Participation d'EDF R&D à DEFT 2022

Philippe Suignard, Xiaomin Huang, Meryl Bothua

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

Participation d'EDF R&D à DEFT 2022

Philippe Suignard, Xiaomi Huang, Meryl Bothua
EDF Lab, 7 bd Gaspard Monge, 91120 Palaiseau, France
philippe.suignard@edf.fr, denise.huang@edf.fr, meryl.bothua@edf.fr

RESUME

Ce papier présente la participation d'EDF R&D à la campagne d'évaluation DEFT 2022. Notre équipe a participé aux deux tâches proposées, l'une sur la prédiction automatique de la note d'un étudiant.e pour sa réponse à une question, d'après une référence existante, la seconde, nouvelle, qui était une tâche de prédiction itérative des notes. Notre équipe s'est classée 1ère sur la première tâche et a été la seule contributrice sur la seconde. Le corpus se composait d'énoncés en informatique avec la correction de l'enseignant et les réponses des étudiant.e.s par question.

ABSTRACT

EDF R&D Participation to DEFT 2022.

This paper describes the participation of EDF R&D at DEFT 2022. Our team worked on the two tasks proposed. This edition included one new task dealing with active learning to predict evaluation on students' answers to specific questions and iteratively improve our method. The corpus was composed of questions about computer science, teacher's correction and students' answers. We finished first for the first task and we were the only team to contribute to the second task.

MOTS-CLÉS: détection de similarité sémantique, SentenceTransformer, Apprentissage Actif, Soft cardinalité
KEYWORDS: Semantic Similarity Detection, SentenceTransformer, Active Learning, Soft cardinality.

1 Introduction

L'édition 2022 du défi fouille de textes (Grouin et al., 2022) portait sur la correction automatique de copies électroniques d'étudiant.e.s (suite des travaux réalisés en 2021) avec deux tâches : une **tâche de base** qui était la même que la tâche 2 de l'édition 2021 et une **tâche continue** qui consistait à interroger le serveur pour récupérer des données, entrainer un système de prédiction de notes,

Atelier DÉfi Fouille de Textes (DEFT), pages 45–54.

l'appliquer sur des données récupérées puis interroger le serveur pour y déposer les notes et récupérer d'autres données et ainsi de suite.

Participer à DEFT est l'occasion de tester des méthodes de calcul de similarité dont les résultats contribuent directement à EDF Commerce et à d'autres entités du groupe EDF.

2 Tâche de base

2.1 Présentation

Cette tâche a pour but d'évaluer les réponses des étudiant.e.s à des questionnaires, c'est à dire à fournir une note comprise entre 0 et 1, en prenant pour référence la correction produite par l'enseignant.

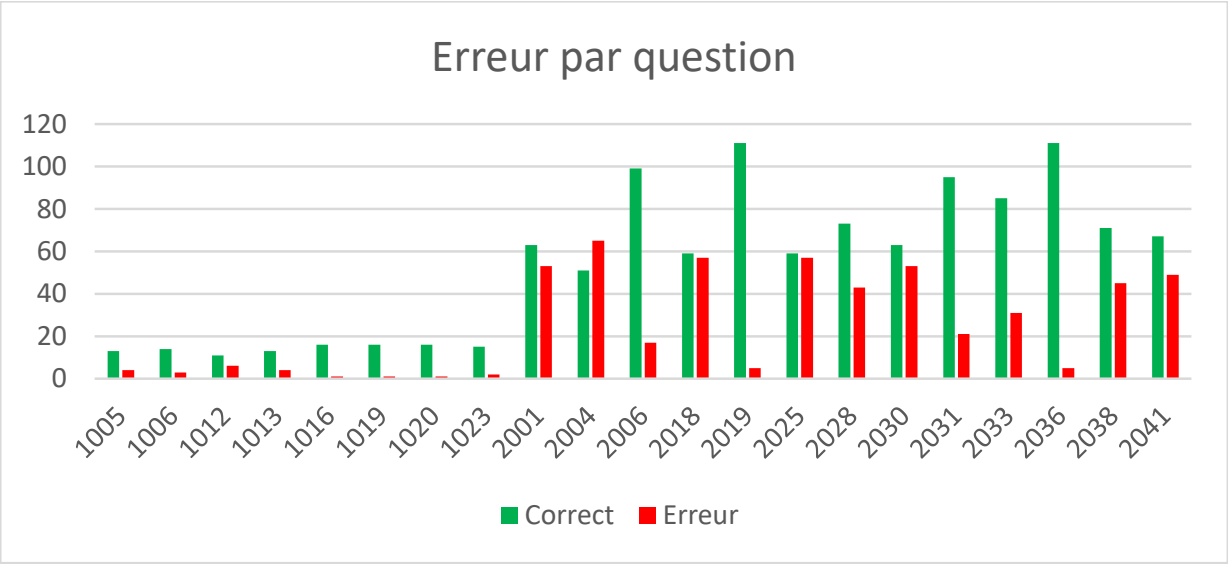
Pour cette tâche, nous avons réutilisé le système développé en 2021 qui nous avait permis d'obtenir la 1ère place sur les tâches 2 et 3, (Suignard, 2021). La démarche mise en œuvre pour résoudre cette tâche est la suivante :

- Pré-traitement des données textuelles ;
- Calcul de « features » ;
- Entrainement d'un classifieur ;
- Application du classifieur sur les données de test.

Nous avons utilisé les données de Train et Test 2021 de la tâche 2 pour constituer le nouveau corpus d'apprentissage. Les données de la tâche 3 ne sont pas utilisées, car elles ne contiennent pas la réponse attendue par l'enseignant.

2.2 Analyse des erreurs du système de 2021

Dans un premier temps, nous analysons les erreurs de notre système développé l'an dernier sur les données de test 2021 :



En abscisse, se trouve le numéro de la question (de 1005 à 2041). Pour chaque question, est affiché le nombre de « copies » correctement classées (en vert) et le nombre d'erreurs (en rouge).

Pour la question 2004, les erreurs sont assez importantes. Une des raisons vient du fait que la réponse de l'enseignant est rédigée à la voix passive « b = mise en forme physique (gras), strong = mise en forme logique (mise en valeur) » alors que les étudiant.e.s utilisent plutôt la voix active « la balise <b> met… » ou « <b> permet de mettre… ». Comme les *features* sont des similarités entre mots, on va légèrement modifier la réponse de l'enseignant en ajoutant des formulations : « b = mise en forme, met en forme, mettre en forme, etc. ».

Pour la question 2018, on constate un nombre assez important d'erreurs dans le corpus, puisque la réponse « Notes déposées le 24/03/2016 » est considérée comme fausse 19 fois mais exacte 40 fois ! Le corpus est corrigé.

Pour la question 2041, la réponse de l'enseignant est « typage du contenu », contenu étant un terme générique. Les bonnes réponses des étudiant.e.s parlent de « type des données » ou « type des attributs ». On ajoute ainsi ces éléments à la réponse de l'enseignant pour augmenter les scores de similarité.

2.3 Les prétraitements

2.3.1 Indications et consignes de l'enseignant

Comme en 2021, la réponse de l'enseignant est constituée de deux parties :

- La réponse attendue ;
- Une série de commentaires ou de précisions. Les deux parties sont séparées par « ---- ».

Ces éléments complémentaires sont sans doute très utiles pour le correcteur, mais peuvent introduire des biais pour notre apprentissage machine :

- « réponse attendue », parfois présent, est superflu ;
- A la question 3021 : « 1 si petite faute sur le texte », comment définir une « petite faute » ?
- A la question 3030 : « 1 point pour la définition, 1 point pour l'exemple », qu'est qu'une définition, qu'est-ce qu'un exemple ?

Tout comme en 2021, ces éléments complémentaires ou précisions sont supprimées. Les questions suivantes sont légèrement modifiées :

- Question 3005 : changement de « i = mise en forme physique em= mise en forme logique » par « i = mise, met, mettre en forme physique em= mise, met, mettre en forme logique »
- Question 3042 : changement de « Les liens doivent décrire leur destination » par « Les liens doivent décrire leur destination, alt, alternatif »

2.3.2 Prétraitements généraux

Les traitements suivants sont appliqués aux données :

- Normalisation des balises "<", ">" en « < » et « > » ;
- Les balises , ,
 en début et fin de texte ont été supprimées ;
- Remplacement des caractères " " par un blanc ;
- Suppression des caractères \n et \t ;
- Insertion d'un caractère blanc avant « < » et après « > » pour faciliter la tokenisation en mots pour les calculs de similarité ;
- Utilisation du caractère blanc pour la séparation des phrases en tokens ;
- Passage en minuscule.

2.4 Les « *features* » utilisées

A partir de *q* la question posée, *a* la réponse de l'élève (« answer ») et *ra* la réponse proposée par l'enseignant (« request answer »), l'idée consiste à calculer des *features* et similarités croisées (entre *a* et *q*, *a* et *ra*, *q* et *ra*) pour ensuite entrainer un classifieur. Nous avons repris les 42 *features* de 2021, auxquelles nous ajoutons une similarité cosinus sur les bigrammes de lettres entre « a » et « q+ra ».

Les *features* sont la softcardinalité (Gimenez, 2015), les similarités de Monge-Elkan, Jaro-Wikler, Damereau-Levenshtein calculées sur les mots et bigrammes et trigrammes de lettres, Cf l'article initial pour plus de précisions.

2.5 Les différents « Run »

Une fois les *features* calculées, un classifieur est entraîné à l'aide du logiciel Weka (Hall, 2009). Plusieurs classifieurs ont été testés et c'est « Random Forest » qui a obtenu le meilleur score sur les données d'entraînements. Pour les « run » 1 et 2, le nombre de classes à prédire a été ramené à 3 :

- 0 si la note était inférieure à 0,25 ;
- 0,5 si la note était comprise entre 0,25 et 0,75 ;
- 1 si la note était supérieure à 0,75.

Le run 1 est entrainé avec 100 arbres de décision et le run 2 avec 200.

Pour le run 3 (100 arbres), le classifieur a été entrainé à prédire la « vraie » note comprise entre 0 et 1 (c'est-à-dire sa valeur numérique). Sur le jeu de test, la valeur prédite a ensuite été arrondie au dixième le plus proche.

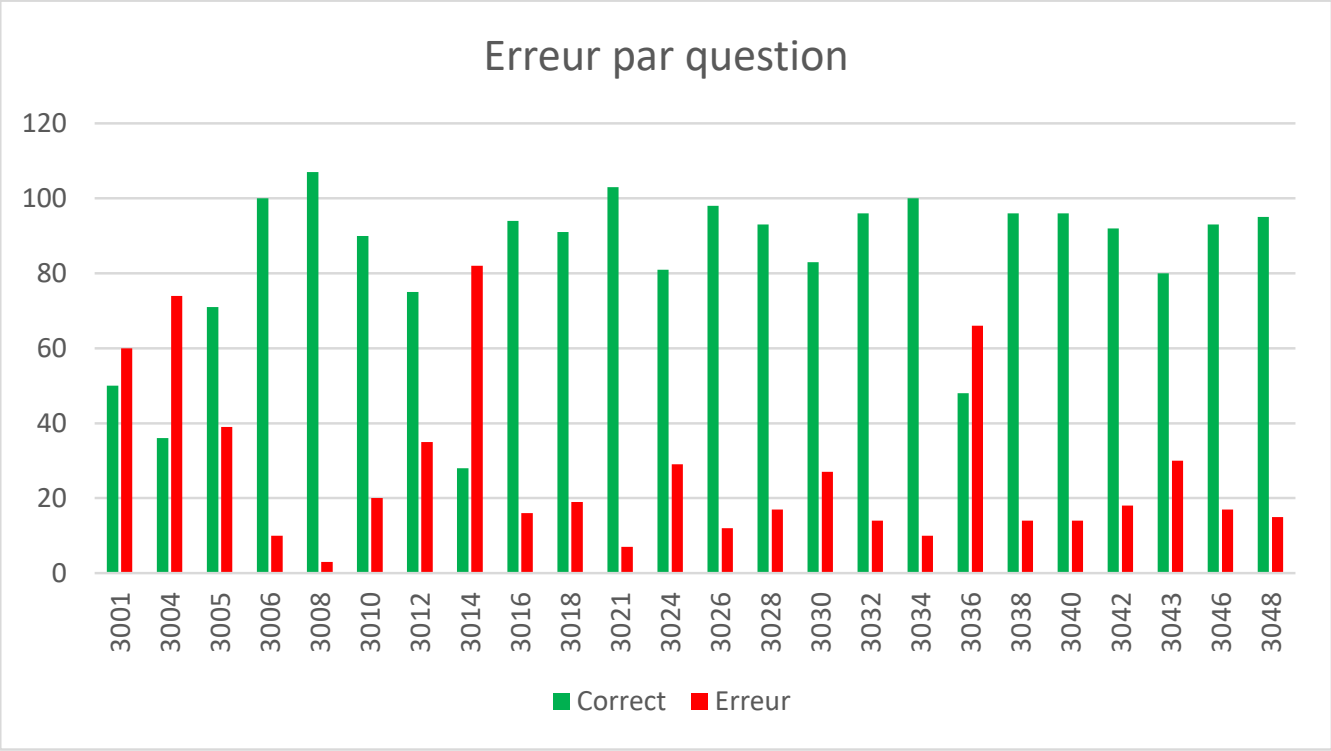
2.6 Résultats

Run	Evaluation
Run 1 :	0,752
Run 2 :	0,756
Run 3 :	0,323
Maximum	0,756
Médiane	0,524
Moyenne	0,542
Minimum	0,323

Tableau 3 : résultats de la tâche de base

2.7 Analyse des erreurs

Les résultats des run étaient fournis par les organisateur.trice.s. de la compétition ainsi que les notes attendues pour les copies du test. Comme au §2.2, nous avons ainsi pu faire une analyse des erreurs du run 2 (celui ayant obtenu le meilleur score), erreurs strictes (un 1 au lieu de 0.8 est considéré comme une erreur) :



On voit ainsi les questions qui ont généré le moins d'erreurs comme les questions 3008 ou 3021 ainsi que celles qui ont généré le plus d'erreurs :

- **Question 3001** : Dans la consigne pour les correcteurs, il y avait 3 niveaux de notations (0, 0.5 et 1) avec des distinctions strictes selon les termes utilisés (« domaine », « serveur », « site »,

« machine », « adresse » et « page »), distinctions peut-être trop subtiles pour notre classifieur ;

- **Question 3004** : à quoi sert la balise « title » de l'en-tête d'une page html ? La réponse attendue était très précise : « titre de la page indiqué par le navigateur ». Toutes les réponses qui déviaient légèrement, obtenaient une note 0.8, voire moins, ce qui explique le nombre important d'erreurs ;
- **Question 3014** : il fallait indiquer le code HTML du champ dans lequel l'utilisateur indiquerait une durée d'intervention. Beaucoup d'erreurs proviennent du fait que la méthode a attribué une note de 1, alors que la note attendue était 0.8 parce que la réponse contenait une erreur (« int », « text », « numeric » au lieu de « number ») et/ou que le nom du champ de saisie (« minutes », « temps », « intervention », « duree intervention »…) était différent de celui fourni par l'enseignant (« duree »). D'autres erreurs venaient du fait que la note attendue était 0,5 (parce que par exemple un seul des deux éléments étaient fournis, type ou nom de la variable à saisir), mais comme la méthode génère plutôt des 1 ou des 0, elle se trompe assez souvent ici ;
- **Question 3036** : la question portait sur une balise parfois orthographié « <Canal> » et parfois « <canal >». Comme la méthode commençait par mettre tous les textes en minuscule, c'est assez logique de la voir commettre beaucoup d'erreurs.

3 Tâche continue

3.1 Présentation et grands concepts d'Active Learning

La tâche continue consiste à concevoir un système de prédiction en intégrant des interactions entre un annotateur et un oracle. Dans notre cas, cela signifie que notre système de prédiction sera amélioré en permanence par la requête de notes réelles d'étudiant.e.s. Le serveur joue un rôle d'oracle qui retourne des notes réelles ou corrige des résultats de la prédiction. Notre système se charge de la prédiction et de choisir des données stratégiques à interroger. L'enjeu de la tâche est d'augmenter la capacité du système à prédire rapidement les bonnes notes, tout en minimisant le nombre de requêtes, sans pour autant chercher à ne faire qu'une seule requête pour l'ensemble du corpus.

Des scripts Python d'interrogation et de soumission sont fournis par les organisateur.trice.s. L'interaction entre le système de prédiction et le serveur se réalise par les trois étapes à chaque itération :

1) Demander la note d'un étudiant.e sur une question avec le script d'interrogation (script n°1)
2) Générer un fichier de prédiction pour tous les étudiant.e.s sur cette question et le déposer sur le serveur, avec les scripts de soumission (scripts n°2 et 3)
3) Demander une nouvelle note (script n°1), affiner son modèle, faire de nouvelles prédictions, sauvegarder et envoyer (scripts n°2 et 3).
4) Itération sur l'ensemble des questions.

L'ensemble de ces étapes correspond aux principes de l'apprentissage actif (*Actif Learning*), qui est un apprentissage semi-supervisé. Pendant l'étape de l'apprentissage (*train*), le modèle interagit avec l'oracle (serveur) pour obtenir des informations sur l'étiquette des nouvelles données. Le modèle n'est censé acquérir que les étiquettes qui améliorent réellement la capacité de prédiction. Ainsi, on peut faire référence à la stratégie basée sur l'échantillonnage par incertitude (*uncertainty sampling*) qui fait

partie des différentes stratégies d'interrogation en Active Learning : dans notre cas il s'agit de requêter dans la base de données, celles pour lesquelles le modèle actuel est le moins certain.

L'apprentissage actif porte souvent sur un ensemble de données non étiquetées. Les étapes sont les suivantes :

1) Etiqueter manuellement un très petit sous-échantillon des données.
2) Une fois que l'on dispose d'une petite quantité de données étiquetées, le modèle doit être entraîné sur celles-ci.
3) Une fois le modèle formé, prédire la classe des données non étiquetées restant.
4) Un score de priorité est associé à chaque donnée non étiquetée en fonction de la prédiction du modèle (les scores les plus couramment utilisés sont entre autres le score de confiance, l'entropie et l'échantillon de marge)
5) Une fois que la stratégie a été choisie pour prioriser l'étiquetage, ce processus peut être répété de manière itérative : un nouveau modèle peut être entraîné sur un nouvel ensemble de données étiquetées. Une fois que le nouveau modèle a été entraîné, les données non étiquetées peuvent être soumises au modèle afin de mettre à jour les scores de priorité pour continuer l'étiquetage. De cette façon, on peut continuer à optimiser la stratégie d'étiquetage au fur et à mesure que le modèle s'améliore.

3.2 Développements réalisés

3.2.1. Prétraitements réalisés

Nous travaillons avec les fichiers « trainT2-Q.tab » et « trainT2-R.tab » fournis par les organisateur.trice.s. Nous requêtons des notes puis, en fonction du résultat retourné par le serveur, nous cherchons la réponse et la question correspondantes dans les fichiers.

Afin de faciliter la lecture des données, nous réorganisons des données de ces deux fichiers sous forme de dictionnaire python. Voici un exemple : {'1001': [['student101','0.5','Ce sont les pages web accessible par tout navigateur.'], ['student108', '0', 'Un réseau mondial'],…}). Des commentaires et des précisions (y compris une réponse de référence) apparus dans une question ne sont pas pris en compte car ils rajoutent du bruit et perturbent la performance du classifieur lors de la vectorisation des données.

Comme indiqué dans la tâche 1, les notes sont regroupées en 3 catégories. Le nettoyage des données est identique à celui de la tâche1. Nous concaténons chaque réponse avec sa question, puis nous les transformons en vecteur comme entrée du modèle en utilisant Sentence-BERT.

3.2.2. Système développé

La chaine de traitement pour la phase d'apprentissage est composée de deux étapes principales : (1) Etape préparatoire (2) Etape d'apprentissage et de prédiction.

Nous choisissons le RadamForest comme classifieur. A chaque apprentissage, 50% des données dans le corpus d'apprentissage (aussi appelé "train pool") sert à l'entraînement et 50% au test. Nous

utilisons la valeur de la probabilité de prédiction comme score de confiance, soit la fonction predict_proba() de Sklearn.

Etape préparatoire

Dans la phase de l'apprentissage, il n'existe initialement aucune donnée. L'objectif de cette étape est de mettre en place un corpus d'apprentissage, qui permette de mettre en route un classifieur. Il est à noter que chaque donnée dans le corpus d'apprentissage est fiable et obtient un score de confiance de "1".

Pour ce faire, nous lançons d'abord lancer le script n° 1 pour demander les premières données à savoir la note d'un étudiant.e sur une question. A cette étape préparatoire, nous utilisons une méthode simple : on lance des requêtes selon l'ordre des étudiant.e.s et des questions jusqu'à obtenir les trois catégories de notes.

Plus précisément, on lance une première requête (*query*) pour obtenir un premier exemple de donnée issu de la base (eg: pour la question 1001, on veut récupérer la note de l'étudiant.e101. Le résultat retourné est 0.5). Cet étudiant.e requêté est considéré comme un étudiant.e de référence. A partir de la première donnée obtenue, le modèle prédit les notes de tous les autres étudiant.e.s à la question 1001. Le système créé et soumet un fichier avec l'étudiant.e de référence, les notes prédites et des scores de confiance. Naturellement, la note à ce stade sera de 0.5 pour chaque étudiant.e avec un score de confiance de 1, le modèle ne connaissant rien d'autre : il n'existe qu'une seule donnée dans le *train pool* et le classifieur est ainsi entrainé avec une donnée. Lorsque ce cycle est terminé, on requête alors un autre étudiant.e sur la même question 1001. Si cette note est différente de 0.5, alors le système génère de nouveau scores de confiance et de nouvelles prédictions. Cette itération se répète jusqu'à ce que le nombre des catégories des notes atteigne à trois.

Cette étape préparatoire est très importante car ces scores de confiance orientent le choix des prochains étudiant.e.s à cibler pour améliorer l'apprentissage. Les étudiant.e.s dont les notes ont un score de confiance bas, seront interrogés. A la sortie de l'étape, on obtient un corpus d'apprentissage avec trois catégories de notes ainsi qu'un étudiant.e avec un score de confiance bas.

Etape d'apprentissage et de prédiction

A cette étape, nous regardons si la note d'un étudiant.e avec un score de confiance bas est correctement prédite. Pour une question, si le modèle arrive à prédire la note de l'étudiant.e le plus incertain, nous pouvons alors supposer que l'apprentissage sur cette question a atteint un niveau idéal. Cette condition permet de garantir que même la note de l'étudiant.e la plus difficile à prédire est correcte :

- Si le score de confiance de la note prédite pour un étudiant.e est bas et que cette note n'est pas déjà présente dans notre corpus d'apprentissage, le système requête sa note de référence (script 1) et ajoute la note corrigée au corpus d'apprentissage. Cet étudiant.e requêté est un étudiant.e de référence. Cette étape permet de s'assurer que toutes les données ajoutées à notre corpus d'apprentissage sont fiables. A chaque itération, on relance un nouvel apprentissage avec les données du corpus d'apprentissage et teste la nouvelle configuration du modèle.

- Si le score de confiance de la note prédite pour un étudiant.e est bas mais que cette note est déjà dans le corpus d'apprentissage, le système vérifie que la note prédite est correcte. Si elle est correcte, on estime que l'apprentissage du modèle est optimal. On arrête ainsi le processus de requêtage et passe à la question suivante. Au contraire, si cette note est déjà dans le corpus mais qu'elle n'est pas correctement prédite, on relance l'entrainement et recommence le processus de prédiction.

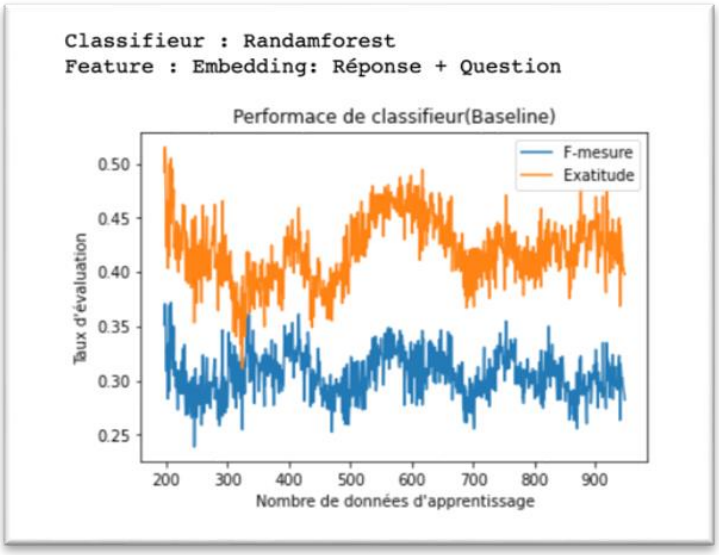
Le modèle est appris sur les 50 questions et est mis à jour chaque fois qu'une nouvelle donnée est ajoutée dans le corpus d'apprentissage. Dans chaque fichier de prédiction, l'étudiant.e de référence est noté et les étudiant.e.s apparus dans le corpus d'apprentissage ont un score de confiance de « 1 ». Etant donné que le nombre de requêtes doit être minimal, il est attendu d'avoir le moins de score de confiance de « 1 » possible dans le fichier de prédiction.

3.3 Résultats

Durant la phase d'apprentissage, la performance du modèle obtient les valeurs suivantes : l'exactitude (*accuracy*) en moyenne est de 0.42 et la F-mesure en moyenne est de 0.31. D'après la courbe d'apprentissage, nous constatons que la performance atteint son maximum au début, puis elle commence à diminuer. Lorsque le nombre de données d'apprentissage atteint environ 480, l'exactitude (*accuracy*) commence à s'améliorer. Quand le nombre de données d'apprentissage est environ de 600, la performance commence à se stabiliser.

	Moyenne	*Maximum*	*Minimum*
Accuracy	*0.42*	*0.51*	*0.31*
F-mesure	*0.31*	*0.37*	*0.24*

Tableau : Performance du modèle pour la phase d'apprentissage



Courbe d'apprentissage et de performance du modèle

Concernant le nombre de données dans le corpus d'apprentissage, nous avons 949 de données. Ce total ne représente que 24.8% sur l'ensemble des données du corpus de base. Ce nombre correspond également au nombre d'interrogations du serveur et au nombre de réapprentissage du modèle. Notre méthode d'apprentissage avec un score de confiance est moins couteuse que du Machine Learning classique car elle permet de ne sélectionner que les données avec un score de confiance bas et donc ne réduire la taille du corpus d'apprentissage. Par exemple pour la question 1002, sur 17 étudiant.e.s ayant répondu, nous n'avons vérifié la note que de 6 étudiant.e.s.

- Nombre de données dans le corpus d'apprentissage : 949 (24.8% du corpus de base)
- Nombre de données dans le corpus de base (fichier « Train-T2-R.tab ») : 3 820
    - Pour les questions 1001 à 1028 (20 questions, 17 étudiant.e.s) : 340
    - Pour les questions 2002 à 2046 (30 questions, 116 étudiant.e.s): 3 480

4 Conclusion

La participation à la campagne DEFT 2022 nous a permis de tester des méthodes de la détection de similarité sémantique et d'apprentissage actif. Ces méthodes pour la fouille de texte pourront être éventuellement appliqués sur les données textuelles au sein de EDF Commerce et d'autres entités du groupe EDF.

Références

GROUIN, C. & ILLOUZ, G. (2022). Notation automatique de réponses courtes d'étudiants : présentation de la campagne DEFT 2022. In : Actes de DEFT. Avignon.

HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., & WITTEN, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, *11*(1), 10-18.

JIMENEZ, S., GONZALEZ, F. A., & GELBUKH, A. (2015). Soft cardinality in semantic text processing: experience of the SemEval international competitions. *Polibits*, (51), 63-72.

SUIGNARD, P., BENAMAR, A., MESSOUS, N., CHRISTOPHE, C., JUBAULT, M., & BOTHUA, M. (2021). Participation d'EDF R&D à DEFT 2021 (EDF R&D Participation to DEFT 2021). In Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles. Atelier DÉfi Fouille de Textes (DEFT) (pp. 72-81).