caractères contenus dans le message ;
- Attribut "Nombre de mots" : le nombre de mots contenus dans le message ;
- Attribut "Ton du message" : le ton employé dans le message. Les valeurs possibles sont :
  - "Négatif" ;
  - "Positif".
- Attribut "Groupe du message" : indique si le message est "seul" ou lié à une conversation, ou complète un message déjà envoyé. Les valeurs possibles sont :
  - "Message seul" ;
  - "Message d'une conversation Driver-Navigator" ;
  - "Message répondant à lui-même" : suite d'un message ne prennant pas en compte une possible réponse de l'autre acteur.
- Attribut "Forme du contenu" : indique la forme de contenu du Feedback. Les valeurs possibles sont :
  - "Instruction" : indique une directive pour réaliser une tâche ;
  - "Explication" : fournit une justification ;

  - "Validation" : "Je suis d'accord", "Je ne pense pas que ça marche".
- Attribut "Contenu du Feedback" : indique le type d'élément important pour le pair programming contenu dans le FB. Les valeurs possibles sont :
  - "Sans élément relatif à la programmation, autre" ;
  - "Sans élément relatif à la programmation, élément relatif au rôle" ;
  - "Relatif à la programmation et avec lien avec la tâche" ;
  - "Relatif à la programmation et sans lien avec la tâche".
- Attribut "Nature" : indique la "catégorie" de FB à laquelle appartient le message. Les valeurs possibles sont :
  - "Validité de la réponse" : indique si la réponse ou la solution proposée par l'un des acteurs semble juste ou non à l'autre acteur ;
  - "Réponse correcte" : donne la solution directement (un bloc de code qu'il faut copier-coller dans l'exercice). Le code n'est peut-être pas juste au sens de l'exercice, et peut contenir des erreurs ; mais on s'intéresse ici plutôt au fait d'envoyer un bloc de code, sans forcément échanger sur la stratégie à utiliser, sans passer la main, sans intention de corriger une erreur ;
  - "Connaissances sur la tâche" : se rapporte à des explications données par rapport au contexte de la tâche, par exemple, dans le cas de l'exercice sur les chiffres romains, les messages de cette catégorie seraient plutôt axés sur les explications ou exemples de calcul ;
  - "Connaissances sur les concepts" : se rapporte à des explications données sur des concepts de programmation : on pourrait retrouver ce genre de message dans deux exercices aux sujets différents. Un exemple serait une explication sur la structure de dictionnaire : ce que c'est, comment l'utiliser etc... ;
  - "Informations sur comment procéder" : se rapporte à toute démarche de recherche de solution ou instruction sur comment se rapprocher de la solution. Là où les catégories « connaissances » concernent plus des explications pour comprendre la tâche à effectuer et les outils à utiliser ; les messages de cette catégorie « avancent » vers une solution au problème. Bien qu'un morceau de code puisse correspondre à cette catégorie, en général, si le code peut aussi aller dans la catégorie « réponse correcte », on préférera affecter le message à cette dernière ;
  - "Erreurs, idées fausses" : correspond à tout message corrigeant un des acteurs après qu'une faute soit commise, ou en prévention d'une erreur souvent commise. Cette catégorie peut ressembler aux catégories « Réponse correcte » et « Informations sur comment procéder » : pour la distinguer de la première, il faut regarder le contexte dans lequel est envoyé le message ; pour la distinguer de la deuxième, on préférera affecter le message à la catégorie « Erreurs, idées fausses » si un mot marquant l'intention de corriger est présent ;
  - "Feedback motivationnel" : correspond aux messages visant à encourager les acteurs à réessayer, à ne pas abandonner ;
  - "Demande d'aide" : correspond aux messages pour lesquels un des acteurs demande de l'aide, un retour, une confirmation ou un passage de la main. Il s'agit souvent du premier message d'un échange initié par le Driver ;

– "No feedback" : le message ne contient pas de feedback utile à l'activité de pair programming.

Les trois dernières catégories ont été difficiles à determiner, mais nous avons justifié nos choix en nous appuyant sur les catégories les plus présentes dans la littérature, et, afin de prouver que notre proposition est bien une classification, nous avons mis en place un guide d'annotation (annexe F), et nous avons chacun (José, Julien, Sébastien et moi-même) annoté un corpus de messages sans se concerter pour s'assurer que nous étions d'accord dans l'annotation. Nous avons calculé notre degré d'accord à l'aide du Kappa de Cohen qui mesure la fidélité inter-évaluation en comparant les résultats des accords de classification par rapport au hasard. Nous obtenons des Kappa de Fleiss (qui est une version du Kappa de Cohen pour plus de deux annoteurs) de 0.40 pour la forme, ce qui témoigne d'un certain accord, surtout pour 4 annoteurs. Les accords deux-à-deux qui sont visibles sur la figure 3 sont aussi acceptables. Cependant, pour les kappas croisés du contenu et de la nature, nous n'étions pas satisfaits, et après une réduction des valeurs possibles, entre deux annoteurs, nous avons obtenu un kappa de 0.56 pour la forme, ce qui témoigne d'un accord plus fiable. Il n'y a pas de nouveau score pour la nature car les nouvelles annotations n'ont pas encore été faites par l'équipe.

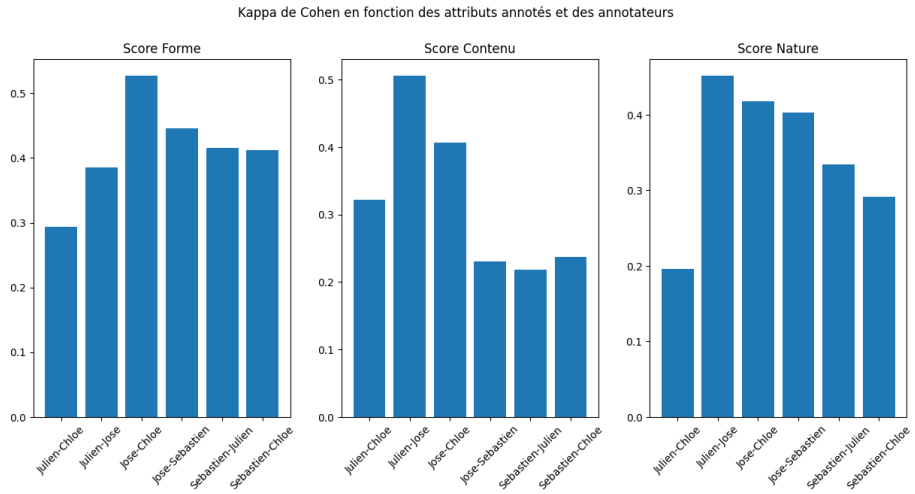

FIGURE 3 – Kappa de Cohen en fonction des attributs annotés et des annotateurs

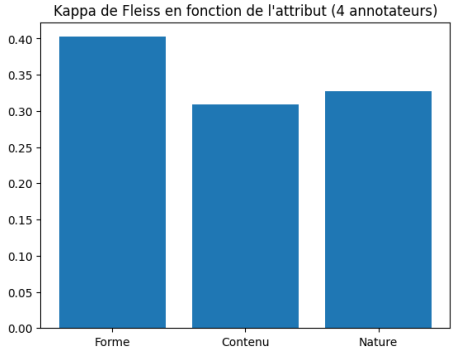

FIGURE 4 – Kappa de Fleiss en fonction de l'attribut (4 annotateurs)

A partir de cette classification, nous avons pu commencer à réfléchir à propos de l'automatisation de l'étiquetage des messages, ce qui sera traité dans la partie suivante.

3 Classification automatique

Le nombre de messages à classifier étant important (nous avons, à ce jour, environ 1700 messages résultant des différentes expériences menées), l'équipe s'est orientée vers une solution d'étiquetage automatique fondée sur l'utilisation d'un LLM.

Pour étiqueter automatiquement les messages, il faut tout d'abord s'assurer du bon format des messages récupérés lors des expériences faites avec l'application de José. Les données étaient stockées au format xApi, un format permettant d'enregistrer et de suivre tous les types d'échanges dans l'application. Il a fallu donc retravailler leur forme pour qu'elle soit plus adaptée au traitement que nous voulons faire.

Les données ont donc été transofrmées au format JSON, un format plus facilement traitable et lisible. Par la suite, en utilisant entre autres la librairie Pandas, deux fichiers au format CSV regroupant les données ont été créés :

— le premier regroupant uniquement les messages du Driver ou du Navigator émis via le tchat ; avec des informations sur la séance et le groupe d'où provennaient les données, ainsi que le rôle (Driver ou Navigator) de l'étudiant envoyant le message au moment de l'envoi ;
— le deuxième regroupant toutes les traces possédées, c'est-à-dire les messages comme dans le premier fichier, mais également le code présent sur la plateforme, les sorties consoles lorsques celles-ci ont été demandées par l'un des acteurs, et les passages de la main sur l'éditeur.

Ce deuxième fichier ne nous concerne pas tout de suite, mais sera très utile lorsque nous nous intéresserons à l'impact produit par les feedbacks, car cet impact ne peut être mesuré uniquement avec les messages envoyés lors de l'exercice.

Comme mentionné très brièvement dans la partie précédente, nous avons séléctionné quelques messages issus du CSV contenant les messages pour en faire

un corpus qui à été utilisé pour tester notre classification et mesurer l'accord inter-juges ; mais ce corpus sera également notre base d'apprentissage pour la classification automatique des messages. Cette classification et tout le code utilisé pendant le stage sont disponibles sur GitHub (annexe E).

Pour les attributs "Source", "Taille du message", et "Nombre de mots", de simples programmes peuvent calculer leurs valeurs en fonction du message passé en paramètre.

Pour les autres attributs, nous allons utiliser la méthode "Few shots", utilisant un grand modèle de language (Large Language Model ou LLM), proposée par Luciano Hidalgo [17], qui à travaillé à l'IRIT pendant un séjour en France. Cette méthode consiste à utiliser un LLM, en l'occurence ici Chat-GPT, en lui donnant des explications sur ce qu'il doit faire, avec des directives précises, et des exemples de données en entrée, ainsi que ce que l'administrateur (ou programmeur) attend comme réponse pour ces données : dans la suite de ce rapport, cette façon de procéder s'appelera le "prompting".

Pour commencer, nous avons donc utilisé la méthode few shots sur des attributs relativements simples, mais non calculables : la forme grammaticale et le ton. Voici un exemple de prompting pour ces attributs :

```
# Identity

You are an expert annotator of utterances. You are
    annotating messages between students using a
    distributed pair programming application with a live
    chat to communicate.
Please **classify** each sentence in the conversation
    utterances according to the given annotation
    guidelines for each category.
You should only use the annotations that are given to you
     for each category you have to annotate.
Every new category is defined with the "##" marker.
The input is in json, and the response should be STRICTLY
     in json format.
The label should be in a field called **annotation** and
    it should be present for every utterance.

# Instructions

## Category **form**

For the first category, **form**, the guidelines are :

- **negativeSentence**
  - **Guidelines:** Use this annotation if the sentence
      is grammaticaly negative. Focus on the form, not the
       meaning.
  - **Examples:** "No, I don't think what you've done is
      the answer." , "I don't know"
```

```
- **imperativeSentence**
  - **Guidelines:** Use this annotation if the sentence
      is giving an order.
  - **Examples:** "Add a parenthesis to line 3", "Write a
      for loop"

- **question**
  - **Guidelines:** Use this annotation if the sentence
      is asking a question, with or without the question
      mark.
  - **Examples:** "Can you help me ?", "Are you okay with
      this solution ?", "What do you think about this"

- **positiveSentence**
  - **Guidelines:** This is the default annotation for
      the "form" category. If none of the above apply,
      then classify the sentence using this annotation.
  - **Examples:** "Yes, I agree.", "Sure", "I think it's
      right", "There's errors"

-> END OF FORM CATEGORY

## Category **tone**

For the second category, **tone**, the guidelines are :

- **negativeTone**
  -**Guidelines:** used **only** if the sentence
      expresses an annoyed tone, or is disrespectful.
  -**Examples:** "Bad", "You suck"

- **positiveTone**
  -**Guidelines:** This is the default annotation for the
      "tone" category. If none of the above apply, then
      classify the sentence using this annotation.
  -**Examples:** "Excellent !", "That's great", "it says
      that there are some errors"

-> END OF TONE CATEGORY
```

Comme expliqué, ce prompting est accompagné d'exemples classifiés, pour expliciter le résultat attendu. Cela prend la forme suivante :

```
```json
{
  "annotations": [
    {
      "message_num": 22,
      "utterances": [
        {
```
```

```
            "text": "Bonjour",
            "form": "positiveSentence",
            "tone" : "positiveTone"
          }
        ]
      }
  ]
}
```

Pour ces annotations, nous avons mesuré la performance du modèle en calculant le Kappa de Cohen entre mes annotations et celles du LLM. Les messages séléctionnés ne font pas partie du corpus de messages mentionné précédemment, car la quantité de données est trop importante par rapport aux capacités des modèles gratuits d'Open AI. J'ai donc à la place annoté une conversation de 25 messages, pour se donner une idée des performances obtenues. Après quelques calibrations, nous avons obtenu un kappa de 0.92 pour la forme, ce qui est un très bon score, qui témoigne d'un très grand accord. Pour le ton, il s'avère que dans les deux classifications, tous les messages ont été classés comme "positiveTone". On peut penser qu'il y a un bon accord, mais je pense que cela est à vérifier avec une base de tests plus large, étant donné que "positiveTone" est la valeur par défaut de l'attribut, il se peut que le modèle annote systématiquement les messages avec cette valeur, peu importe le message en lui-même.

Ensuite viennent les catégories plus complexes, "Forme", "Contenu", et "Nature". Leur difficulté vient du fait que les annotations dépendent en partie de l'interprétation du message, et que, déjà au sein de l'équipe, lors d'annotations sur le corpus de messages, nous ne sommes pas tous parfaitement d'accord. Le prompting se doit donc d'être précis, avec le moins d'ambiguités possibles, et avec beaucoup d'exemples pour être le plus exhaustif possible. À plus long terme, comme dit plus tôt, le corpus de messages annotés pourra servir d'exemple pour le modèle. Pour l'instant, nous nous en servirons seulement en partie comme exemple ; et nous utiliserons le reste du corpus pour mesurer la performance du modèle.

Pour l'instant, seules les catégories "Nature" et "Contenu" sont annotées automatiquement, avec des prompts de la même forme que les précédents :

```
## Category **content**

For the third category, **content**, the guidelines are :

- **relatedToProgramming_relatedToTask**
  -**Guidelines:** used when elements of programming are
      in the message, and these elements are linked to the
       task currently being handled
  -**Examples:** "you should use a for loop and inside
      append your value"

- **relatedToProgramming_notRelatedToTask**
  -**Guidelines:** used when elements of programming are
```

```
    in the message, and these elements are generic and
    not linked to the task currently being handled
  -**Examples:** "a list is a collection of objects", "in
     python, you do not need to type your variables"

- **notRelatedToProgramming_relativeToRoles**
  -**Guidelines:** used when there are no elements of
    programming in the message, but that words relative
    to pair programming roles are mentioned
  -**Examples:** "give me the driver role", "hand me over
    "

- **notRelatedToProgramming_Other**
  -**Guidelines:** This is the default annotation for the
     "content" category. If none of the above apply,
    then classify the sentence using this annotation.
  -**Examples:** "hello", "how are you", "i think we're
    done", "yes", "no", "i don't think so"

-> END OF CONTENT CATEGORY

## Category **nature**

For the fourth category, **nature**, the guidelines are :

-**responseValidity**
  -**Guidelines:** used when the message tells the
    student is what they are doing is the correct
    response or not
  -**Examples:** "i don't think that works", "yes, i
    think that's right"

-**motivationnalFeedback**
  -**Guidelines:** used when the aim of the message is to
     keep the student going, to encourage them
  -**Examples:** "keep going", "don't give up"

-**correctResponse**
  -**Guidelines:** used when the message gives the answer
     to the problem. Can be code, or just the way to
    achieve the goal.
  -**Examples:** "You should write this : print(value)",
    "list = [1,2,3,4] line 45", "do a for loop"

-**taskKnowledge**
  -**Guidelines:** used when the aim of the message is to
     help the student with information relative to the
    current task.
  -**Examples:** "the roman numerals here are going to be
     put in a dictionnary to access them easily"
```

```
-**conceptKnowledge**
  -**Guidelines:** used when the aim of the message is to
      help the student with information about programming
      concepts
  -**Examples:** "A dictionnary is like a list but
     instead of using an integer index you use what's
     called a key"

-**howToProceed**
  -**Guidelines:** used when the message gives hints to
     answer to the problem.
  -**Examples:** "maybe you could try a for loop ?"

-**errors**
  -**Guidelines:** used when the messsage points out
     errors or misconceptions on the student's work or
     train of thoughts
  -**Examples:** "maybe you could try a for loop ?"

-**askingForHelp**
  -**Guidelines:** used when the aim of the message is to
      ask the other student to help with a task, or
     concepts
  -**Examples:** "you forgot to return your value", "a is
      a string, not an int"

- **noFeedback**
  -**Guidelines:** This is the default annotation for the
      "nature" category. If none of the above apply, then
      classify the sentence using this annotation.
  -**Examples:** "hello", "how are you", "i think we're
     done", "yes", "no", "i don't think so"

-> END OF NATURE CATEGORY
```

De la même façon, nous allons évaluer les performaces du modèle avec un Kappa de Cohen sur une conversation de 25 messages. Les scores obtenus sont 0.76 pour le contenu, ce qui est encore une fois un bon score témoignant d'un accord certain ; et de 0.32 pour la nature, ce qui est un score bien plus mauvais. Cela peut s'expliquer premièrement par le fait que cette catégorie est la dernière à avoir été implantée (et donc nécessite encore quelques ajustements, notamment dans le prompting), et qu'il y a au total neuf valeurs possibles, ce qui rend plus difficile un accord.

4 Conclusion

Les résultats observés ne sont pas définitifs car mes travaux ne sont pas terminés. Dans l'absolu, nous aimerions finir d'annoter le corpus de messages sélection-

nés afin d'avoir un accord entre quatre humains et un LLM, pour prouver le bon fonctionnement du modèle, et également avoir une base d'exemples plus conséquente pour le prompting du modèle.

Dans l'idéal, nous aimerions aussi essayer d'utiliser Mistral AI plutôt qu'Open AI, prémièrement pour utiliser un LLM français, mais également voir si le prompting en français ne serait pas plus efficace que le prompting en anglais couplé avec des exemples en français.

L'objectif suivant serait de mesurer l'impact des messages sur la qualité du code produit selon leurs attributs. Plusieurs pistes s'offrent à nous : compter la variation du nombre de lignes de code ou de mots après un message ou bien étudier les impressions des consoles auxquelles nous avons accès, par exemple.

Pour l'instant, la production est donc une (presque complète) classification de messages textuels courts dans le cadre du pair programming distribué. Cette classification, basée sur un état de l'art des classifications déjà existantes dans le cadre de l'apprentissage de la programmation, nous permet de classifier automatiquement, à l'aide d'un LLM et de la méthode "few-shots", les données récupérées d'expériences réalisées par José. Cette classification permet alors de réduire la complexité des données étudiées, en les "réduisant" à des catégories limitées en nombre.

5 Références

[1] Jolivet, S., Yessad, A., Muratet, M., Lesnes-Cuisiniez, E., Grugeon-Allys, B., & Luengo, V, *Rétroactions dans un environnement numérique d'apprentissage : modèle de description et décision*, Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation, 29(2), 87-123.

[2] Susanne Narciss. 2008. *Feedback strategies for interactive learning tasks. Handbook of research on educational communications and technology (2008)*, 125–144.

[3] Keuning, H., Jeuring, J., Heeren, B. (2018) *A Systematic Literature Review of Automated Feedback Generation for Programming Exercises*, ACM Transactions on Computing Education 19, 3 :1–3 :43

[4] Shute, V. J. (2008) *Focus on formative feedback*, Review of educational research, 78(1), 153-189.

[5] Leibold, N. et Schwarz, L. M. (2015) *The art of giving online feedback*, Journal of Effective Teaching, 15(1), 34-46.

[6] Mason, B. J., & Bruning, R. (2001) *Providing feedback in computer-based instruction : What the research tells us*, Center for Instructional Innovation, University of Nebraska–Lincoln : 14. Retrieved June 1, 2006

[7] Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. (1995) *Cognitive tutors : lessons learned*, J. Learning Sci., 4, 167–207.

[8] VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M. (2005) *The Andes physics tutoring system : lessons learned*, Int. J. Artif. Intell. Educ., 15 147–204

[9] Gilman, D. A. (1969) *Comparison of several feedback methods for correcting errors by computer-assisted instruction*, Journal of Educational Psychology, 60(6), 503 – 508.

[10] G. Deeva, D. Bogdanova, E. Serral, M. Snoeck, and J. De Weerdt, *A review of automated feedback systems for learners : Classification framework, challenges and opportunities*, Computers & Education, vol. 162, p. 104 094, Mar. 2021, issn : 0360-1315. doi : 10 . 1016 / j . compedu . 2020.104094. [Online]. Available : https://www.sciencedirect.com/science/article/pii/S036013152030292X

[11] F. M. van der Kleij, T. J. H. M. Eggen, C. F. Timmers, and B. P. Veldkamp, *Effects of feedback in a computer-based assessment for learning*, Computers & Education, vol. 58, no. 1, pp. 263–272, Jan. 2012, issn : 0360-1315. doi : 10 . 1016 / j . compedu . 2011 . 07 . 020. [Online]. Available : https://www.sciencedirect.com/science/article/pii/S0360131511001783

[12] U. Mertens, B. Finn, and M. A. Lindner, *Effects of computer-based feedback on lower- and higher-order learning outcomes : A network meta-analysis.*, Journal of Educational Psychology, vol. 114, no. 8, pp. 1743–1772, Nov. 2022, issn : 1939-2176, 0022-0663. doi : 10.1037/edu0000764. [Online]. Available : https://doi.apa.org/doi/10.1037/edu0000764

[13] Kulhavy, R.W., Stock, W.A, *Feedback in written instruction : The place of response certitude*, Educ Psychol Rev 1, 279–308 (1989). https://doi.org/10.1007/BF01320096

[14] Hattie, J. et Timperley, H. (2007), *The Power of Feedback*, Review of Educational Research, 77(1), 81-112. https://doi.org/10.3102/003465430298487

[15] Murray, J., Gasson, R. et Smith, J. (2018), *Toward a taxonomy of written feedback messages*, A. Lipnevich et J. Smith (dir.), The Cambridge Handbook of Instructional Feedback (p. 79-96). Cambridge University Press.

[16] Small, M. et Lin, A. (2018). *Instructional feedback in mathematics*, A. Lipnevich et J. Smith (dir.), The Cambridge Handbook of Instructional Feedback (p. 79-96). Cambridge University Press.

[17] Hidalgo Sepulveda Luciano, *Few Shot Tagging Example* https://github.com/Lucky-Hidalgo/IRIT_Few-Shot-Tagging-Example

A Proposition de Stage

Proposition de stage

Année universitaire 2024/2025

Laboratoire de recherche : Institut de Recherche en Informatique de Toulouse (IRIT), Technologies pour la Formation et l'Apprentissage (TECFA)

Équipe d'accueil : TALENT, TECFA

Encadrement : Julien Broisin, José Colin, Sébastien Jolivet

Contacts : julien.broisin@irit.fr ; jose.colin@irit.fr ; Sebastien.Jolivet@unige.ch

Titre : Analyse des messages textuels pour l'apprentissage de la programmation : le cas du pair programming distribué

Mots-clés : Apprentissage de la programmation ; pair programming ;

Contexte de recherche

Des travaux ont montré de multiples impacts positifs du pair programming (PP) dans l'enseignement [2, 6]. Le PP est fréquemment étudié dans l'enseignement supérieur, mais des recherches menées sur des lycéens [7] et dans des cours d'introduction à l'informatique au niveau universitaire [8] ont démontré des bienfaits similaires lorsqu'il est utilisé avec des programmeurs novices. D'autre part, l'apprentissage hybride est aujourd'hui de plus en plus répandu dans l'éducation. Dans ce contexte, le distributed pair programming (DPP) permet aux étudiants de s'engager à distance dans des activités de pair programming. Selon Schenk et al. [5], le DPP se déroule dans un environnement virtuel où un éditeur partagé est fourni aux utilisateurs, afin que toutes les modifications du code source effectuées par un participant soient transférées sur l'écran de l'autre participant. L'environnement virtuel garantit également qu'un seul utilisateur peut modifier le code source à tout moment. Comme dans le PP traditionnel, cet utilisateur est appelé le Driver et son partenaire est appelé le Navigator. Le rôle du Driver est de travailler activement sur la tâche de programmation en rédigeant le code, tandis que le Navigator observe et assiste le Driver en suggérant des améliorations et en identifiant les parties du code problématiques.

Dans le cadre d'un travail de doctorat, nous avons développé une application de distributed pair programming [1]. Cette plate-forme web intègre différentes fonctionnalités telles qu'un éditeur de code partagé ; un mécanisme de changement de rôle permettant aux participants de passer du rôle de Driver à celui de Navigator ; un outil de messagerie instantanée (ou *chat*) ; et une console permettant de tester les fonctions définies dans l'éditeur de code partagé. À ce jour, deux expérimentations s'appuyant sur cette application ont été mises en œuvre dans l'enseignement supérieur à l'Université de La Réunion. Ces expérimentations ont chacune impliqué environ 150 étudiant·e·s inscrit·e·s en 1ère année de Licence. Les étudiant·e·s étaient distribué·e·s spatialement dans l'amphithéâtre de manière aléatoire (i.e., ils·elles devaient utiliser l'outil de messagerie instantanée pour communiquer) et devaient réaliser une ou plusieurs activités de pair programming. Pour chacune des activités, l'ensemble des messages échangés dans l'outil de messagerie instantanée, ainsi que les évolutions du code source produit par chaque paire d'étudiants, ont été enregistrés.

L'objectif général du stage est d'évaluer l'impact des messages textuels du Navigator sur le code produit par le Driver. En d'autres termes, nous souhaitons évaluer l'effet du feedback fourni par

le Navigator au Driver sur les actions réalisées par ce dernier. Il s'agira donc, à partir des données collectées lors des deux expérimentations mentionnées plus haut, d'identifier quels messages ont un impact sur la production du code, et de caractériser cet impact, lors d'une activité de pair programming réalisée à distance.

Travail à réaliser

Le travail à réaliser par le ou la candidate comporte 4 phases principales :

- Un travail bibliographique portant sur les types de feedback fournis aux apprenants [3, 4] dans le cadre de l'apprentissage de la programmation ;
- Mettre en forme les données collectées lors des 2 expérimentations en vue de leur analyse ;
- À partir des types de feedback identifiés et des données mises en forme dans la phase précédente, proposer une classification (automatique) des messages textuels échangés dans le cadre d'une activité de distributed pair programming ;
- Analyser, selon le(s) type(s) de messages transmis par le Navigator, l'effet de ces messages sur le code produit par le Driver.

Selon l'avancée des travaux, la proposition d'un outil de guidage visant à aider le Navigator à formuler des messages pertinents et utiles aux tâches à réaliser par le Driver pourra être envisagée.

Contexte technologique : Python/R.

Contexte méthodologique : approche itérative.

Contexte juridique : cession des droits patrimoniaux du stagiaire sur l'ensemble de ses productions dans le cadre du stage au laboratoire d'accueil.

Contexte de travail : le stage est effectué sur une durée de 4 ou 5 mois (dates précises à définir avec le ou la candidate), à l'IRIT.

Références

[1] Colin, J., Hoarau, S., Declercq, C., & Broisin, J. (2024). 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 (pp. 527-533).

[2] Faja, S. : Pair programming as a team based learning activity : A review of research. Issues in Information Systems XII, 207–216 (01 2011).

[3] Jolivet, S., Yessad, A., Muratet, M., Lesnes-Cuisiniez, E., Grugeon-Allys, B., & Luengo, V. (2022). Rétroactions dans un environnement numérique d'apprentissage: modèle de description et décision. Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation, 29(2), 87-123.

[4] Keuning, H., Jeuring, J., Heeren, B., 2018. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. ACM Transactions on Computing Education 19, 3:1–3:43.

[5] Papadakis, S. : Is pair programming more effective than solo programming for secondary education novice programmers ? a case study. International Journal of Web-Based Learning and Teaching Technologies 13 (01 2018).

[6] Salleh, N., Mendes, E., Grundy, J. : Empirical studies of pair programming for cs/se teaching in higher education : A systematic literature review. IEEE Transactions on Software Engineering 37(4), 509–525 (2011).

[7] Schenk, J., Prechelt, L., Salinger, S. : Distributed-pair programming can work well and is not just distributed pair-programming. In : Companion Proceedings of the 36th International Conference on Software Engineering. p. 74–83. ICSE Companion 2014 (2014).

[8] Wood, K., Parsons, D., Gasson, J., Haden, P. : It's never too early : pair programming in cs1. In : Proceedings of the Fifteenth Australasian Computing Education Conference - Volume 136. p. 13–21. ACE '13 (2013).

B Synthèse de lecture

https://docs.google.com/document/d/15xzZPIz5Deb-FP0Llt8N33ZethdV5mnq/edit?usp=sharing&ouid=108323778746346018590&rtpof=true&sd=true

C Tableur récapitulatif des classifications

https://docs.google.com/spreadsheets/d/1Fihz6ZokVXiiD_T_F0ymmGVvYkQehtmu/edit?usp=sharing&ouid=108323778746346018590&rtpof=true&sd=true

D Arborescence des Classifications

https://drive.google.com/file/d/1YODnonr2Wx1LTkBHYKWrpgOflGSvgnk1/view?usp=sharing

E Depôt Github

https://github.com/chloeboireaudevier/StageMIDL

F Guide d'annotation

Guide annotation grille attributs

- Forme grammaticale du FB (feedback)

Attribut décrivant si le FB est affirmatif (ex : « oui, je suis d'accord »), négatif (« non, je ne pense pas que ce que tu as fait est juste »), ou interrogatif (« peux-tu m'aider ? »). Par défaut, une phrase est affirmative si elle ne comporte pas d'élément de négation (« ne [...] pas », etc.), ou d'interrogation. Ne dépend pas du contexte de l'exercice.

- Source du message

Attribut indiquant la source du message : Driver ou Navigator. Ne dépend pas du contexte de l'exercice.

- Taille du message

Valeur numérique indiquant le nombre de caractères du message (l'apostrophe, la virgule et autres signes de ponctuation comptent pour un caractère chacun). Ne dépend pas du contexte de l'exercice.

- Nombre de mots

Valeur numérique indiquant le nombre de mots du FB. Un mot est compté par une suite de lettres. On change de mot si la suite de lettre est interrompue par n'importe quel signe de ponctuation, un tiret, une apostrophe, etc... Ne dépend pas du contexte de l'exercice.

- Forme du contenu

Indique le contenu de FB en choisissant une catégorie selon les suivantes :

– instruction (directive pour réaliser une tâche),

– question (pose une question ou demande une clarification),

– correction (corrige ou identifie une erreur),

– explication (fournit une justification),

– encouragement (encourage un des acteurs a réessayer ou ne pas abandonner),

– description (décrit quelque chose, comme la sortie de console par exemple),

– validation (« je suis d'accord », « je ne pense pas que ça marche »)

– autre (tout ce qui ne rentre pas dans une des catégories précédentes).

Peut dépendre du contexte de l'exercice, pour certaines catégories.

- Groupe du message

Indique si le message est « seul » ou lié à une conversation, ou complète un message déjà envoyé. Valeurs possibles : message seul (indique un message envoyé seul, ou débutant une conversation), message d'une conversation Driver-Navigator (message en réponse à un autre message de l'autre acteur), message répondant à lui-même (message complétant un autre message du même auteur). Dépend du contexte de l'exercice.

- Ton du Message

Cet attribut étudie le ton du FB. Dans l'article de Murray, une différence est faite entre le FB négatif, qui peut décourager l'élève (« Pas assez de détail. »), et le FB positif, qui aurait un ton plutôt amical

et/ou encourageant (« Excellente utilisation de [...] ! »). Par défaut, on considère qu'un FB est positif (tant qu'il n'y a pas d'indicateur négatif, le FB est positif). Ne dépend pas du contexte de l'exercice.

- Contenu du FB

Attribut identifiant les éléments importants du feedback. Les valeurs possibles sont :

– Concepts clés tâche : indique que, dans le FB, se trouvent des mots liés à la tâche elle même. Dépend du contexte de l'exercice (ex : dans le cas d'un exercice où l'on demande la création d'une fonction « coincide » qui donne les indices pour lesquels deux tableau coïncident, le mot « coïncide » n'a, sans contexte, pas de lien avec le pair programming ; mais dans le cas de cet exercice, il a un intérêt évident ; c'est donc un concept clé tâche, et l'attribut est mis à « Concepts clés tâche »). Cet attribut semble être mobilisé, notamment dans les travaux de Narciss, dans la catégorie relevant des connaissances à propos des contraintes liées à la tâche : on identifie ce genre de message à la catégorie concepts clés tâche ; l'attribut est également mobilisé dans la catégorie Devlopmental FB de Murray, qui mentionne des informations sur la manière de progresser, sur le sujet, ou des alternatives.

– Concepts clés programmation : indique que des concepts clés de programmation sont présents dans le FB (ex : « boucle », « dictionnaire », « liste », « compiler », « tester », « exécuter »,...) Ces mots sont en langage naturel, et pas en python. Ne dépend pas du contexte de l'exercice. Cet attribut est mobilisé dans des travaux comme ceux de Narciss, Shute, et Keuning ; notemment dans la façon de donner des exemples à propos des concepts traités (qui sont ici centrés autour de la programmation), comme l'utilisation, la création et l'instanciation de dictionnaires.

– Mots clés python : indique que des mots clés python sont présents dans le FB (ex : « for », « while », « return »). Ne dépend pas du contexte de l'exercice.

– Demande du passage de la main : indique que le message demande à ce que la main soit passée entre les acteurs : « passe-moi la main », « prends la main toi ». Ne dépend pas du contexte de l'exercice.

– Pas de contenu utile au PP : indique que le message ne contient pas d'information utile dans le cas de l'activité de pair programming (ex : « salut », « bonjour », « t'es où dans l'amphi ? »)

– Autre : pour tout message qui ne correspondrait pas à une des catégories ci-dessus.

- Nature du FB

Attribut indiquant la « catégorie » de FB à laquelle appartient le message. Les différentes valeurs possibles sont :

– Validité de la réponse : indique si la réponse ou la solution proposée par l'un des acteurs semble juste ou non à l'autre acteur

– Réponse correcte : donne la solution directement (un bloc de code qu'il faut copier-coller dans l'exercice). Le code n'est peut-être pas juste au sens de l'exercice, et peut contenir des erreurs ; mais on s'intéresse ici plutôt au fait d'envoyer un bloc de code, sans forcément échanger sur la stratégie à utiliser, sans passer la main, sans intention de corriger une erreur.

– Connaissances sur la tâche : se rapporte à des explications données par rapport au contexte de la tâche, par exemple, dans le cas de l'exercice sur les chiffres romains, les messages de cette catégorie seraient plutôt axés sur les explications ou exemples de calcul.

– Connaissances sur les concepts : se rapporte à des explications données sur des concepts de programmation : on pourrait retrouver ce genre de message dans deux exercices au sujet différents. Un exemple serait une explication sur la structure de dictionnaire : ce que c'est, comment l'utiliser etc...

– Informations sur comment procéder : se rapporte à toute démarche de recherche de solution ou instruction sur comment se rapprocher de la solution. Là ou les catégories « connaissances » concernent plus des explications pour comprendre la tâche à effectuer et les outils à utiliser ; les messages de cette catégorie « avancent » vers une solution au problème. Bien qu'un morceau de code puisse correspondre à cette catégorie, en général, si le code peut aussi aller dans la catégorie « réponse correcte », on préférera affecter le message à cette dernière.

– Erreurs, idées fausses : correspond à tout message corrigeant un des acteurs après qu'une faute soit commise, ou en prévention d'une erreur souvent commise. Cette catégorie peut ressembler aux catégories « Réponse correcte » et « Informations sur comment procéder » : pour la distinguer de la première, il faut regarder le contexte dans lequel est envoyé le message ; pour la distinguer de la deuxième, on préférera affecter le message à la catégorie « Erreurs, idées fausses » si un mot marquant l'intention de corriger est présent.

– No feedback : le message ne contient pas de feedback utile à l'activité de pair programming.

– FB motivationnel : correspond aux messages visant à encourager les acteurs à réessayer, à ne pas abandonner.

– Demande d'aide : correspond aux messages pour lesquels un des acteurs demande de l'aide, un retour, une confirmation ou un passage de la main. Souvent le premier message d'un échange initié par le Driver.

G Dossier complet

Lien vers un dossier contenant toutes les productions du stage. Dernière mise à jour : 09/06/25

https://drive.google.com/drive/folders/11qbwNHIhSN4h1T6rAJVwPHjcO7cREX9K?usp=sharing