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

QUEER*@DEFT2021 : Identification du Profil Clinique de patients et Notations Automatique de Copies d'Étudiants

Yoann Dupont[1] Carlos-Emiliano González-Gallardo[1] Gaël Lejeune[1]
Alice Millour[1] Jean-Baptiste Tanguy[1, 2]
(1) Sens Texte Informatique Histoire (STIH), Sorbonne Université
(2) Observatoire des Textes et des Connaissances (OBTIC), Sorbonne Université
{yoann.dupont, carlos-emiliano.gonzalez-gallardo, gael.lejeune,
alice.millour, jean-baptiste.tanguy} @sorbonne-universite.fr

RÉSUMÉ

Nous présentons dans cet article notre contribution aux 3 tâches de la campagne d'évaluation du défi Fouille de Texte 2021. Dans la tâche d'identification de de profil clinique (tâche 1) nous présentons une méthode de recherche d'information basé sur un index dérivé du MeSH. Pour la tâche de notation automatique à partir d'une correction (tâche 2), nous avons expérimenté une méthode de similarité de vecteurs de chaînes de caractères. Pour la tâche de notation à partir de copies déjà notées (tâche 3) nous avons entraîné un réseau de neurones LSTM.

ABSTRACT

QUEER@DEFT2021 : Patients Clinical Profile Identification and Automatic Student Grading

We present in this article our contribution to the 3 tasks of the DEFT 2021 evaluation campaign. For the task of clinical profile identification (task 1) we present an information retrieval method using an index derived from the MeSH. For the automatic grading task from a correction (task 2), we computed similarity on character n-gram vectors. For the scoring task from already graded copies (task 3) we trained an LSTM neural network.

MOTS-CLÉS : MESH, Modèles en caractères, modèle LSTM.

KEYWORDS: MESH, Character-level Models, LSTM model.

1 Introduction

Le Défi Fouille de Textes (DEFT), créé en 2005, porte pour l'édition 2021(Grouin *et al.*, 2021) sur la classification de cas cliniques et la correction automatique de copies d'étudiants à travers trois tâches. Pour un cas clinique donné, la tâche 1 vise à identifier le profil clinique du patient concerné par le type de maladie de toutes les pathologies présentes dans le cas. Elle s'appuie sur les annotations antérieures en informations démographiques et cliniques (pathologies, signes ou symptômes, etc.). Les tâches 2 et 3 concernent la prédiction de notation de réponses d'étudiants à des questions courtes liées au domaine de l'informatique. Les réponses attendues peuvent être de différents types.

— réponse courte correspondant à une question ouverte ;
— réponse exacte correspondant à l'exécution d'un programme court ;
— code XML, HTML.

∗. Question Understanding for Event Extraction and Retrieval

Atelier DÉfi Fouille de Textes (DEFT), pages 95–107.

Remarquons d'entrée de jeu que les tâches 2 et 3 cherchent à développer des modèles capables, pour une réponse donnée, de proposer une note uniquement en observant le contenu intrinsèque de cette réponse. L'intersection des jeux de questions entre l'ensemble d'apprentissage et celui d'évaluation est vide, et l'identité de l'étudiant auteur de la réponse n'est naturellement pas pris en considération. Ces tâches visent ainsi à modéliser la langue (*rationnelle* dans le cas des réponses demandant du code, *naturelle* sinon) en fonction d'un système de notation.

Nous avons décidé de participer au trois tâches où nous explorons des différentes techniques de Recherche d'information, analyses statistiques et réseaux de neurones artificielles. Les jeux de données et métriques d'évaluation étant propres à chaque tâche, nous les présentons successivement.

2 T1 - Identification du profil clinique du patient

La tâche 1 s'articule autour de deux ressources : un corpus de documents annotés au format BRAT [1] (Stenetorp *et al.*, 2012) et un fichier de réponses attendues au format tabulaire. Les annotations disponibles sont celles de DEFT 2020 (Cardon *et al.*, 2020). Le corpus d'entraînement est constitué de 167 documents annotés au format BRAT, celui d'évaluation de 108 documents au même format.

Nous pouvons remarquer plusieurs éléments. Le premier est que la distribution entre les classes est très déséquilibrée, la classe "etatsosy" étant représentée dans environ 90% des documents, là où les classes minoritaires entre 1% et 5% des documents. Nous remarquons également que les distributions des classes dans l'ensemble d'entraînement et d'évaluation sont similaires. La seule différence notable est pour la classe "nerveux", qui passe d'un support relatif de 24,6% dans l'ensemble d'entraînement à 42,6% pour l'ensemble d'évaluation.

2.1 Méthode

D'abord nous avons considéré traiter la tâche comme un problème d'apprentissage automatique, plus précisément, comme un problème de classification automatique multiclasses étant donné l'existence d'un corpus d'entraînement avec 167 cas cliniques et 24 classes différentes. Après avoir effectué quelques expériences exploratoires, nous nous sommes aperçu que la distribution déséquilibrée entre les classes, ajouté au nombre réduit de cas cliniques et la très grande variabilité des justifications pour chaque élément du profil clinique d'un patient produisent un système de classification très silencieux. Finalement nous avons décidé d'aborder cette tâche comme un problème de Recherche d'information (RI) car nous bénéficions de la base de connaissance de référence dans le domaine biomédical. Dans le reste de cette section, nous présenterons les différents modules de notre système et une évaluation interne utilisée pour décider quelles configurations de modules seront sélectionnées pour l'évaluation officielle de DEFT2021.

2.1.1 Index inversé du MeSH

Le Médical Subject Headings (MeSH) est le thésaurus de référence dans le domaine biomédical. Nous avons obtenu le MeSH bilingue anglais-français (fMeSH) produit par l'Institut national de la

1. http://brat.nlplab.org

santé et de la recherche médicale (Inserm) et disponible en format XML. Il comprend 16 catégories thématiques mais pour nos besoins nous avons effectué une étape de filtrage pour garder uniquement les entrées correspondant au chapitre [C] - Maladies. Ce chapitre contient une arborescence de 26 grands groupes de maladies.

Une fois ces entrées identifiées, nous avons obtenu les termes (descripteurs) français associés a chaque entrée et son code correspondant dans l'arborescence. Ainsi dans l'index inversé un descripteur ou terme sera associé à une ou plusieurs maladies. Pour le processus d'indexation nos avons créé trois index inversés (Manning *et al.*, 2008) en suivant différents types de pré-traitements pour chaque descripteur et terme. Pour des raisons pratiques, dans la suite de l'article nous ferons référence aux index inversés simplement comme index.

— *simple* : seuls les caractères alphanumériques, les apostrophes et les traits d'union sont retenus, les espaces sont considérés comme séparateurs pour la tokenisation.
— *spacy_mots* : un token est considéré comme un élément du terme s'il n'est pas un mot vide ou un signe de ponctuation. La tokenisation, filtrage des mots vides et étiquetage morpho-syntaxique pour la détection de signes de ponctuation ont été effectués avec la bibliothèque Python Spacy [2] et le modèle pré-entraîné "fr_core_news_md" [3].
— *spacy_lemmes* : équivalent à *spacy_mots*, mais les lemmes des mots sont stockés dans l'index plutôt que leurs formes.

Le tableau 1 montre la quantité de descripteurs et termes différents dans chacun des index. Nous pouvons observer que le nombre de termes entre *simple* et *spacy_mots* ne varie pas beaucoup, néanmois *spacy_lemmes* présente une réduction de 4,36% par rapport à l'index *simple*.

index	**nombre de termes (descripteurs)**
simple	21 658
spacy_mots	21 384
spacy_lemmes	20 713

TABLE 1 – T1 : Nombre de termes et descripteurs pour chaque index

2.1.2 Détection des phrases négatives

Une des difficultés principales concerne les phrases dites négatives ou hypothétiques. Seules les maladies attestées dans le cas clinique, y compris celles dans le passé, doivent être conservées pour établir le profil du patient. Les maladies mentionnées mais absentes ou hypothétiques ne doivent pas être annotées. Nous avons attesté ce phénomène dans une proportion importante de cas cliniques ce qui a une grande répercussion au niveau des faux positifs. Pour cela nous avons donc implémenté une phase de détection des phrases négatives basée sur les couples mot $\leftrightarrow$ classe grammaticale décrits dans le tableau 2. Si une phrase contient au moins un de ces couples, la phrase est considérée comme négative et elle n'est pas prise en considération.

2. https://spacy.io/
3. https://github.com/explosion/spacy-models/releases//tag/fr_core_news_md-3.0.0

mot	classe grammaticale
ni	conjonction
pas	adverbe
aucun	déterminant
absence	nom
négative	verbe

TABLE 2 – T1 : Couples mot ↔ classe grammaticale

2.1.3 Filtrage des annotations hommes/femmes

Une difficulté supplémentaire de la tâche vient de l'existence de deux classes mutuellement exclusives, à savoir *homme* et *femme*, pour les maladies et complications liées à l'appareil uro-génital (ainsi que les complication dues à la grossesse pour les *femmes*). La particularité de ces annotations vient du fait que la classe demande connaissance extérieure à la justification donnée. Par exemple, une *anurie* sera la justification d'une maladie de l'appareil uro-génital, mais il faudra par ailleurs trouver la justification concernant le sexe du patient. Pour prendre en compte cette exclusion mutuelle, nous avons d'abord annoté *femme* et/ou *homme* en fonction des entrées de l'index trouvées dans le texte, puis nous avons appliqué des filtres en cas d'incohérence. Ces filtres fonctionnent sur le même principe que l'index, mais vont ici servir à supprimer une classe plutôt que d'en rajouter une.

2.1.4 Augmentation de l'index

L'un des problèmes principaux de notre méthode est le relativement faible rappel en comparaison de la précision. Afin de pallier ce problème, nous avons utilisé deux méthodes pour rendre l'index plus couvrant : 1. augmentation avec l'ensemble d'entraînement et 2. ajout manuel d'entrées.

Pour la première méthode, nous avons intégré des entrées correspondant aux justifications présentes dans l'ensemble d'entraînement. Cette méthode est représentée par l'indice *au* dans le tableau 4.

Nous avons également ajouté manuellement de nouvelles entrées à l'index constitué depuis le MeSH afin d'y intégrer des connaissances, aussi bien du domaine que linguistiques. Cette méthode est représentée par l'indice *up* dans la section 2.1.5. Certaines entrées du MeSH sont générales, comme par exemple "abus de drogues" ou "addiction à une substance". Ces éléments, bien que présents dans le MeSH, n'apparaissent pas dans le texte étant donné leur nature générique. Pour le cas de la classe *chimiques*, nous avons ajouté divers noms de drogues, comme par exemple "cannabis", "ecstasy", ou "alcool". Nous avons également ajouté divers termes en rapport avec les drogues concernées, comme par exemple "tabagisme", "alcoolique", "addiction" ou encore "empoisonnement". Pour la classe *endocriniennes*, nous avons rajouté "hormone" ainsi que différents dérivés flexionnels. De même pour les classes *femme* et *homme* où nous avons rajouté des termes spécifiques comme "ovaire", "testicule". Le tableau 3 donne un aperçu de la quantité d'ajouts manuels.

classe	nombre de termes ajoutés
virales	1
homme	13
femme	21
endocriniennes	6
chimiques	33

TABLE 3 – T1 : Nombre de termes ajoutés par classe

2.1.5 Expériences et évaluation interne

Nous avons conduit neuf expériences à partir des différentes configurations des modules de notre système. Pour toutes les configurations, le pré-traitement et la tokenisation des cas cliniques correspondent à celui de l'index utilisé.

- $T1_{bs}$: les requêtes se font sur l'index *simple*
- $T1_{bs\_scy}$: les requêtes se font sur l'index *spacy_mots*
- $T1_{up}$: $T1_{bs}+$ augmentation de l'index en suivant la méthode d'ajout manuel d'entrées
- $T1_{scy\_up}$: $T1_{bs\_scy}+$ augmentation en suivant la méthode d'ajout manuel d'entrées
- $T1_{scy\_up\_neg}$: $T1_{scy\_up}+$ détection des phrases négatives
- $T1_{scy\_up\_neg\_au}$: $T1_{scy\_up\_neg}+$ augmentation en suivant la méthode d'augmentation avec l'ensemble d'entraînement
- $T1_{scy\_up\_neg\_au\_n15}$: $T1_{scy\_up\_neg\_au}+$ n-grams de taille jusqu'à 15
- $T1_{scy\_neg\_au\_n15\_lem}$: les requêtes se font sur l'index *spacy_lemmes* avec détection des phrases négatives, augmentation en suivant la méthode d'augmentation avec l'ensemble d'entraînement et n-grams de taille jusqu'à 15
- $T1_{scy\_up\_neg\_au\_n15\_lem}$: $T1_{scy\_neg\_au\_n15\_lem}+$ augmentation en suivant la méthode d'ajout manuel d'entrées

Le tableau 4 donne un aperçu de la performance de chaque expérience effectuée. Nous pouvons observer que chaque configuration affecte différemment les scores de précision, rappel et F-mesure. L'ajout du filtrage des phrases négatives impacte très positivement la précision, tandis que l'utilisation de lemmes produit les meilleurs résultats en termes de rappel ; néanmoins la précision est fortement affectée négativement. Du point de vue de la F-mesure, utiliser les mots à la place des lemmes mais implémenter le reste des modules, semble être la meilleure option suite à la valeur maximale obtenue.

2.2 Résultats

Dans cette section, nous présenterons les résultats de nos systèmes sur la tâche 1 et analyserons diverses erreurs du systèmes. Les résultats officiels de QUEER sur la tâche 1 sont donnés dans le tableau 5 et le détail par classe est donné dans la figure 1.

Comme nous pouvons le voir dans le tableau 5, la principale source d'erreur de notre système est le silence. Les classes les plus concernées sont "nerveux" (21), "blessures" (14), "digestif" (13), "infections" et "osteomusculaires" (10). Nous nous concentrerons sur ces classes dans cette section.

Pour la classe "nerveux", une partie des silences ont rapport au sommeil, comme "endormissement" ou "somnolente". Ces termes apparaissent bien dans le MeSH, mais dans un contexte différent ou

Système	Précision	Rappel	F-mesure
$T1_{bs}$	0,783	0,673	0,724
$T1_{bs\_scy}$	0,760	0,702	0,729
$T1_{up}$	0,788	0,728	0,757
$T1_{scy\_up}$	0,766	0,753	0,759
$T1_{scy\_up\_neg}$	**0,810**	0,730	0,767
$T1_{scy\_up\_neg\_au}$	0,808	0,763	0,784
$T1_{scy\_up\_neg\_au\_n15}$	0,809	0,765	**0,786**
$T1_{scy\_neg\_au\_n15\_lem}$	0,759	0,739	0,748
$T1_{scy\_up\_neg\_au\_n15\_lem}$	0,762	**0,779**	0,770

TABLE 4 – T1 : Résultats sur la tâche 1 sur une validation croisée à 5 plis

Configuration	Précision	Rappel	F-mesure
$T1_{bs}$	0,819	0,677	0,741
$T1_{scy\_up\_neg}$	**0,843**	0,684	0,755
$T1_{scy\_up\_neg\_au\_n15}$	0,838	**0,734**	**0,782**

TABLE 5 – T1 : Résultats officiels QUEER

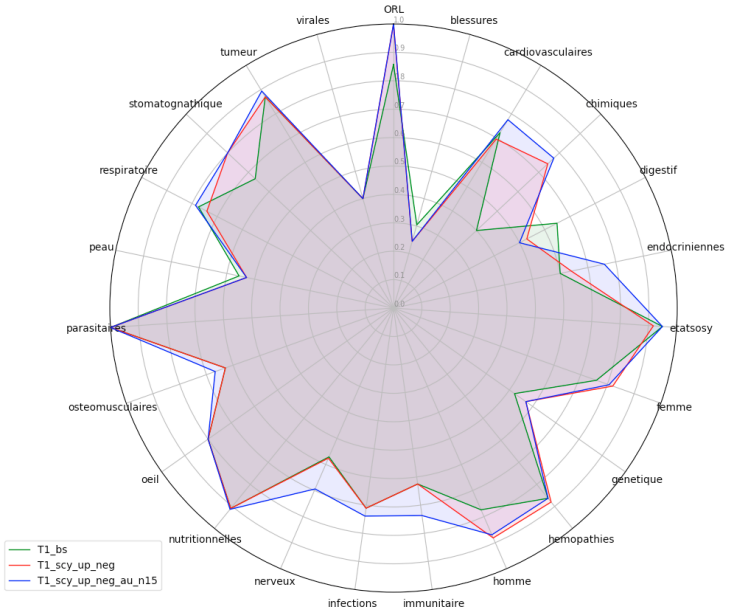

FIGURE 1 – T1 : Résultats officiels QUEER par classe (F-mesure)

une sous forme différente. Par exemple "endormissement" dans l'ensemble d'évaluation apparaît comme "trouble de l'endormissement". "Somnolente" est présent dans le MeSH sous la forme du nom commun "somnolence". Une piste pour améliorer le rappel de notre système pourrait être de nominaliser les formes trouvées dans le texte ("somnolente" deviendrait donc "somnolence"). En ce qui concerne la classe "blessures", le même constat semble se faire : il est détaillé différents types de plaies et autres blessures, cependant absents du MeSH. Pour les termes de la catégorie

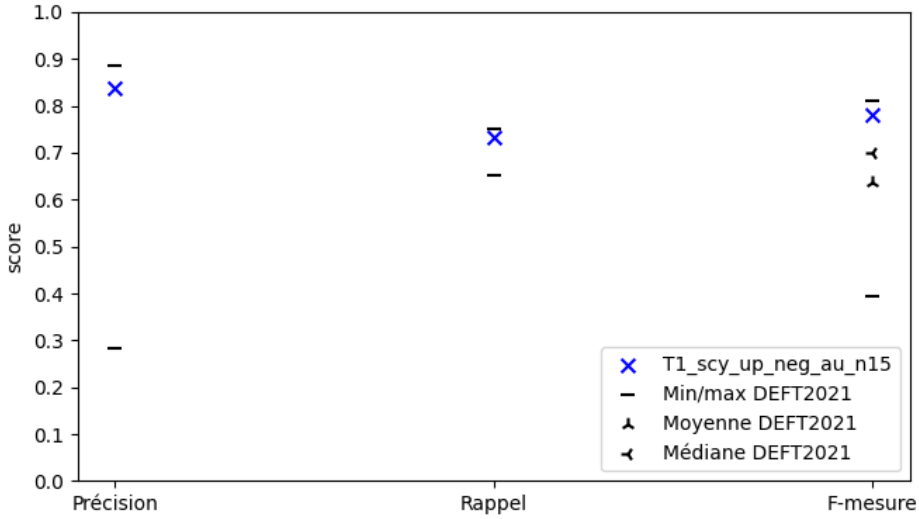

FIGURE 2 – T1 : Résultats QUEER VS DEFT2021

"digestif", le problème de rappel est surtout dû à la généricité des entrées du MeSH comme évoqué dans la section 2.1.4. En effet, de nombreux faux négatifs sont des instances particulières de "tumeur maligne gastrique" ou "tumeur hépatique". Pour la classe "infections", nous avons principalement relevé des noms de bactéries présentes dans le MeSH comme "escherichia coli" ou "klebsiella" qui apparaissaient seules dans le texte mais sous la forme de "infections à <bactérie>" dans le MeSH.

Il semble y avoir principalement deux décalages entre le MeSH et les cas cl²niques qui nuisent au rappel de notre système. Certains termes du MeSH sont plus précis que leur usage, là où d'autres termes du MeSH sont plus généraux que dans l'usage. Pour le premier cas, nous pourrions recourir à la fouille de motifs pour détecter les entrées ayant des constructions similaires pour en extraire des éléments spécifiques (eg : les bactéries des infections recensées dans le MeSH). Pour le second cas, nous pourrions rechercher des lexiques afin d'instancier les termes génériques utilisés dans le MeSH.

La performance de la meilleure configuration de notre système ($T1_{scy\_up\_neg\_au\_n15}$) par rapport au reste de participants à cette tâche est présentée dans la figure 2. Il est intéressant d'analyser le rang de valeurs concernant la précision et rappel. En ce qui concerne la précision, nous pouvons observer un intervalle de valeurs de 0,602 qui est très large si nous le comparons à l'intervalle du rappel qui est de 0,099. Cette différence de rang nous amène à conclure qu'il est beaucoup plus compliqué de diminuer les faux négatifs indépendamment de la méthode utilisée étant donnée un point de départ relativement élevé par rapport à la précision. Il est aussi possible que ce comportement est un résultat de la distribution déséquilibrée des classes. Notre système se positionne très proche des scores maximaux de précision, rappel et F-mesure avec une différence de -0,047 , -0,016 et -0,032 respectivement. En ce qui concerne la F-mesure, notre système est positionné bien au dessus de la moyenne et dans le 3^e quartile.

3 T2 - Évaluation automatique de copies d'après une référence

La tâche 2 s'articule autour de deux ressources : un fichier de questions et un fichier de réponses produites par des étudiants. Outre la question elle-même, le fichier de questions contient, pour certaines d'entre elles, un ou plusieurs éléments de réponse attendus. Ceux-ci s'accompagnent dans certains cas d'indications de notation à l'usage du correcteur. De ce fait, le fichier de questions contient également des éléments de réponse incorrects.

Les données d'entraînement et d'évaluation sont composées de questions distinctes (entraînement : 50, évaluation : 21 questions). En revanche, les réponses (entraînement : 3 820, évaluation : 1 644) ont été fournies par le même ensemble de 118 étudiants.

3.1 Méthode de correction fondée sur des similarités de chaînes de caractères

Nous avons vu la tâche 2 comme un problème consistant à vérifier la proximité entre la correction et la réponse donnée. Nous nous sommes basés sur un calcul de similarité entre la réponse de l'étudiant et une réponse modèle. Nous montrons que cette méthode peut être utilisée avec simplement le corrigé mais qu'elle fonctionne mieux avec des exemples de réponses déjà corrigées.

Nous avons testé différentes manières de vectoriser : en mots, en chaînes de caractères à l'intérieur des mots ou non. Nous avons comparé différentes mesures de similarité afin de comprendre leur influence sur la qualité des résultats, dans l'esprit de ce qui a été fait dans Buscaldi *et al.* (2020). Dans cet article, il était aussi montré que vectoriser des n-grammes de caractères mots ou non-mots permettait d'encoder des relations de l'ordre de la sémantique de manière plus souple qu'avec des mots. Nous y voyons deux raisons principales. D'une part, la vectorisation en n-grammes de caractères rend possible une racinisation non-supervisée : les racines de mots pertinentes sont en effet calculées en fonction des données traitées. D'autre part, avec des n-grammes suffisamment longs, il est possible d'encoder des relations séquentielles entre les mots. À nouveau, cela est fait de manière plus souple qu'avec des mots puisque certains descripteurs extraits sont en réalité des combinaisons "forme + racine". Enfin, les n-grammes de caractères sont plus robustes aux problèmes de variation locale (casse, orthographe…) que les approches en mots. Concernant la tâche elle-même, la méthode appliquée fonctionne comme suit :

1. Une similarité de 0 est attribuée aux non-réponses ;
2. Une similarité de 1 est attribuée aux réponses qui sont strictement contenues dans la correction ;
3. On vectorise le reste des réponses et la correction ;
4. On utilise le score de similarité multiplié par un correctif C avec $C >= 1$;
5. On arrondit au plus proche sachant les notes possibles.

Le coefficient est le produit de deux éléments : le nombre de points maximum que l'on peut obtenir sur la question (qui peut être différent de 1) et un correctif destiné à compenser la tendance du score de similarité à sous-estimer la proximité entre les bonnes réponses et la correction (en particulier sur les questions ouvertes où les vecteurs sont plus creux). Pour les réponses ne rentrant pas dans les cas 1 et 2 décrits ci-dessus on calcule donc la note comme suit (avec $corr$ la correction, rep la réponse de l'étudiant, $coeff$ le nombre de points maximum et C le correctif) : $note = sim(corr, rep) * coeff * C$. Enfin, cette note est arrondie à la valeur la plus proche parmi les notes possibles.

	Unigrammes	**Bigrammes**	**`char_wb(1,9)`** [4]	**`char(1,9)`**
Données brutes	0,607(p=0,044)	0,598(p=0,047)	0,566(p=0,048)	0,62 (p=0,035)
Minuscules	0,604(p=0,05)	0,599(p=0,045)	0,584(p=0,063)	0,632(p=0,034)
Sans balises p	0,604(p=0,05)	0,598(p=0,047)	0,566(p=0,048)	0,623(p=0,034)
Minuscules sans balises p	0,604(p=0,05)	0,598(p=0,188)	0,583(p=0,061)	0,636(p=0,033)

TABLE 6 – T2 : Corrélation de Spearman et *p-value* sur le jeu de données d'entraînement selon les pré-traitements effectués et le type de vectorisation (mesure de similarité : cosinus)

3.2 Paramétrage de la méthode et résultats

Nous présentons dans le tableau 6 l'importance des différents types de caractéristiques utilisées pour la vectorisation (en colonnes) et de pré-traitements (en lignes). Concernant les caractéristiques utilisées, nous comparons les mots (uni-grammes et bi-grammes) et les chaînes de caractères (restreintes à l'intérieur des mots ou libres). La tokenisation en mots est réalisée avec le tokeniseur par défaut de *scikit-learn* (Pedregosa *et al.*, 2011). Nous avons restreint les deux représentations en n-grammes de caractères à l'intervalle $1 \leq N \leq 9$.

La première observation que nous pouvons faire est qu'utiliser les caractères à l'intérieur des mots (`char_wb` dans le tableau) fonctionne moins bien qu'utiliser les mots eux-mêmes. De façon assez inattendue pour nous, les bi-grammes de mots donne de moins bons résultats que les uni-grammes. Enfin, utiliser des chaînes de caractères sans restriction sur les frontières de mots (colonne `char`) donne les meilleurs résultats. C'est en particulier vrai pour les réponses impliquant du code pour lesquelles la représentation en mots est inadaptée (on peut voir un résultat concordant sur de la détection de plagiat dans Brixtel *et al.* (2009)). Ces observations se reflètent à la fois sur la corrélation de Spearman et sur la *p-value*.

Concernant les pré-traitements, nous en avons testé deux très simples : passage en minuscules et suppression des balises "p" entourant la correction et les réponses. On n'observe pas d'influence significative sur les résultats des représentations en mots. L'apport est par contre plus significatif sur les approches en caractères. Le passage en minuscules est nettement plus influent et la combinaison avec le débalisage offre un certain gain pour l'approche en n-grammes de caractères.

Nous avons testé différentes mesures de similarité. Il apparaît que l'indice de Jaccard et la distance de Bray-Curtis donnaient les moins bons résultats. Le coefficient de Dice donnait les meilleurs résultats, le cosinus ayant tendance à sous-estimer les notes. En corrigeant la similarité par le coefficient C nous obtenions alors les meilleurs résultats avec la distance cosinus, $C = 3$ étant la valeur entière avec laquelle nous avions les meilleurs résultats.

4 T3 - Poursuite automatique de correction à partir de premières corrections

La tâche 3 s'articule autour de deux ressources : un fichier de questions et un fichier de réponses. Aucun élément de réponse n'est fourni. Néanmoins, au moins une des réponses produites par les

4. Nous utilisons ici la terminologie du COUNTVECTORIZER de *scikit-learn* : *characters between word boundaries*

étudiants a obtenu la note maximale.

Les données d'entraînement et d'évaluation sont composées de questions distinctes (entraînement : 11, évaluation : 6 questions). Les réponses du jeu de données d'évaluation (387 réponses) ont été produites par un sous-ensemble de 197 étudiants parmi les 213 ayant produit les réponses du jeu d'entraînement (769 réponses).

4.1 Méthodes

Trois méthodes ont été explorées pour la poursuite automatique de correction à partir de premières corrections : (M1) une méthode par pondération TF-IDF (Aizawa, 2003; Ramos *et al.*, 2003), (M2) une méthode par régression linéaire (Aggarwal & Zhai, 2012) et (M3) une méthode utilisant un réseau de neurones artificiels LSTM (Zhou *et al.*, 2015; Xiao *et al.*, 2018).

Les méthodes de pondération TF-IDF et de régression linéaire ayant été entraînées à prédire une note pour une question donnée – et l'ensemble de test ne comprenant pas les mêmes question que l'ensemble d'apprentissage –, elles n'ont pas été présentées au challenge. Néanmoins, nous les décrivons *infra* ainsi que leurs résultats sur l'ensemble d'apprentissage.

4.1.1 (M1) Méthode de pondération TF-IDF

Cette méthode fait intervenir la méthode de pondération TF-IDF [5]. Pour chaque question de l'ensemble d'apprentissage, les valeurs des TF-IDF de toutes les réponses entre elles sont calculées et stockées dans une matrice carrée. Le calcul de cette matrice a été réalisé en utilisant la classe *TfidfVectorizer* de la librairie *scikit-learn* (version 0.23.2) (Pedregosa *et al.*, 2011). Ensuite, pour chaque réponse de l'ensemble d'apprentissage, une note peut lui être prédite : celle de la réponse qui maximise le TF-IDF – il s'agit de la note de la réponse qui a un TF-IDF avec la réponse courante supérieur à tous les autres TF-IDF de la question courante.

4.1.2 (M2) Régression linéaire

Cette méthode apprend, pour chaque question, un modèle de régression linéaire. Afin d'apprendre à prédire les notes, les réponses sont vectorisées comme suit :

- stylométrie : longueur de la réponse ; nombre de caractères numériques ; nombre de caractères non numériques ; nombre de caractères d'espacement ; nombre de caractères n'étant pas d'espacement ; nombre de caractères de mots ; nombre de caractères n'étant pas de mots ; nombre de caractère en capital ;
- stemmes (*PorterStemmer* de la librairie *nltk* (Bird, 2006)) : chaque dimension du vecteur correspond à un stemme de l'ensemble des réponses de l'ensemble d'apprentissage.

Une fois les réponses vectorisées, l'apprentissage est réalisé en utilisant la classe *LinearRegression* de la librairie *scikit-learn*.

5. *term frequency-inverse document frequency*

4.1.3 (M3) Réseau de neurones LSTM

La troisième méthode, la seule présentée, est un réseau de neurones constitué avec la libraire Python *keras* (version 2.4.3) (Brownlee, 2016). À la différence des deux premières, cette méthode n'est pas spécialisée par question. Un unique réseau est constitué pour toutes les questions et réponses. Il y a donc une prétention à modéliser la valeur (notative) d'une réponse, quelle que soit la question. Théoriquement, cela pose problème : une réponse peut être correcte pour une question et incorrecte pour une autre. Une voie d'amélioration serait l'intégration de l'énoncé de la question à la modélisation.

Pré-traitements des données Les réponses sont mises en bas de casse et les caractères non alphanumériques remplacés par des espaces. La segmentation est opérée par le *Tokenizer* de *keras*.

Structure du réseau Le réseau est constitué d'une première couche *Embedding* qui vectorise les réponses (nombre de dimension des vecteurs : 100). À cette première couche succèdent les couches profondes du réseau, spécifiques à chaque *run* présenté :

run 1 : Une couche *LSTM (120, dropout = 0,2)*, suivie d'une couche *Dense(120, activation = "relu")*, suivie d'une couche *Dense(21, activation = "linear")*, suivie d'une couche *Dense(21, activation = "softmax")*. Apprentissage réalisé sur 10 époques.

run 2 : Une couche *LSTM (100, dropout = 0,2)*, suivie d'une couche *Dense(100, activation = "relu")*, suivie d'une couche *Dense(100, activation = "linear")*, suivie d'une couche *Dense(21, activation = "softmax")*. Apprentissage réalisé sur 10 époques.

run 3 : Une couche *LSTM (120, dropout = 0,2)*, suivie d'une couche *Dense(120, activation = "relu")*, suivie d'une couche *Dense(21, activation = "linear")*, suivie d'une couche *Dense(21, activation = "softmax")*. Apprentissage réalisé sur 100 époques.

4.2 Résultats

4.2.1 M1 et M2 : méthodes non présentées

Questions	M1 Précision	M1 Rappel	M1 F-mesure	M2 r2
5002	0,200	0,104	0,078	0,022
5003	0,151	0,389	0,218	0,156
5004	0,111	0,244	0,152	0,090
5006	0,581	0,344	0,357	0,479
5007	0,395	0,589	0,453	-0,719
5008	0,439	0,267	0,191	0,715
5010	0,603	0,189	0,214	-0,137
5013	0,370	0,444	0,354	-0,943
5014	0,102	0,193	0,111	-0,186
2011	0,800	0,845	0,781	###[6]
2013	0,584	0,319	0,205	-0,276

TABLE 7 – T3 : Résultats, sur l'ensemble d'apprentissage, pour les méthodes M1 (Précision, Rappel et F-mesure) et M2 (coefficient de détermination r2), non présentées

Le tableau 7 montre pour la méthode M1 des résultats peu satisfaisants, dans la mesure où ils peinent à dépasser les résultats obtenus sur l'ensemble de test par la meilleure équipe ($F - mesure = 0,510$). On observe aussi des disparités entre les questions, la question numéro 5002 obtenant des résultats très proches de 0.

Les résultats de cette première méthode *baseline* ont conforté l'idée de procéder à l'apprentissage de modèle de notation spécifique à chaque question, lesquelles sont substantiellement différentes les unes des autres (comme proposer du code HTML *vs.* proposer une réponse en langage naturel). Le tableau 7 montre aussi des résultats mitigés pour la méthode M2. La question 5006 est corrigée relativement correctement quand les questions à $r2 < 0$ ne le sont pas du tout.

4.2.2 M3 : méthode présentée

Questions	**run 1**	**run 2**	**run 3**
2012	0,413	0,375	0,356
5001	0,113	0,038	0,094
5005	0,266	0,172	0,188
5009	0,532	0,532	0,362
5011	0,149	0,170	0,085
5012	0,000	0,000	0,000
Moyenne	0,278	0,241	0,212

TABLE 8 – T3 : Précision des 3 runs sur l'ensemble de test

Le tableau 8 montre que les trois *runs* offrent des performances similaires, avec une supériorité pour le *run 1*. Effectivement, mis à part l'agencement des couches profondes et le nombre d'époques, ils sont égaux. On comprend donc bien que l'agencement des couches profondes et le nombre d'époques n'ont pas un impact majeur sur les performances du système. En outre, on observe aussi que la question 5012 est toujours mal notée et que la question 5009 est toujours la mieux notée.

5 Conclusion

Dans cet article, nous avons présenté différentes approches pour les trois tâches du Défi Fouille de Textes 2021. En ce qui concerne la tâche 1 nous avons décidé de créer un système inspiré de la recherche d'information pour identifier le profil d'un patient à partir de son cas clinique. Notre système s'est montré très performant avec une F-mesure de 0,782 ; bien au-dessus de la moyenne des équipes. Néanmoins le taux de faux négatifs est un élément à travailler pour diminuer le silence du système avec des analyses plus précises du contenu des cas cliniques. Dans la tâche 2 (notation automatique à partir d'une correction) nous avons proposé une méthode fondée sur une similarité de distribution de chaînes de caractères (mots ou non-mots). Cette méthode a obtenu une F-mesure de 0,63 ce qui la situe juste au-dessus de la médiane des participants au défi. Nous envisageons d'appliquer cette méthode à la tâche 3. Cette dernière a été abordée en utilisant un réseau de neurones LSTM et a obtenu une F-mesure de $0,278$ qui la situe au-dessus de la moyenne et de la médiane.

6. Les trois dièses ### signifient que la valeur est très grande ($< 10^{24}$) mais négative.

Références

AGGARWAL C. C. & ZHAI C. (2012). A survey of text classification algorithms. In *Mining text data*, p. 163–222. Springer.

AIZAWA A. (2003). An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, **39**(1), 45–65.

BIRD S. (2006). NLTK : the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, p. 69–72.

BRIXTEL R., LESNER B., BAGAN G. & BAZIN C. (2009). De la mesure de similarité de codes sources vers la détection de plagiat : le "Pomp-O-Mètre". In *Actes des 7èmes Journées MajecSTIC'09*, p. 8 p., Avignon, France. Actes électroniques, HAL : hal-01066127.

BROWNLEE J. (2016). *Deep learning with Python : develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery.

BUSCALDI D., FELHI G., GHOUL D., LE ROUX J., LEJEUNE G. & ZHANG X. (2020). Calcul de similarité entre phrases : quelles mesures et quels descripteurs ? In R. CARDON, N. GRABAR, C. GROUIN & T. HAMON, Éds., *DEFT@JEP/TALN/RECITAL 2020*, p. 14–25, Nancy, France : ATALA. HAL : hal-02784738.

CARDON R., GRABAR N., GROUIN C. & HAMON T. (2020). Présentation de la campagne d'évaluation DEFT 2020 : similarité textuelle en domaine ouvert et extraction d'information précise dans des cas cliniques (presentation of the deft 2020 challenge : open domain textual similarity and precise information extraction from clinical cases). In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Atelier DÉfi Fouille de Textes*, p. 1–13.

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 *Conférence sur le Traitement Automatique des Langues Naturelles (TALN, 28e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 23e édition). Atelier DÉfi Fouille de Textes*.

MANNING C. D., SCHÜTZE H. & RAGHAVAN P. (2008). *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

PEDREGOSA F., VAROQUAUX G., GRAMFORT A., MICHEL V., THIRION B., GRISEL O., BLONDEL M., PRETTENHOFER P., WEISS R., DUBOURG V. *et al.* (2011). Scikit-learn : Machine learning in python. *the Journal of machine Learning research*, **12**, 2825–2830.

RAMOS J. *et al.* (2003). Using TF-IDF to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, p. 29–48 : Citeseer.

STENETORP P., PYYSALO S., TOPIĆ G., OHTA T., ANANIADOU S. & TSUJII J. (2012). BRAT : a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, p. 102–107.

XIAO L., WANG G. & ZUO Y. (2018). Research on patent text classification based on word2vec and LSTM. In *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, volume 1, p. 71–74 : IEEE.

ZHOU C., SUN C., LIU Z. & LAU F. (2015). A C-LSTM neural network for text classification. *arXiv preprint arXiv :1511.08630*.