Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles
Lille, France, 28 juin au 2 juillet 2021

Participation d'EDF R&D à DEFT 2021

Philippe Suignard, Alexandra Benamar, Nazim Messous,
Clément Christophe, Marie Jubault, Meryl Bothua
(1) EDF Lab, 7 bd Gaspard Monge, 91120 Palaiseau, France
philippe.suignard@edf.fr, alexandra.benamar@edf.fr, nazim.messous@edf.fr,
clement.christophe@edf.fr, marie.jubault@edf.fr, meryl.bothua@edf.fr

RESUME

Ce papier présente la participation d'EDF R&D à la campagne d'évaluation DEFT 2021. Notre équipe a participé aux deux dernières tâches proposées (T2 et T3), deux tâches sur le calcul de similarité sémantique entre textes courts, et s'est classée 1ère sur ces deux tâches. Cette édition proposait deux nouvelles tâches pour l'évaluation automatique de réponses d'étudiants à des questions d'enseignants. Le corpus se composait d'une centaine d'énoncés en informatique avec la correction de l'enseignant et les réponses d'une cinquantaine d'étudiants en moyenne par question, sur 2 ans. La tâche 2 consistait à évaluer les réponses des étudiants en prenant pour référence la correction produite par l'enseignant et la tâche 3 à évaluer les réponses d'étudiants à partir d'un ensemble composé d'un énoncé et de plusieurs réponses d'étudiants déjà corrigées par l'enseignant.e.

ABSTRACT

EDF R&D Participation to DEFT 2021.
This paper describes the participation of EDF R&D at DEFT 2021. Our team worked on the second and the third tasks (T2 and T3). This edition included two new tasks dealing with automatic evaluation on students' answers to specific questions. The corpus was composed of a hundred questions about computer science, teacher's correction and students' answers. The second task consisted in evaluating students' answers regarding teachers' suggestions and the third task in evaluating students' answers regarding other students' answers, already corrected by the teacher. We finished first for both tasks.

MOTS-CLÉS : détection de similarité sémantique, CamemBERT, Soft cardinalité
KEYWORDS: Semantic Similarity Detection, CamemBERT, Soft cardinality.

1 Introduction

Une partie de l'édition 2021 du défi fouille de textes (Grouin *et al.*, 2021), à savoir les tâches 2 et 3, portait sur des calculs de similarité entre phrases. 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 2 : « Evaluation automatique de copies d'après une référence existante »

2.1 Présentation

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

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.

2.2 Les prétraitements

2.2.1 Prétraitements des runs 1 de T2 et T3 et remarques sur les indications et consignes de l'enseignant

Dans le fichier « trainT2.tab », la 5ème colonne est constituée de la réponse que l'enseignant aimerait trouver. En plus de la réponse attendue, l'enseignant vient parfois ajouter des commentaires ou des précisions. Ces éléments supplémentaires sont très utiles pour le correcteur, mais peuvent introduire des biais pour notre apprentissage machine :

- 19 questions sur les 50 contiennent ce genre de précisions. 31 autres n'en contiennent pas ;
- A la question 2014, il est précisé : « 0,7 si réponse imprécise ». Ce genre de commentaire est très difficilement interprétable par une machine ;
- A la question 1004 « Quelle balise HTML contient les informations destinées au navigateur et aux moteurs de recherche ? », la consigne de l'enseignant est : « head 0 si html 0.5 si meta », c'est-à-dire qu'il considère la réponse « head » comme bonne, la réponse « html » comme fausse et « meta » comme à moitié bonne. Comme nos *features* sont obtenues à partir de calculs de similarités, les réponses « head », « html » ou « meta » risque d'obtenir les mêmes scores. Dans d'autres cas, il est mentionné « 1 si l'étudiant répond ceci, 0.7 s'il dit cela, 0.5 s'il mentionne tel ou tel mot, etc. ».

De manière générale, ces commentaires ou précisions viennent « perturber » les classifieurs. On décide donc de les enlever pour les runs 1. Les résultats obtenus sur les données d'apprentissage montrent un gain notamment pour prédire la note 0,5.

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.

2.2.2 Prétraitements réalisés pour les runs 2 et 3 de T2 et T3

Pour les runs 2 et 3 des tâches 2 et 3, nous avons réalisé six prétraitements que nous détaillons ci-dessous.

- **Suppression des tags html** : nous avons repéré une liste de tags utilisés uniquement pour la mise en forme : écriture en gras, affichage sous forme de liste et saut de ligne. Nous avons donc supprimé les tags suivants :
  -

  -
  -
  -

  Ces suppressions ont été effectuées sur les fichiers questions réponses des tâches 2 et 3.

- **Suppression ou modification des entités html** : après analyse du fichier trainT2-Q.tab fourni pour la tâche 2, nous avons constaté la présence de plusieurs entités html que nous avons supprimées ou remplacées :
  - Le code entité html de l'espace insécable * *
  - Le code entité html du signe < *&lt.*
  - Le code entité html du signe > *&gt.*

  L'entité html * * a été supprimée de l'ensemble de nos données. Les entités *&lt* et *&gt*, n'ont pas été supprimées car elles sont parfois dans l'objet des questions et les suggestions de l'enseignant. Ces entités sont de plus utilisées pour délimiter le début et la fin de chaque balise html. En d'autres termes, ce ne sont pas des entités superflues. Leur suppression ferait perdre de l'information et influerait sur la similarité entre une réponse suggérée par l'enseignant et la réponse d'un étudiant. Les entités *&lt* et *&gt* ont donc été remplacées par les caractères '<' et ' >'. Voici un exemple ligne 3 du fichier trainT2-Q.tab :

  Quelle est la déclaration de type de document (doctype) d'une page web en HTML 5 (première ligne de la page HTML) ?
 <!DOCTYPE html>

  Devient à la place :

  Quelle est la déclaration de type de document (doctype) d'une page web en HTML 5 (première ligne de la page HTML) ?
 <!DOCTYPE html>

- **Suppression des sauts de ligne et des espaces supplémentaires** : nous avons supprimé les caractères \n et \t qui se trouvaient dans le fichier trainT2-Q.tab et le fichier trainT2-R3.tab.

- **Restructuration de l'information** : nous avons restructuré les deux fichiers trainT2-Q.tab et trainT2-R.tab dans un nouveau fichier .json en vue de l'application de nos modèles et notamment pour nos calculs de similarité.
  Pour la question, "Quelle structure de données la fonction pg_fetch_assoc retourne-t-elle ?", la réponse de l'enseignant est " un tableau associatif 0.7 si tableau ou

array 0.5 pour tuple
 ". Après application des prétraitements, les suggestions sont : 1. tableau ou array, 2. un tableau associatif, 3. Tuple.
Ces différentes suggestions et leurs notations sont très importantes. Nous avons donc divisé la réponse de l'enseignant en plusieurs parties (chaque partie est une suggestion de l'enseignant avec la note associée), puis sauvegardé ces différentes parties séparément en précisant qu'il s'agit de la réponse d'un enseignant. Nous avons ensuite renseigné pour une question donnée les réponses des élèves contenues dans le fichier trainT2-R.tab avec la note associée, en précisant qu'il s'agit d'une réponse élève. Cela nous permettra par la suite pour une question donnée de comparer la similarité entre la réponse donnée par chaque élève et chaque suggestion que l'enseignant a fait. Voici un exemple de résultat obtenus pour la question 1004 :
{**'numero question'**: '1004', **'text'**: 'Quelle balise HTML contient les informations destinées au navigateur et aux moteurs de recherche ?', **'Reponses'**: [{**'AuteurReponse'**: 'Enseignant', **'reponse'**: ' meta', **'note'**: 0.5}, {'AuteurReponse': 'Enseignant', 'reponse': ' html ', 'note': 0}, {'AuteurReponse': 'Enseignant', 'reponse': 'head ', 'note': 1}, {'AuteurReponse': 'eleve', 'id_etudiant': 'student101', 'reponse': 'NO_ANS', 'note': '0'}, {'AuteurReponse': 'eleve', 'id_etudiant': 'student108', 'reponse': 'la balise head <head></head>', 'note': '1'}

- **Suppression de la ponctuation et des lettres capitales.**

2.3 Les « features » utilisées

2.3.1 La « Softcardinalité »

La cardinalité d'un ensemble désigne le nombre d'éléments appartenant à cet ensemble. Si $\boldsymbol{S} = \{"\boldsymbol{a}","\boldsymbol{b}","\boldsymbol{c}"\}$, alors $|\boldsymbol{S}| = \boldsymbol{3}$. Ce nombre est indépendant des relations qu'entretiennent, entre eux, les différents éléments de l'ensemble. Mais si *a* et *b* sont proches l'un de l'autre d'un point de vue sémantique ou d'un point de vue lexical, on pourrait imaginer que cette cardinalité soit plus faible (2.5 ou 2.8 par exemple). C'est ce que permet la « softcardinalité » introduite par (Jimenez, 2015). Si $\boldsymbol{S} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_n}\}$, on définit la softcardinalité par $|\boldsymbol{S}|' = \sum_{i=1}^{n} \frac{\boldsymbol{1}}{\sum_{j=1}^{n} \boldsymbol{sim}(\boldsymbol{s_i}, \boldsymbol{s_j})}$. La similarité entre deux termes étant définie ici par : $\boldsymbol{sim}(\boldsymbol{t_1}, \boldsymbol{t_2}) = \frac{\boldsymbol{2} * \left|\boldsymbol{t_1}^{[2:3]} \cap \boldsymbol{t_2}^{[2:3]}\right|}{\left|\boldsymbol{t_1}^{[2:3]}\right| + \left|\boldsymbol{t_2}^{[2:3]}\right|}$ avec $\boldsymbol{t}^{[2:3]}$ l'ensemble des bigrammes et trigrammes du terme t. Si $\boldsymbol{t} = "\boldsymbol{maison}"$,
$\boldsymbol{t}^{[2:3]} = \{"\boldsymbol{ma}","\boldsymbol{ai}","\boldsymbol{is}","\boldsymbol{so}","\boldsymbol{on}","\boldsymbol{mai}","\boldsymbol{ais}","\boldsymbol{iso}","\boldsymbol{son}"\}$

Cette méthode est utilisée car elle avait produit de très bons résultats lors d'une compétition similaire de SemEval 2013 (Dzikovska, 2013).

2.3.2 Similarité de Monge-Elkan

Soit deux suites de mots de longueurs différentes : $\boldsymbol{S} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_n}\}$ et $\boldsymbol{T} = \{\boldsymbol{t_1}, \boldsymbol{t_2}, \dots, \boldsymbol{t_p}\}$. La similarité de Monge-Elkan entre ces deux suites de mots est définie de la manière suivante à partir d'une similarité entre deux mots sim_{mot} :

$$sim_{MongeElkan}(S, T, sim_{mot}) = \frac{1}{|S|} * \sum_{i=1}^{|S|} \underset{j=1 \text{ à } |T|}{max} sim_{mot}(s_i, s_j)$$

Pour la similarité entre les mots sim_{mot}, on utilisera une similarité stricte (1 si les mots sont identiques et 0 sinon) ou une similarité de type Damereau-Levenshtein. La similarité de Monge-Elkan n'étant pas symétrique, on construit la similarité symétrique suivante :

$$sim_{ME}(S, T, sim_{mot}) = \sqrt{sim_{MongeElkan}(S, T, sim_{mot}) * sim_{MongeElkan}(T, S, sim_{mot})}$$

2.3.3 Autres similarités

Les autres *features* utilisent les similarités plus connues comme cosinus, Jaro-Wikler, Damereau-Levenshtein et qui n'ont pas besoin d'être présentées.

2.4 Les différents Run

2.4.1 Run1

***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*). L'article initial de (Jimenez, 2015) en calculait 42. Mais un premier calcul d'importance des *features* pour la classification a montré que les similarités les plus importantes étaient celles calculées entre *a* et *ra*, c'est-à-dire entre la réponse de l'élève et la réponse proposée par l'enseignant, ce qui paraît assez logique. On a donc réduit le nombre de *features* de l'article initial de 42 à 18 et en avons ajouté d'autres basées sur des similarités croisées entre *a*, *q* et *ra*. Au total, nous avons également 42 *features*.

Quelques précisions : dans le tableau suivant, $X \backslash Y$ correspond à l'ensemble X auquel est enlevé Y. $X \cap Y$ correspond à l'intersection (au niveau des mots) des chaines de caractères X et Y. $X \cup Y$ correspond à l'union (au niveau des mots) des chaines de caractères X et Y.

$\|a\|'$	$\frac{\|a \cap ra\|'}{\min(\|a\|', \|ra\|')}$	$cosinus_{bi}(a, ra)$
$\|q\|'$	$\frac{\|a \cap ra\|'}{\max(\|a\|', \|ra\|')}$	$cosinus_{bi}(q, ra)$
$\|ra\|'$	$\frac{\|a \cap ra\|' * (\|a\|' + \|ra\|')}{2 * \|a\|' * \|ra\|'}$	$cosinus_{tri}(a, q)$[1]
$\|a \cup q\|'$	$\|a \cup ra\|' - \|a \cap ra\|'$	$cosinus_{tri}(a, ra)$
$\|a \cup ra\|'$	$cosinus_{mot}(a, q)$[2]	$cosinus_{tri}(q, ra)$
$\|q \cup ra\|'$	$cosinus_{mot}(a, ra)$	$cosinus(a, q \cup ra)$
$\|a \cap ra\|'$	$cosinus_{mot}(q, ra)$	$cosinus_{bi}(a, q \cup ra)$
$\|a \backslash ra\|'$	$similarité_{Damereau\_Levenshtein}(a, q)$[3]	$cosinus_{tri}(a, q \cup ra)$
$\|ra \backslash a\|'$	$similarité_{Damereau\_Levenshtein}(a, ra)$	$sim_{ME}(a, q, sim_{DamereauLevenshtein})$[4]
$\frac{\|a \cap ra\|'}{\|a\|'}$	$similarité_{Damereau\_Levenshtein}(q, ra)$	$sim_{ME}(a, ra, sim_{DamereauLevenshtein})$
$\frac{\|a \cap ra\|'}{\|ra\|'}$	$similarité_{JaroWinkler}(a, q)$[3]	$sim_{ME}(q, ra, sim_{DamereauLevenshtein})$
$\frac{\|a \cap ra\|'}{\|a \cup ra\|'}$	$similarité_{JaroWinkler}(a, ra)$	$sim_{ME}(a, q, sim_{stricte})$[5]
$\frac{2 * \|a \cap ra\|'}{\|a\|' + \|ra\|'}$	$similarité_{JaroWinkler}(q, ra)$	$sim_{ME}(a, ra, sim_{stricte})$
$\frac{\|a \cap ra\|'}{\sqrt{\|a\|' * \|ra\|'}}$	$cosinus_{bi}(a, q)$[6]	$sim_{ME}(q, ra, sim_{stricte})$

Tableau 1 : Description des 42 features

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 ce *run1*, 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.

2.4.2 Run 2

2.4.2.1 Calcul des embeddings de suggestions enseignant.e.s et de réponses étudiant.e.s

Afin de calculer les similarités entre les suggestions d'un.e enseignant.e pour une question donnée et la réponse d'un.e étudiant.e, nous avons utilisé le modèle de langue CamemBERT (Martin *et al.*, 2019), en français, basé sur l'architecture RoBERTa (Liu *et al.,* 2019). Nous avons ainsi encodé

[1] - cosinus calculé sur la chaine de caractères complète découpée en trigrammes
[2] - cosinus calculé sur les occurrences des mots
[3] - similarité calculée sur la chaine de caractères complète
[4] - calculé sur la chaine de caractères complète découpée en mots
[5] - calculé sur la chaine de caractères complète découpée en trigrammes
[6] - cosinus calculé sur la chaine de caractères complète découpée en bigrammes

l'ensemble de *n* mots sous la forme d'un vecteur, obtenant *n* vecteurs de taille 768. Nous obtenons une matrice de taille *n**768. Nous avons ensuite utilisé la librairie Flair afin de calculer nos embeddings de phrases à partir de la matrice. Cette librairie propose trois méthodes différentes pour générer des embeddings de phrases :

- *max* : une fois notre matrice *n**768 obtenue, nous prenons pour chaque colonne, la valeur maximum afin de former l'embedding final de notre document.
- *mean* : une fois notre matrice *n**768 obtenue, nous prenons pour chaque colonne la moyenne de la colonne afin de former l'embedding final de notre document.
- *min* : une fois notre matrice *n**768 obtenue, nous prenons pour chaque colonne la valeur minimum afin de former l'embedding final de notre document.

2.4.2.2 Calcul de distance et similarité avec seuils

Une fois les embeddings obtenus, nous avons mesuré la similarité entre la réponse de chaque étudiant.e avec chacune des suggestions de l'enseignant.e. L'enseignant.e ne donne pas systématiquement plusieurs suggestions : il ou elle peut en donner plusieurs ou se contenter de donner la réponse avec la note 1, qui est toujours donnée. C'est la raison pour laquelle nous avons défini des seuils de similarité pour affecter les notes 0, 0.5 ou 1. Nous n'avons pas cherché à rentrer dans la complexité de prédiction de notes intermédiaires comme 0.2, 0.7 ou 0.8. De manière empirique, nous avons fixé le premier seuil à 0.92. Quand le score de similarité obtenu était inférieur à 0.92 alors la note de la suggestion de l'étudiant.e n'était pas affectée à l'élève. Si aucune suggestion valant 0.5 point n'a été faite par l'enseignant.e, mais qu'il a suggéré une réponse valant 0, alors on affecte à l'élève la note 0. Dans le cas contraire, on affecte la note 0.5. Si l'enseignant.e n'a fait aucune suggestion valant 0 ou 0.5 point, alors on définit un second seuil de 0.905. Le plus grand score de similarité obtenu est supérieur à ce seuil. Nous affections par conséquent la note de 0.5 sinon la note 0.

2.4.2.3 Résultats obtenus en phrase d'entraînement

Une fois les embeddings calculés et les distances obtenues, nous affectons à l'élève la note de la suggestion pour laquelle nous avons obtenu le plus grand score de similarité. Les résultats suivants ont été obtenus sur la phase d'entraînement, par calcul de la précision en fonction des distances et similarité calculés et en fonction des paramètres d'embeddings choisis :

Distances et similarité	**Précision**		
	max	*mean*	*min*
Euclidean distance	*0.570*	*0.549*	*0.580*
Wasserstein distance	*0.537*	*0.538*	*0.548*
Cosinus Similarity	***0.588***	*0.579*	*0.579*
Manhattan distance	*0.417*	*0.412*	*0.417*
Jaccard distance	*0.396*	*0.396*	*0.333*

Le meilleurs score obtenu est celui avec le calcul de similarité cosinus avec le paramètre max pour le calcul de l'embedding. C'est la méthode que nous avons sélectionné pour l'évaluation du run 2 sur le jeu de test.

2.4.3 Run 3

Dans cette partie, nous utilisons Sentence-BERT (Reimers *et al.*, 2019), un réseau de neurones siamois qui a pour objectif de prédire une similarité cosinus entre deux phrases. Dans notre étude, nous avons utilisé le modèle CamemBERT (Martin *et al.*, 2019) en français, basé sur l'architecture RoBERTa (Liu *et al.,* 2019). Pour utiliser cette architecture, nous avons besoin de paires de phrases parallèles et de scores de similarité entre ces phrases.

Dans cette étude, nous avons testé l'utilisation de trois sources de données d'apprentissage :

- La question posée par le ou la professeur.e et la réponse de l'étudiant.e.
- La réponse attendue par le ou la professeur.e et la réponse de l'étudiant.e.
- La concaténation entre la question posée par le ou la professeur.e et la réponse attendue par le ou la professeur.e (ConcatProf.) et la réponse de l'étudiant.e.

Nous avons divisé le jeu de données en trois sous-ensembles : l'ensemble d'apprentissage (60% des paires), l'ensemble de validation (20% des paires) et l'ensemble de test (20% de paires). Les résultats obtenus sont présentés dans la Table 2. Ayant obtenu de meilleurs scores en utilisant uniquement la réponse des professeur.e.s, nous utilisons cette source de données pour notre soumission finale.

Données	**Précision**
Question prof. + Réponse étud.	*0.687*
Réponse prof. + Réponse étud.	***0.742***
ConcatProf. + Réponse étud.	*0.725*

Tableau 2 : Comparaison des résultats obtenus sur l'ensemble d'apprentissage avec Sentence-CamemBERT.

2.5 Résultats obtenus en phase de test sur la tâche 2

Run	**Evaluation**
Run 1 :	**P=0,682**
Run 2 :	P=0,594
Run 3 :	P=0,638
Maximum	**P=0,682**
Médiane	P=0,627
Moyenne	P=0,607
Minimum	P=0,448

Tableau 3 : résultats de la tâche 2

3 Tâche 3 : « Poursuite automatique de l'évaluation de réponses d'étudiants à partir de premières évaluations »

3.1 Présentation

Pour un ensemble composé d'un énoncé et de plusieurs réponses d'étudiant.e.s déjà corrigées par l'enseignant.e, il s'agit d'évaluer les autres réponses d'étudiant.e.s pour cet énoncé (fournir des notes comprises entre 0 et 1).

3.2 Les différents runs

3.2.1 Run 1

Les prétraitements appliqués sont les mêmes que pour le run1 de la tâche 2. Dans cette tâche, un petit nombre de réponses d'étudiant.e.s notées sont connues. La démarche suivie ici va s'appuyer sur ces notes connues pour prédire les notes inconnues. Pour cela, pour une question donnée, on va calculer la similarité ($\boldsymbol{sim_{ME}}$décrite en section 2.3.2 et calculée sur les trigrammes de caractères**)** entre une réponse d'étudiant.e dont on ne connait pas la note et les réponses des étudiant.e.s qui ont été notées. On retiendra la réponse avec laquelle la similarité est la plus élevée et on attribuera la note correspondante.

3.2.2 Runs 2 et 3

Pour cette tâche, nous utilisons Sentence-BERT, présenté en Section 2.4.3 de l'article. Ici, l'objectif est de prédire la note d'un.e étudiant.e en connaissant celles des autres pour cette question. Pour cela, nous construisons notre schéma en utilisant les élèves ayant obtenu une note maximale de 1/1. Nous souhaitons ensuite calculer la similarité des réponses des autres élèves aux réponses complètes. Pour cela, nous avons besoin de beaucoup de données. Pour répondre à ce problème, nous construisons tout d'abord deux tables de données. La première contient les réponses de tous les étudiant.e.s, les notes associées à ces réponses et l'identifiant de la question posée. La deuxième contient les mêmes métadonnées, mais concerne uniquement les réponses ayant obtenu des notes maximales. Nous procédons ensuite à une projection de ces deux tables sur la colonne des identifiants de questions. En d'autres termes, pour chaque question, nous associons toutes les paires possibles *réponse de l'étudiant.e - réponse correcte d'un.e autre étudiant.e.* Nous obtenons ainsi un jeu de données contenant 34 156 paires de réponses. Avec Sentence-BERT, nous essayons ensuite de prédire la similarité cosinus de ces paires de réponses, en utilisant la note obtenue à la question.

En sortie de Sentence-BERT, nous obtenons une similarité cosinus associée à chaque paire de réponses. Ces valeurs doivent ensuite être transformées en note pour cette tâche. Pour cela, nous divisons le corpus en trois notes possibles : 0, 0.5 et 1. Empiriquement, les seuils permettant d'obtenir les meilleurs résultats sur le jeu de données d'entraînement sont : les valeurs de similarités inférieures à 0.33 correspondent à des notes de 0, celles inférieures à 0.66 correspondent à des notes de 0.5 et les autres à des notes de 1 (run 2). Nous proposons une deuxième solution, qui consiste à arrondir les notes à une décimale (run 3).

3.3 Résultats

Run	Evaluation
Run 1 :	**P=0,510 (corrélation = 0,65),**
Run 2 :	P=0,382 (corrélation = 0,56)
Run 3 :	P=0,292 (corrélation = 0,59)
Maximum	**P=0,510 (corrélation = 0,65)**
Médiane	P=0,241
Moyenne	P=0,264
Minimum	P=0,133 (corrélation = 0,60)

Tableau 4 : résultats de la tâche 3

4 Conclusion

L'équipe texte de la R&D EDF a participé pour la 4e année consécutive au Défi Fouille de Texte dans le cadre de la conférence TALN (Traitement Automatique du Langage Naturel). Cette campagne nous a permis de tester plusieurs méthodes de calcul de similarité dont les résultats prometteurs pourront être utilisés directement à EDF Commerce et à d'autres entités du groupe EDF. Nous sommes arrivés 1ers sur la tâche 2 et 1ers sur la tâche 3. Participer à ce défi est pour nous l'occasion d'échanger sur des méthodes de traitement automatique du langage avec des universitaires et des industriels.

Références

DZIKOVSKA, M. O., NIELSEN, R. D., BREW, C., LEACOCK, C., GIAMPICCOLO, D., BENTIVOGLI, L., ... & DANG, H. T. (2013). *Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge*. NORTH TEXAS STATE UNIV DENTON.

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 DEFT. Lille.

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.

LIU, Y., OTT, M., GOYAL, N., DU, J., JOSHI, M., CHEN, D., ... & STOYANOV, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

MARTIN, L., MULLER, B., SUAREZ, P. J. O., DUPONT, Y., ROMARY, L., DE LA CLERGERIE, É. V., ... & SAGOT, B. (2019). Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894*.

REIMERS, N., & GUREVYCH, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.