Actes de la 29e Conférence sur le Traitement Automatique des Langues Naturelles
Avignon, France, 27 juin au 1er juillet 2022

Participation de l'équipe TGV à DEFT 2022 : Prédiction automatique de notes d'étudiants à des questionnaires en fonction du type de question

Vanessa Gaudray Bouju[1*] Margot Guettier[1*] Gwennola Lerus[1*] Gaël Guibon[1, 2*]
Matthieu Labeau[2] Luce Lefeuvre[1]
(1) DTIPG SNCF, 1-3 avenue François Mitterrand, 93210 Saint-Denis, France
(2) LTCI, Télécom Paris, Institut Polytechnique de Paris, 19 place Marguerite Perey, 91120 Palaiseau, France
firstname.lastname@sncf.fr, firstname.lastname@telecom-paris.fr

RÉSUMÉ

Cet article présente l'approche de l'équipe TGV lors de sa participation à la tâche de base de DEFT 2022, dont l'objectif était de prédire automatiquement les notes obtenues par des étudiants sur la base de leurs réponses à des questionnaires. Notre stratégie s'est focalisée sur la mise au point d'une méthode de classification des questions en fonction du type de réponse qu'elles attendent, de manière à pouvoir mener une approche différenciée pour chaque type. Nos trois runs ont consisté en une approche non différenciée, servant de référence, et deux approches différenciées, la première se basant sur la constitution d'un jeu de caractéristiques et la seconde sur le calcul de TF-IDF et de la fonction de hashage. Notre objectif premier était ainsi de vérifier si des approches dédiées à chaque type de questions sont préférables à une approche globale.

ABSTRACT

Team TGV at DEFT 2022 : automatic prediction of students' grades according to the different question types.

In this paper we present the work of the TGV team for the DEFT 2022 challenge. We tackled the base task only, which consists of automatically grading students based on their answers to several questions. Our strategy consider this task as a classification task with multiple approaches, each being specific to a question type leading to different types of expected answers. DEFT 2022 allowed us to send three runs for the final test. From our 3 runs, two runs with approaches specialized for each question type : the first one using extracted features and the second one using vectors from TF-IDF or Hashing Vectorizations. The other one is a run using all the questions as one global set and is used to enable comparison with the first two ones. By doing so, our main objective is to verify if approaches dedicated to question types are more suitable for this task than a global one.

MOTS-CLÉS : Classification, évaluation automatique, approche différenciée, arbre de décision.

KEYWORDS: Classification, automatic grading, differentiated approach, decision tree.

∗. Contribution égale

Atelier DÉfi Fouille de Textes (DEFT), pages 23–35.

1 Introduction

L'équipe TGV (TAL à Grande Vitesse) est composée de six talistes issus de la SNCF et de Télécom Paris. Nous avons choisi de nous focaliser uniquement sur la tâche de base, qui consiste en la prédiction automatique de notes d'étudiants à des questionnaires en se fondant sur leurs réponses et sur la correction proposée par l'enseignant. L'analyse du corpus a mis en lumière plusieurs catégories de questions, qui entraînent différents types de réponses, comme du code ou de la langue naturelle. Suite à ce constat, nous avons décidé d'aborder la tâche par le biais d'approches différenciant les questions et les types de réponses attendues.

Dans ce travail, nous avons cherché à vérifier s'il est nécessaire et bénéfique de prendre en compte les types de questions pour prédire les notes des étudiants. Afin d'appuyer cette hypothèse, nous avons mis en place une recherche des meilleurs hyperparamètres pour chaque type de question. Bien entendu, le cadre du défi étant limité en temps, le présent travail peut être considéré comme une première étape avant de rechercher la performance pure sur chaque type de question.

Dans la suite, nous présentons dans un premier temps notre état de l'art, principalement réalisé autour de la tâche de l'année dernière (Section 2). Nous exposons ensuite nos observations sur le corpus (Section 3). Cela nous permet d'une part d'introduire les caractéristiques qui nous semblent pertinentes à étudier (Section 4), et d'autre part les approches de classification choisies (Section 5). Nous mettons finalement en avant les pistes non retenues pour la tâche mais qui s'avèrent intéressantes (Section 6), avant de conclure (Section 8).

2 État de l'art

La tâche de base de cette année avait déjà été proposée lors du DEFT 2021 (Grouin *et al.*, 2021). Pour notre état de l'art, nous nous sommes donc essentiellement basés sur les travaux menés l'année passée dans le but d'identifier les pistes à approfondir ainsi que pour nous assurer d'apporter de nouvelles méthodes non explorées.

La tâche avait en grande partie été traitée par les participants comme une tâche de classification, bien qu'elle aurait très bien pu être abordée par une tâche de régression. Cela s'explique par une sous-représentation des classes dans le jeu de validation : seules 4 notes différentes y sont présentes. La plupart des équipes avait opté pour des méthodes centrées sur le calcul de scores de similarités. Les différents tests avaient montré que les meilleurs résultats étaient généralement obtenus en calculant la similarité entre la réponse proposée par le professeur et la réponse de l'élève, sans prise en compte de la question. Des plongements lexicaux avaient également été utilisés pour calculer la similarité à partir de vecteurs, bien que, sans *fine-tuning*, les modèles CamemBERT (Martin *et al.*, 2020) et SentenceBERT (Reimers & Gurevych, 2019; Wang *et al.*, 2021) s'étaient avérés moins performants. L'utilisation de *Sentence-Transformers* multilingues (Reimers & Gurevych, 2020; Wang *et al.*, 2021) s'était à ce propos révélée plus efficace. Enfin, à partir de ces scores et de caractéristiques (*features*) linguistiques identifiées, les participants avaient pu entraîner différents classifieurs. Le run ayant remporté les meilleurs résultats (soit $0,682$) utilisait des forêts aléatoires d'arbres de décision, un Random Forest (Breiman, 2001; Suignard *et al.*, 2021).

L'étude des méthodes de 2021 nous a par ailleurs permis de remarquer certaines pistes qui n'avaient pas été abordées. En effet, les questions avaient peu été prises en compte par les participants, de même

que les informations sur les étudiants. Tout en gardant à l'esprit les méthodes les plus performantes, nous avons choisi de nous concentrer sur ces paramètres, non exploités jusque-là, afin de mesurer leur apport.

3 Le Corpus

3.1 Caractéristiques du corpus

Le corpus est issu d'un questionnaire en informatique, adressé à des étudiants de cette filière en vue d'évaluer leurs connaissances. Le corpus comprend deux types de données :

— les données des enseignants : les questions, leur correction et leur barème de notation ;
— les données des étudiants : les réponses et les notes obtenues pour chacune des réponses.

Le corpus d'évaluation comporte 50 questions et 3 820 réponses d'étudiants tandis que celui de validation comprend, pour sa part, 21 questions et 1 644 réponses d'étudiants.

Dans ce questionnaire d'évaluation, les élèves sont interrogés sur des définitions de notions informatiques (« *qu'est-ce que... ?* ») ou sur des cas pratiques, dans lesquels ils doivent notamment donner une portion de code ou un résultat (« *quelle est la balise qui...* ? », « *dans le code, quel est le résultat de...* ? »). Les réponses des étudiants varient donc, en longueur ou en contenu, en fonction de ce qui est attendu. De plus, chaque étudiant a sa propre méthode pour répondre. Certains introduisent par exemple les réponses attendues avec des syntagmes du type « *la réponse est...* », qui peuvent causer du bruit. Par ailleurs, lorsque la réponse doit être du code, des balises et autres caractères spéciaux sont présents. Ce n'est alors pas seulement le vocabulaire de l'élève qui est évalué mais également l'aspect logique de sa réponse.

Enfin, si l'on s'intéresse aux notes à prédire, elles semblent s'étendre de 0 à 1 avec un pas de $0,1$, bien que certaines n'apparaissent pas du tout dans le corpus. La grande majorité des notes, soit 90% du corpus d'entraînement, consiste d'ailleurs en 0, 1 ou $0,5$. Nous remarquons de plus que, dans le fichier de validation, seules ces trois notes-là et la note $0,2$ sont représentées. Les autres variations sont absentes de ce fichier, bien que plus présentes dans le fichier d'entraînement. Enfin, le barème n'est pas systématiquement précisé par le professeur et certaines notes intermédiaires sont parfois même attribuées sans qu'elles aient été envisagées dans la correction fournie.

3.2 Prétraitements réalisés

Afin de rendre nos données exploitables, nous avons effectué certains nettoyages et prétraitements. Le texte des questions et des réponses contenant parfois des balises HTML servant à la mise en page, nous les avons supprimées. Les balises correspondant aux extraits de code présents dans certaines questions et réponses étant pour leur part échappées, nous avons pu les conserver et annuler leur échappement. Nous avons également supprimé les retours à la ligne ainsi que les espaces superflus. À certains endroits, toutefois, il nous a fallu ajouter des espaces entre les balises `<p>` avant de supprimer ces dernières, autrement des éléments appartenant à deux paragraphes différents (correction proposée et barème) se trouvaient juxtaposés et indissociables.

En observant le corpus, nous avons remarqué que certaines réponses n'ont pas été laissées vides

par l'élève sans pour autant constituer une vraie réponse puisqu'elles contiennent uniquement des caractères non alphanumériques comme les points d'interrogations avec répétition typographique « *? ? ?* » ou diverses émoticônes « :'(». Afin d'éviter de fausser les métriques que nous allions calculer, nous avons changé ces réponses en « NO_ANS » de manière à pouvoir leur réserver les mêmes traitements.

Enfin, nous avons souhaité récupérer les réponses proposées par les professeurs indépendamment du barème. En effet, ces deux éléments se trouvent dans la même colonne alors qu'ils n'ont pas la même nature : les propositions de réponses doivent être sémantiquement et/ou formellement proches des réponses des élèves alors que le barème peut donner des indications sans lien direct (par exemple, « *Compté 1 même si faute de frappe ou par ex vald* »). Le traitement de cette deuxième partie étant plus compliqué, nous nous sommes laissé la possibilité de mettre ces éléments de côté. Nous avons donc créé une nouvelle colonne dans le fichier des questions ne contenant que la partie réponse proposée afin de pouvoir plus facilement la prendre en compte seule.

4 Sélection des caractéristiques

4.1 Caractéristiques statistiques

Nous avons identifié différentes caractéristiques à étudier en nous basant notamment sur diverses métriques :

- Des marqueurs permettant d'identifier la tournure de la question (« *qu'affiche* », « *quel code* », « *comment* »…) ;
- Le nombre de mots et le nombre de caractères dans : La question, la réponse du professeur et celle de l'élève ;
- Le nombre de minuscules et de majuscules dans : la question ;
- Le nombre et le pourcentage de caractères de code dans : la question, la réponse du professeur et celle de l'élève ;
- Les entités nommées (ENT) pour le nom de langages informatiques par modèle *camembert-ner* [1] ;
- Les catégories morphosyntaxiques (POS) par modèle *flair/upos-multi* (Akbik *et al.*, 2018).

4.2 Prise en compte des numéros d'étudiants

Les numéros des étudiants étant une donnée fournie, nous faisons l'hypothèse qu'ils pourraient constituer une caractéristique intéressante. Il pourrait en effet exister des « profils étudiants ». Ainsi, selon les notes déjà obtenues par un élève, nous pourrions évaluer son niveau, ce qui conditionnerait ses chances d'obtenir un 1 ou un 0.

Pour vérifier cette hypothèse, nous avons mis au point un modèle probabiliste qui donne, pour chaque élève, une note au hasard à une réponse donnée. Ce modèle se base sur les probabilités pour l'élève d'avoir chacune des notes possibles. Cette caractéristique seule s'est avérée insuffisante mais nous avons pensé que, combinée aux autres, elle pourrait nous aider à mieux choisir les notes en éliminant les plus improbables dans les situations de doute. Nous avons donc jugé intéressant de la conserver.

1. https://huggingface.co/Jean-Baptiste/camembert-ner

5 Classification des questions

Nos observations combinées à la description du corpus faite par les organisateurs (Grouin *et al.*, 2021) nous ont permis d'identifier la présence de trois types de questions en fonction du type de réponse qu'elles attendent (Table 1).

Type de question	Exemple de question	Réponse d'élève
réponse attendue	Qu'affiche le code suivant : <code> <?php function add_presence ($note) $note++ ; ; $note=14 ; add_presence($note) ; echo $note ; ?> </code>	14
réponse en langue formelle	Quel code permet d'afficher la note de Jacques Blanc ?<code> <?php $notes=array("Jean Bleu"=>14, "Jacques Blanc"=>15) ; ?> </code>	echo$notes["Jacques Blanc"]
réponse en langue naturelle	Qu'est-ce que le World Wide Web ?	Ce sont les pages web accessible par tout navigateur

TABLE 1 – Classification des questions en fonction du type de réponse attendue

Nous faisons l'hypothèse que les critères d'évaluation ne sont pas les mêmes en fonction du type de question posée. En effet, nous pouvons supposer que les réponses en langue naturelle laissent plus de liberté dans la tournure et dans le choix des mots employés, tandis que les réponses en langue formelle nécessitent de vérifier non seulement le texte mais également la cohérence formelle de l'enchaînement des caractères de code. Ainsi, une stratégie d'évaluation différente peut être mise en place pour chacun de ces types de réponse attendue. Toutefois, afin de pouvoir utiliser cette classification comme paramètre dans notre méthode, il nous a fallu vérifier que la classification automatique des questions selon les trois types identifiés était bien réalisable. Pour cela, nous avons commencé par annoter les questions manuellement en fonction de leur catégorie pour ensuite mettre en place des méthodes de classification à intégrer à notre chaîne de traitements.

5.1 Arbre de décision appris

En se basant sur les caractéristiques relevées précédemment (section 4.1), nous avons pu entraîner avec *scikit-learn* un arbre de décision pour qu'il parvienne à reconnaître le type de chaque question. L'arbre de décision a pour avantage de nous présenter un rendu visuel des choix effectués par l'algorithme. De cette façon, nous pouvons vérifier qu'il n'y a pas de biais qui se créent. Nous avons ainsi pu réduire les informations transmises pour l'apprentissage lorsque nous avons par exemple remarqué que l'arbre considérait le numéro de la question comme discriminant. Toutefois, la taille assez réduite des données entraîne une inévitable création de biais et les meilleurs taux de précision obtenus $(0, 94)$ ne nous garantissent pas l'adaptation de l'algorithme à d'autres données puisqu'il résulte sans doute d'un surapprentissage, indiqué par une grande variabilité des résultats de l'arbre de décision lors de runs aléatoires.

5.2 Arbre de décision manuel

En parallèle, nous avons élaboré manuellement un arbre de décision (algorithme 1) en partant de l'observation statistique des données pour identifier les critères les plus discriminants. Nous avons finalement retenu deux caractéristiques se basant uniquement sur les réponses des étudiants. Pour ne pas fausser les métriques, nous avons tout d'abord veillé à ne pas comptabiliser les réponses consistant en des « NO_ANS ». Nous avons ensuite utilisé la moyenne des pourcentages de « caractères de code » présents dans les réponses de chaque étudiant à une question donnée, afin d'identifier les réponses formelles lorsque le résultat dépasse un certain seuil. Enfin, pour distinguer les deux autres types de question, nous avons calculé la longueur moyenne des réponses d'étudiants à chaque question restante et nous avons là aussi déterminé un seuil, au-delà duquel la réponse est classée en langue naturelle ou autrement en réponse attendue.

Algorithme 1 Arbre de décision manuel

```
Suppression des NO_ANS
if % moyen des % de caractères code dans les réponses > 3 then
    classif = "langue formelle"
else if longueur moyenne des réponses< 70 caractères then
    classif = "réponse attendue"
else
    classif = "langue naturelle"
end if
```

Cet algorithme nous permet d'obtenir un taux de précision de $0,98$ sur les données d'entraînement et de $0,9$ sur les données de validation. Les tentatives de le complexifier ont plutôt fait baisser ses performances. Aussi, bien qu'il ne soit pas parfait, il nous semblait plus fiable que l'arbre appris et nous avons considéré la marge d'erreur associée comme acceptable. Nous avons donc choisi de le conserver pour la suite des traitements.

6 Stratégies additionnelles

6.1 Tâche de regression

Notre approche initiale fut de considérer la tâche comme une tâche de regression afin de prédire une note allant de 0 à 1. Nous avons donc mis en place une descente de gradient stochastique (SGD) sur des représentations vectorielles obtenues par une fonction de hachage. Ensuite, nous avons fait correspondre la valeur prédite (nombre flottant ente 0 et 1) à l'un des paliers considérés. Cette correspondance a été effectuée de deux manières : par l'arrondi à la note connue la plus proche ou par une distance euclidienne entre les représentations pour trouver la note la plus proche. Le meilleur score de précision obtenu a été de 0,36 en validation (visible en Table 2) en considérant la deuxième approche de correspondance, ainsi qu'un score de RMSE de 0.5153. Considérant le déséquilibre des classes en validation (*i.e.* que 4 notes différentes là où la regression considère toute valeur comme possible), nous avons délaissé cette approche au profit de la tâche de classification.

6.2 Embeddings et similarité

Nous avons songé à ajouter aux caractéristiques précédentes des scores de similarité calculés entre la réponse du professeur et celle de l'élève. En effet, les approches de l'année passée ont montré que la prise en compte de la question faisait baisser les résultats, aussi ne l'avons-nous pas utilisée dans le calcul.

Pour réaliser les embeddings des réponses proposées et des réponses d'étudiants, nous avons utilisé le modèle multilingue *stsb-xlm-r-multilingual*, issu d'un *Sentence-Transformer* (Reimers & Gurevych, 2019). Utilisé par l'équipe Nantalco l'année passée, ce modèle leur avait permis d'atteindre l'un des meilleurs scores (Wang *et al.*, 2021).

Nous avons testé quelques mesures de similarité différentes, certaines donnant des indications sur la proximité globale entre deux phrases (similarité cosinus) et d'autres permettant de s'intéresser aux différences plus subtiles (distance de Levenshtein (Levenshtein *et al.*, 1966)). Nous nous sommes notamment intéressés à celles se plaçant au niveau du caractère, qui nous semblaient plus appropriées pour les réponses attendues et pour celles en langue formelle puisqu'une petite variation peut s'avérer assez pénalisante (une erreur dans une balise de code pourra rendre l'ensemble de la réponse fausse).

Finalement, nous avons uniquement retenu la similarité cosinus, qui s'adaptait mieux à l'ensemble des questions. Nous l'avons calculée de deux façons différentes :

— similarité entre la réponse proposée sans le barème et la réponse de l'élève,
— *idem* mais en supprimant les déterminants ainsi que, dans le cas de la réponse de l'élève, les mots communs avec la question qui ne sont pas dans la réponse proposée (de façon à ne pas prendre en compte les phrases réponses).

Avant d'ajouter cette caractéristique à nos données, nous avons tenté de prédire les notes uniquement à partir de ces scores de similarité pour évaluer leur pertinence. Nous avons pour cela dû définir des seuils manuels qui, utilisés seuls, s'avèrent insuffisant. Toutefois, cette façon de faire nous a permis d'observer une tendance globale qui consiste à sous-évaluer les réponses en langue naturelle et à surévaluer celles en langue formelle. Nous pouvons voir là une indication quant au fait qu'il est effectivement pertinent de distinguer le type de question et que le score de similarité doit être interprété différemment pour chacun d'eux.

C'est cependant par manque de temps que nous n'avons pas pu ajouter les scores de similarité aux modèles utilisés pour les runs. En outre, pour pousser plus loin l'idée d'approche différenciée, il aurait été intéressant de creuser l'effet des mesures se situant au niveau des caractères dans le cas des réponses attendues et formelles.

6.3 Prise en compte du barème

Dans les pistes que nous avons définies jusqu'à présent, nous n'avons pas porté beaucoup d'attention au barème : soit nous prenons en compte l'ensemble de la correction du professeur, soit nous ne nous occupons que de la partie réponse proposée. Le traitement du barème, dans les cas où il est constitué de commentaires sur des notions périphériques (concernant la façon de noter les fautes d'orthographe par exemple), nécessite des traitements assez complexes sur lesquels nous avons choisi de ne pas nous attarder. Néanmoins, dans certains cas, notamment dans celui des réponses attendues, plusieurs réponses toutes faites sont proposées et associées à une note (par exemple, « head 0 si html 0.5 si

meta »).

Pour ces cas spécifiques, nous avons étudié de quelle manière il est possible de prendre en compte chacune des réponses proposées, dont nous supprimons les éléments inutiles (exemple : « si ») et la note associée. Nous avons exploré une méthode de *matching* direct des éléments attendus, en incluant un calcul de la distance de Levenshtein afin de passer outre certaines fautes de frappe. Néanmoins, ces cas où le barème et les réponses proposées sont donnés clairement étant assez minoritaires, le grand nombre d'exceptions rencontrées nous a poussés à laisser cette méthode de côté. En effet, l'utilisation de règles formelles ne s'adapte pas bien à des données aussi diverses.

6.4 Classifications structurée des questions

En reprenant l'idée décrite dans la Section 4.2, nous avons considéré les réponses à chaque question comme autant de séquences d'observations, où chaque séquence contient autant d'éléments qu'il y a d'étudiants qui ont répondu à la question. Ceux-ci ont toujours la même position et les mêmes étudiants répondent aux questions dans les données d'entraînement et celles de validation. Ceci nous a permis d'adopter une approche structurée : nous avons pour cela utilisé le modèle le plus simple à notre disposition, un perceptron structuré, obtenu via *seqlearn* [2]. Bien que le modèle soit relativement performant, nous avons manqué de temps pour l'exploiter, la combinaison avec d'autres modèles n'étant pas évidente.

6.5 Classifications par vote et par agrégation

En faisant l'hypothèse que les performances obtenues en utilisant différentes caractéristiques d'une part, et différents classifieurs de l'autre, sont complémentaires, nous avons cherché à combiner ces résultats via des méthodes *ensemblistes* adaptées à combiner des modèles très différents : le vote (*voting*) et l'agrégation (*stacking*). Nos implémentations sont directement inspirées des classes correspondantes de *scikit-learn* (Pedregosa *et al.*, 2011). Cependant, le vote de différents classifieurs a toujours donné de très mauvais résultats ; et l'agrégation, bien que plus efficace, était toujours décevante en validation.

7 Les runs

Un premier élément commun à tous les runs que nous avons lancés est l'attribution automatique de la note 0 aux réponses correspondant à « NO_ANS ». Nous avons ensuite lancé l'algorithme de détection du type de question (Section 5.2) sur les données pour récupérer cette information. Il nous a ensuite fallu déterminer les trois méthodes que nous souhaitions tester à l'occasion des runs à soumettre. L'ensemble des résultats principaux obtenus est visible en Table 2.

2. http://larsmans.github.io/seqlearn/

Strategie	Run	Val	Test	QFormelle	QAttendue	QNaturelle
all_featsonly	1	0,754	0,491			
qtype_featsonly	2	0,709	0,536	**0,7541**	**0,7513**	**0,6227**
qtype_vect	3	0,687	**0,624**	0,7504	0,7329	0,5792
all_vect		0,644				
structured		0,53				
student_history		0,49				
regression_mapping		0,36				

TABLE 2 – Scores de précision obtenus pour les données de test (test) et de validation (val), ainsi que la précision par type de question (Q) quand l'approche est dissociée.

7.1 Run 1 : caractéristiques toutes ensemble (*all_featsonly*)

Le premier run que nous avons choisi de faire a vocation à servir de référence pour les deux autres. Il consiste en une utilisation des caractéristiques sur tout le dataset. La caractéristique catégorielle du type de question est prise en compte mais est mise au même niveau que les autres caractéristiques que nous avons décrites précédemment. Les meilleurs hyperparamètres trouvés concernent une classification par descente de gradient stochastique (SGD classifier) (Zadrozny & Elkan, 2002), qui a été utilisé pour le run soumis.

Lors du test sur les données d'entraînement, nous avions atteint un taux de précision de 0,754. Toutefois, lors de l'évaluation sur les données de test, la précision a chuté à 0,491. Ce résultat moins bon était escompté et va dans le sens de notre théorie, qui est qu'un entraînement global est moins performant qu'une prise en compte du type de question comme critère principal d'évaluation.

7.2 Run 2 : par type de question avec caractéristiques (*qtype_featsonly*)

Le deuxième run que nous avons soumis repose sur le même principe, si ce n'est que la prédiction du type de question a été réalisée en amont afin d'entraîner un classifieur spécifique pour chacune des trois catégories. Les meilleurs hyperparamètres trouvés sont les suivants :

— langue formelle : un SGD classifier (Zadrozny & Elkan, 2002) qui atteint 75,41 % en validation,
— langue naturelle : un RandomForest (Breiman, 2001) qui atteint 62,27 % en validation,
— réponse attendue : un Gradient based One-Side Sampling (GOSS) (Ke *et al.*, 2017) qui atteint 75,13 % en validation.

Sans surprise, les notes des réponses en langue naturelle sont plus compliquées à prédire que les deux autres catégories. Nous voyons bien, par ailleurs, que chaque catégorie a ses propres caractéristiques car, à paramètres égaux, les hyperparamètres les plus performants ne sont pas les mêmes pour chacune d'elles.

Finalement, sur l'ensemble des données de test, nous avons atteint une précision de 0,536. Ce résultat est plus faible que ce que nous espérions. Nous pouvons supposer qu'un certain nombre de caractéristiques étaient trop adaptées aux données d'entraînement et pas assez généralisables. Toutefois, ce résultat est meilleur que celui de notre premier run, ce qui semble confirmer que la prise

en compte du type de question dès la phase d'entraînement permet d'obtenir de meilleurs résultats.

7.3 Run 3 : par type de question et vecteurs (*qtype_vect*)

Le troisième run prend lui aussi en compte le type de question en amont afin d'entraîner un classifieur différent pour chacune des catégories. Toutefois, au lieu de prendre en compte la liste de caractéristiques précédente, nous avons souhaité tester la performance de la fonction de hachage et du TF-IDF au niveau des caractères. En effet, d'après les expérimentations réalisées par l'équipe QUEER l'année passée, se placer au niveau du caractère permet d'obtenir de meilleurs résultats (Dupont *et al.*, 2021). Nous supposons que c'est tout particulièrement vrai dans le cas des réponses en langue formelle, où les balises et autres « caractères de code » ont tout autant d'importance que les mots.

Les meilleurs hyperparamètres trouvés sont les suivants :

— langue formelle : un *Dropout Additive Regression Tree* (DART) (Korlakai Vinayak & Gilad-Bachrach, 2015) cumulé à un TF-IDF au niveau des caractères qui atteint 75,04 % en validation,
— langue naturelle : un SGD (Zadrozny & Elkan, 2002) cumulé à une fonction de hachage (Moody, 1988) qui atteint 57,92 % en validation,
— réponse attendue : un *Gradient Boosting Decision Tree* (GBDT) (Friedman, 2001; Ke *et al.*, 2017) cumulé à une fonction de hachage qui atteint 73,29 % en validation.

Avec ces paramètres également, nous observons qu'il est plus difficile de prédire les notes des réponses en langue naturelle. Nous remarquons également un écart légèrement plus important que précédemment entre les performances pour les réponses en langue formelle et pour les réponses attendues, en faveur des premières.

Finalement, cette méthode nous a permis d'obtenir les meilleurs résultats de l'ensemble de nos trois runs puisque nous atteignons ainsi 0,624 de précision. Alors que les scores de validation étaient équivalents à ceux de la méthode du run 2, nous observons qu'ils se sont ici moins dégradés lors de la confrontation aux données de test. Nous pouvons supposer que les paramètres sur lesquels les algorithmes ont appris sont davantage transposables que les caractéristiques calculées précédemment.

8 Conclusion

Dans le cadre de ce défi, nous avons mis en place trois modèles de notation automatique de réponses d'étudiants à des questionnaires. Pour cela, nous avons fait l'hypothèse que les critères d'évaluation divergent en fonction du type de réponse attendue. En effet, la présence de différents types de questions dans le corpus nous a amenés à adopter une approche différenciée. Nous avons essayé de valider notre intuition en mettant en place deux principes : 1) d'une part, une approche sans différenciation et, 2) d'autre part, deux approches différenciées ; l'une se basant sur diverses caractéristiques relevées sur l'ensemble du corpus et l'autre se basant sur une vectorisation du texte. Les résultats obtenus peuvent confirmer l'importance d'adapter le modèle aux différents types de questions.

L'efficacité des méthodes mises en place demeure dépendante de la performance de l'algorithme de

détection du type de question. Une simple amélioration de cet algorithme devrait permettre d'aboutir à de meilleurs résultats. En effet, sur les données de test, sa précision était de 0,92, ce qui laisse une marge de progression. En outre, nous avons remarqué la présence de questions mixtes, qui attendent une réponse précise et également une justification, donc une réponse en langue naturelle. De façon à améliorer les résultats, ce type de questions aurait besoin d'un traitement spécifique, même si on l'imagine bien plus difficile à prendre en compte.

Il aurait également été possible de réaliser davantage de tests au niveau du choix des caractéristiques puisque le run ne les prenant pas en compte (Run 3) est celui qui a obtenu les meilleurs résultats. En effet, nous avons proposé un nombre assez élevé de caractéristiques basées sur des métriques et peut-être que leur pertinence n'était pas toujours avérée et a fini par entraîner des biais dans les modèles liés à un surapprentissage. De plus, nous aurions pu introduire dans les modèles utilisés les caractéristiques basées sur des scores de similarité que nous avions calculés ou encore tester de nouvelles mesures de similarité. Ces mesures demeurent en effet les plus importantes pour cette tâche. Nous aurions également pu nous intéresser davantage aux métriques qu'il est possible de réaliser au niveau des caractères, étant donné que se positionner à ce niveau s'avère souvent assez discriminant, surtout dans le cas des réponses en langue formelle.

Enfin, inclure les pistes que nous avons laissées de côté et pousser plus loin l'idée du traitement spécifique par type de question aurait pu également améliorer les résultats. Jusque-là, nous avons seulement étudié une méthode de matching des réponses attendues, sans pour autant la mettre en place dans le modèle final. Nous avons également réfléchi à des méthodes qu'il pourrait être intéressant de tester pour les deux autres types de questions, sans les avoir implémentées toutefois. Concernant les réponses en langue formelle, faire appel à des outils de correction et de vérification de code pourrait être intéressant. Concernant les réponses en langue naturelle, il pourrait être pertinent d'identifier des mots-clés dans les réponses attendues, de les associer à un lexique de synonymes et de les retrouver ensuite dans les réponses des élèves.

Au regard de l'ensemble de ces observations, et notamment des difficultés auxquelles nous nous sommes confrontés, nous pensons qu'il serait possible assez aisément d'améliorer les résultats de cette tâche si les données de départ étaient davantage harmonisées. Nous proposons cette piste dans le cas où la volonté serait de créer un outil de notation automatique intégré à une plateforme de questionnaires étudiants. En cadrant davantage la manière dont le professeur entre ses propositions de réponse et le barème associé, il serait possible de mettre en place des méthodes plus fiables et par conséquent plus performantes.

Références

AKBIK A., BLYTHE D. & VOLLGRAF R. (2018). Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, p. 1638–1649.

BREIMAN L. (2001). Random forests. *Machine learning*, **45**(1), 5–32.

DUPONT Y., GONZÁLEZ-GALLARDO C.-E., LEJEUNE G., MILLOUR A. & TANGUY J.-B. (2021). Queer@deft2021 : Identification du profil clinique de patients et notation automatique de copies d'étudiants. In *Actes de l'atelier Défi Fouille de Textes@TALN 2020 Classification de cas cliniques et correction automatique de copies d'étudiants. Atelier DÉfi Fouille de Textes*, p. 95–107, Lille, France : Association pour le Traitement Automatique des Langues. QUEER@DEFT2021 : Patients Clinical Profile Identification and Automatic Student Grading.

FRIEDMAN J. H. (2001). Greedy function approximation : a gradient boosting machine. *Annals of statistics*, p. 1189–1232.

GROUIN C., GRABAR N. & ILLOUZ G. (2021). Classification de cas cliniques et évaluation automatique de réponses d'étudiants : présentation de la campagne deft 2021. In *Actes de l'atelier Défi Fouille de Textes@TALN 2020 Classification de cas cliniques et correction automatique de copies d'étudiants. Atelier DÉfi Fouille de Textes*, p. 1–13, Lille, France : Association pour le Traitement Automatique des Langues. Clinical cases classification and automatic evaluation of student answers : Presentation of the DEFT 2021 Challenge.

KE G., MENG Q., FINLEY T., WANG T., CHEN W., MA W., YE Q. & LIU T.-Y. (2017). Lightgbm : A highly efficient gradient boosting decision tree. In I. GUYON, U. V. LUXBURG, S. BENGIO, H. WALLACH, R. FERGUS, S. VISHWANATHAN & R. GARNETT, Éds., *Advances in Neural Information Processing Systems*, volume 30 : Curran Associates, Inc.

KORLAKAI VINAYAK R. & GILAD-BACHRACH R. (2015). DART : Dropouts meet Multiple Additive Regression Trees. In G. LEBANON & S. V. N. VISHWANATHAN, Éds., *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 de *Proceedings of Machine Learning Research*, p. 489–497, San Diego, California, USA : PMLR.

LEVENSHTEIN V. I. *et al.* (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, p. 707–710 : Soviet Union.

MARTIN L., MULLER B., ORTIZ SUÁREZ P. J., DUPONT Y., ROMARY L., DE LA CLERGERIE É., SEDDAH D. & SAGOT B. (2020). CamemBERT : a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 7203–7219, Online : Association for Computational Linguistics.

MOODY J. (1988). Fast learning in multi-resolution hierarchies. *Advances in neural information processing systems*, **1**.

PEDREGOSA F., VAROQUAUX G., GRAMFORT A., MICHEL V., THIRION B., GRISEL O., BLONDEL M., PRETTENHOFER P., WEISS R., DUBOURG V., VANDERPLAS J., PASSOS A., COURNAPEAU D., BRUCHER M., PERROT M. & DUCHESNAY E. (2011). Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

REIMERS N. & GUREVYCH I. (2019). Sentence-bert : Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* : Association for Computational Linguistics.

REIMERS N. & GUREVYCH I. (2020). Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing* : Association for Computational Linguistics.

SUIGNARD P., BENAMAR A., MESSOUS N., CHRISTOPHE C., JUBAULT M. & BOTHUA M. (2021). Participation d'edf r&d à deft 2021. In *Actes de l'atelier Défi Fouille de Textes@TALN 2020 Classification de cas cliniques et correction automatique de copies d'étudiants. Atelier DÉfi Fouille de Textes*, p. 72–81, Lille, France : Association pour le Traitement Automatique des Langues. EDF R&D Participation to DEFT 2021.

WANG X., LIU X. & YUE Y. (2021). Mesure de similarité textuelle pour l'évaluation automatique de copies d'étudiants. In *Actes de l'atelier Défi Fouille de Textes@TALN 2020 Classification de cas cliniques et correction automatique de copies d'étudiants. Atelier DÉfi Fouille de Textes*, p. 63–71, Lille, France : Association pour le Traitement Automatique des Langues. Textual similarity measurement for automatic evaluation of students' answers.

ZADROZNY B. & ELKAN C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 694–699.