Participation de l'équipe du LIMICS à DEFT 2020

Perceval Wajsbürt, Yoann Taillé, Guillaume Lainé, Xavier Tannier

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)
Nancy, France, 08-19 juin 2020

Participation de l'équipe du LIMICS à DEFT 2020

Perceval Wajsbürt[1], Yoann Taillé[1,2], Guillaume Lainé[1], Xavier Tannier[1]
(1) Sorbonne Université, Inserm, LIMICS, Paris, France
(2) Institut des Sciences du Calcul et des Données, Sorbonne Université, Paris, France

RÉSUMÉ

Nous présentons dans cet article les méthodes conçues et les résultats obtenus lors de notre participation à la tâche 3 de la campagne d'évaluation DEFT 2020, consistant en la reconnaissance d'entités nommées du domaine médical. Nous proposons deux modèles différents permettant de prendre en compte les entités imbriquées, qui représentent une des difficultés du jeu de données proposées, et présentons les résultats obtenus. Notre meilleur run obtient la meilleure performance parmi les participants, sur l'une des deux sous-tâches du défi.

ABSTRACT

Participation of team LIMICS in the DEFT 2020 challenge

In this article, we present the methods developed and the results obtained during our participation in task 3 of the DEFT 2020 evaluation campaign, consisting of the recognition of named entities in the medical field. We propose two different models allowing to take into account the nested entities, which represent one of the difficulties of the proposed data set, and present the results obtained. Our best run obtains the best performance among the participants, on one of the two subtasks of the challenge.

MOTS-CLÉS : Reconnaissance d'entités nommées, apprentissage profond, entités imbriquées.

KEYWORDS: Named entity recognition, deep learning, nested entities.

1 Introduction

Nous présentons dans cet article les méthodes que nous avons conçues pour répondre à la tâche 3 de la campagne d'évaluation DEFT 2020. Cette tâche porte sur la détection d'entités nommées dans des textes de descriptions de cas cliniques (Grabar *et al.*, 2018). Les différents types d'entités sont, d'une part, les *pathologies* et *signes ou symptômes* (tâche 3.1), d'autre part, l'*anatomie*, les *examens*, *substances*, *doses*, *modes d'administration*, *traitements* (chirurgical ou médical), *valeurs*, *moment*.

Le corpus se compose de 100 fichiers pour l'entraînement (8350 annotations) et 67 fichiers pour le test (3800 annotations).

Le jeu d'entraînement fourni par les organisateurs est composé de 100 documents (8350 annotations), tandis que le jeu de test est composé de 67 documents (3800 annotations). Plus de détails sur le défi sont présents dans Cardon *et al.* (2020).

Nous décrivons dans cet articles nos 3 soumissions officielles, dont l'une a obtenu la meilleure performance sur la tâche 3.2.

Atelier DÉfi Fouille de Textes, pages 108–117.
HAL : hal-02784747.

2 Méthode

Nous présentons deux approches visant à traiter le problème des mentions imbriquées, qui sont la particularité principale du jeu de données DEFT et qui sont le sujet d'un regain d'attention dans la communauté TAL depuis 2018. La première difficulté rencontrée est qu'il n'est pas aisé voire impossible, selon la nature des chevauchements, d'encoder sans perte toutes les mentions en une unique séquence de tags.

2.1 CamemBERT + Layered BiLSTM CRF

Ce modèle tente de tirer partie de la combinaison de méthodes actuellement performantes en traitement automatique des langues et en reconnaissance d'entités nommées imbriquées.

Il comprend d'abord une composante CamemBERT (Martin *et al.*, 2020), un modèle BERT pré-entraîné sur des données françaises, afin d'obtenir une représentation des mots de notre corpus qui prenne en compte leur contexte. Une couche dense est ajoutée après cette composante afin de limiter la taille des représentations et d'en faire correspondre les dimensions avec celles des couches suivantes.

Vient ensuite une succession de combinaisons de LSTM bidirectionnels et de CRF, utilisées dans le modèle de Lample *et al.* (2016). Ce modèle stratifié s'inspire de Ju *et al.* (2018). Chacune de ses strates est capable de prédire des entités à partir de séquences de représentations de mots. Si une strate prédit une entité, les représentations de la zone correspondant à cette prédiction sont fusionnées pour obtenir une représentation de l'entité, qui est transmise à la strate suivante. L'enchaînement de strates s'arrête quand aucune entité n'est détectée par la dernière strate, ou quand un nombre défini de strates est atteint.

Comme le montre la Figure 1, les strates détectent d'abord les entités les plus courtes puis les entités plus longues les incluant. La fusion des représentations quand une entité est détectée se fait en moyennant les représentations comprises dans la zone prédite. Les paramètres du LSTM de chaque strate sont partagés entre toutes les strates.

2.1.1 Apprentissage

Une première expérience à été effectuée en entraînant ce modèle de manière multiclasse, mais elle s'est révélée moins performante que plusieurs modèles binaires. Pour chacune des 10 classes, un modèle CamemBERT + Layered BiLSTM-CRF a donc été entraîné. En inspectant le corpus, nous avons fixé le nombre maximal de couches par modèle à 2, correspondant à la profondeur maximale d'imbrication d'entités nommées du même type.

2.1.2 Paramètres du modèle

La composante CamemBERT est initialisée avec les poids de CamemBERT-base, pré-entraîné sur OSCAR (Ortiz Suárez *et al.*, 2019). Seules ses 4 dernières couches sont ré-entraînées sur nos données. Les paramètres des couches suivantes sont initialisés aléatoirement selon la méthode de (Glorot & Bengio, 2010). La couche dense et les LSTM bidirectionnels ont une dimension de 200 unités, et les

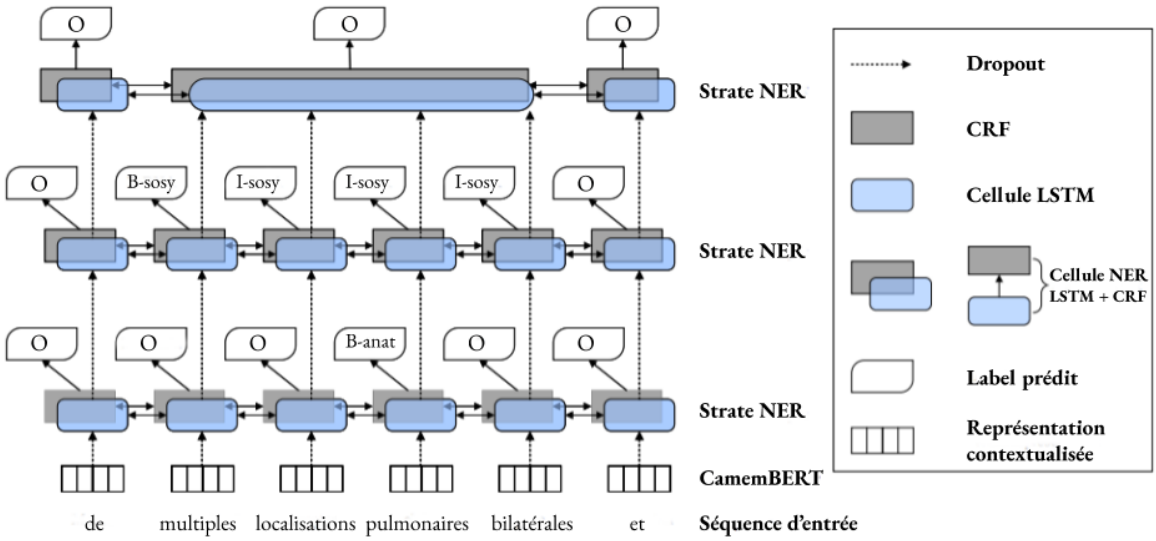

FIGURE 1 – Schéma adapté de Ju *et al.* (2018) de la structure du modèle CamemBERT + Layered BiLSTM-CRF. Dans l'exemple, "pulmonaires" est imbriquée dans "multiples localisations pulmonaires bilatérales".

Classe	sosy	moment	pathologie	mode	substance	examen	anatomie	dose	valeur	traitement
Nombre d'époques	53	42	64	41	49	41	46	45	40	82

TABLE 1 – Nombre d'époques retenu pour l'entraînement de chaque modèle binaire par classe

LSTM comportent deux couches cachées. Du dropout (Srivastava *et al.*, 2014) est appliqué avec une probabilité 0.3 dans les LSTM.

Chaque modèle binaire est optimisé par backpropagation avec Adam (Kingma & Ba, 2015) sans weight decay, avec un pas d'apprentissage de $5 \times 10^{-}5$ pour la composante CamemBERT et $5 \times 10^{-}3$ pour le reste. Le nombre d'époques d'apprentissage pour chaque modèle a été déterminé de la manière suivante : si on n'observe pas d'amélioration du coût sur les données d'apprentissage pendant 5 époques, on arrête l'entraînement. Le nombre d'itérations retenu par classe est présenté dans le tableau 1. Sur une carte graphique NVIDIA Quadro K5200, l'apprentissage sur 100 documents pour toutes les classes prend environ 25 minutes.

2.2 Iterative Greedy NER

2.2.1 Modèle

Nous détaillons ici un nouveau modèle ainsi qu'une procédure permettant d'en apprendre les paramètres. Le modèle est un Transformer (Vaswani *et al.*, 2017) prenant en entrée une séquence de mots ainsi qu'une liste de mentions déjà extraites (liste vide à la première itération), et prédit en sortie une liste de nouvelles mentions. Les mentions prédites à chaque itération ne se chevauchent pas, en revanche l'ensemble des mentions prédites à la fin des itérations peuvent se chevaucher.

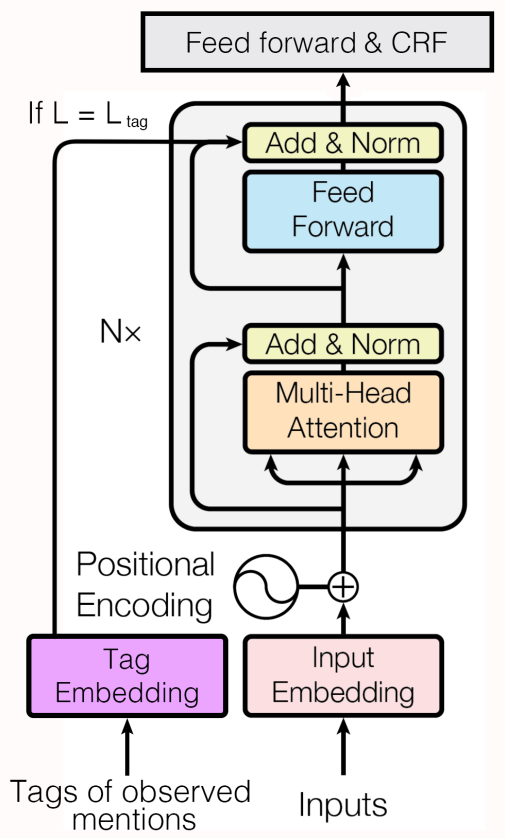

FIGURE 2 – Notre modèle basé sur un Transformer modifié pour intégrer à la fin de la couche L_{tag} les tags des mentions précédemment observées — schéma adapté de Vaswani *et al.* (2017)

Afin d'être manipulables par un réseau de neurones, les mots sont transformés en embeddings. De même, les mentions sont manipulées sous forme de tags au format BIO, chaque tag étant converti en embeddings pour être sommé avec les autres tags à la même position.

Nous calculons les probabilités des séquences de tags en chaînant un transformer CamemBERT (Martin *et al.*, 2020) avec un CRF linéaire (Lafferty *et al.*, 2001). Tandis que les embeddings de tokens sont intégrés par le transformer en amont du modèle, nous intégrons les embeddings de tags à la sortie d'une des couches du modèle (Figures 2, 3).

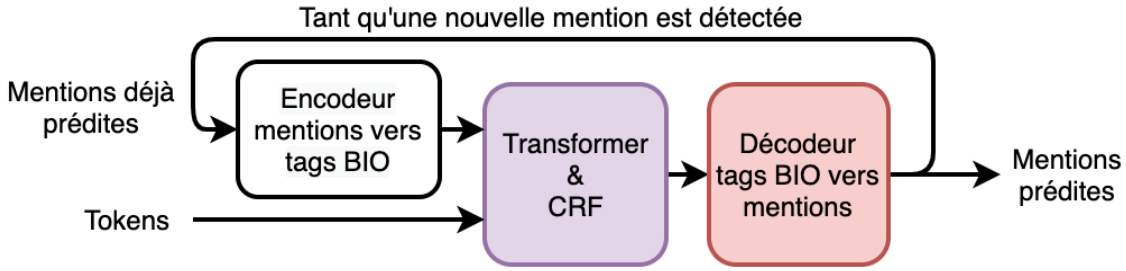

FIGURE 3 – Schéma de prédiction du modèle

2.2.2 Inférence

Pour chaque phrase du corpus à annoter, notre modèle commence par prédire la séquence de mentions la plus probable à partir de la seule séquence de tokens en entrée, puisqu'aucune mention n'a déjà été prédite. Puis, tant qu'une ou plusieurs mentions sont détectées, on les ajoute à la liste des mentions observées, et on ré-itère la prédiction.

2.2.3 Apprentissage

Pendant l'entraînement, on cherche à maximiser les probabilités assignées par le modèle aux listes de mentions à retrouver, avec toujours la difficulté de traiter les mentions chevauchantes.

Comme dans le modèle précédent, on choisit de procéder en plusieurs étapes, et de ne faire prédire par le modèle à chaque exécution que des mentions non imbriquées. Il existe cependant plusieurs combinaisons de mentions de ce type, sans qu'on puisse dire a priori si l'une est meilleure que l'autre. Autrement dit, il existe plusieurs chemins de prédictions possibles menant à une même liste de mentions. Pour deux mentions imbriquées

(annotations que l'on nomme T et H), on peut prédire T en premier, puis H sachant T, ou le contraire, c'est-à-dire choisir d'optimiser entre deux objectifs :

— $P(T, H) = P(T, H|T) \times P(T)$
— $P(T, H) = P(T, H|H) \times P(H)$

Dans cette situation symétrique, une solution qui consisterait à optimiser les deux chemins en même temps, par exemple en sommant plusieurs coûts (*loss*), pourrait conduire à une situation non optimale dans laquelle le modèle est indirectement incité à détecter une combinaison incorrecte des deux mentions, par exemple ces deux mentions non imbriquées mais sous-optimales :

Des modèles comme celui de Ju *et al.* (2018) choisissent une stratégie à l'avance (mentions plus petites d'abord, par exemple), mais le risque est de ne pas profiter de toutes les interdépendances qui font que certaines mentions sont plus simples à trouver quand on connaît les autres. Une autre solution consiste à choisir l'ordre d'extraction menant à la meilleure performance par le modèle, mesurée en F-mesure. On applique ainsi une stratégie gloutonne qui sélectionne, parmi les combinaisons sans chevauchement de mentions non-observées dans ce batch, la plus proche des mentions prédites en terme de recouvrement. Intuitivement, cela signifie qu'on privilégie une combinaison plus facile à prédire pour le modèle.

En pratique, pour choisir à chaque itération cette meilleure combinaison, on définit un graphe de recouvrement $\mathcal{O}$ dans lequel un lien entre deux mentions indique qu'elles se chevauchent. De ce graphe, on extrait les composantes connexes C_i, qui peuvent être vues comme les zones du texte dans lesquelles on observe des recouvrements. On choisit dans chaque composante C_i la mention la plus

proche d'une des mentions prédites, en terme de similarité F1 des intervalles (c'est-à-dire, des tags BIO des tokens qui les composent), puis on réitère en supprimant ses mentions voisines de la liste des candidats.

Enfin, pendant l'entraînement, afin de simuler une extraction en cours, on sélectionne au hasard dans chaque phrase une partie des mentions que l'on définira comme étant déjà extraites par le modèle.

Cette procédure est reprise sous forme de pseudo-code à l'algorithme 1.

Algorithm 1 Algorithme d'apprentissage du modèle

$\mathcal{O}$, le graphe de recouvrement des mentions du corpus
$\mathcal{C}$, les composantes connexes de $\mathcal{O}$
M^{obs}, les mentions observées pour chaque batch
M_i, les mentions de la phrase i
X_i, les tokens de la phrase i
$M^{\text{predicted}}$, les mentions prédites pour chaque batch
M^{target}, les mentions utilisées pour calculer la loss
f, le modèle
for t = 1, . . ., T **do**
    on échantillonne un mini-batch $I_t \subset \{1, \ldots, n\}$ de taille b
    $M^{\text{obs}} \leftarrow \{\}$
    for phrase $i \in I_t$ **do**
        on échantillonne un sous-ensemble P des mentions de la phrase i
        $M^{\text{obs}} \leftarrow M^{\text{obs}} \bigcup P$
    end for
    $M^{\text{predicted}} \leftarrow \arg\max_M P(M|X_i, M^{\text{obs}})$
    $M^{\text{target}} \leftarrow \{\}$
    for phrase $i \in I_t$ **do**
        for composante $j \in$ phrase i **do**
            $m^{closest} \in C_j \setminus M^{obs}$ la mention la plus similaire à une des mentions de $M^{\text{predicted}}$
            $M^{\text{target}} \leftarrow M^{\text{target}} \bigcup \{m^{\text{closest}}\}$
        end for
        $l \leftarrow -\log f(M^{\text{target}}|X_i, M^{\text{obs}})$
        backprop(l)
    end for
end for

2.2.4 Paramètres du modèle

Modèle de base Le Transformer est initialisé avec les poids de CamemBERT-large, pré-entrainé sur CCNET (Wenzek *et al.*, 2019). Les paramètres restants sont initialisés aléatoirement selon la méthode de (Glorot & Bengio, 2010). Du dropout (Srivastava *et al.*, 2014) est appliqué avec une probabilité 0.2 dans le Transformer et 0.4 dans les couches qui le suivent. Les 19 premières couches du transformer sont figées à leur valeur initiale. On optimise les paramètres par backpropagation sur 60 epochs avec Adam (Kingma & Ba, 2015) sans weight decay. Deux pas d'apprentissage sont utilisés : un pour la partie transformer, initialisé à 5×10^{-5}, et un pour le reste du modèle, initalisé à 1×10^{-2}. On divise les pas d'apprentissage par 4 à l'epoch 25, 16 à l'epoch 40 et 64 à l'epoch 50.

		Précision	Rappel	F1
LIMICS run 1	pathologie	0,2644	0,2771	0,2706
	sosy	0,4539	0,3081	0,3670
	Total	0,4223	0,3045	0,3538
LIMICS run 2	pathologie	0,4280	0,6446	0,5144
	sosy	0,6491	0,5668	0,6052
	Total	0,6086	0,5758	0,5917
LIMICS run 3	pathologie	0,5122	0,6325	0,5660
	sosy	0,6885	0,5668	0,6218
	Total	0,6598	0,5744	0,6141
Meilleur DEFT	**Total**			**0,6603**
Médiane DEFT	**Total**			0,4557

TABLE 2 – Résultat de la tâche 3.1 (*pathologie* et *signe ou symptôme*)

Les embeddings de tags sont introduits dans le transformer à la couche $L_{tag} = 18$. Sur une carte graphique GeForce GTX 1080, l'apprentissage sur 100 documents prend environ 20 minutes.

Vote mono-classifieur. Ayant constaté les fortes variations de performance entre différents modèles identiques entraînés avec des amorces aléatoires différentes (variations dans chaque classe mais moyennes finales similaires), nous avons également soumis un dernier run résultant de trois entraînements du modèle Iterative Greedy NER, avec des amorces différentes, puis un vote majoritaire.

3 Résultats et discussion

Les résultats de nos systèmes sont présentés dans les Tables 2 pour la tâche 3.1 et 3 pour la tâche 3.2. Le run 1 correspond au modèle Layered BiLSTM, les run 2 et 3 correspondent au modèle Iterative Greedy NER, le dernier étant la version ensembliste.

Le modèle Iterative Greedy NER obtient de meilleurs résultats que le modèle Layered BiLSTM, sur toutes les classes. Sur la tâche 3.1, il obtient un score F1 de 0.59 (avec délimitation exacte des mentions), et sa version ensembliste un score de 0.61. Sur la tâche 3.2, il obtient le score 0.756 et sa version ensembliste 0.763, meilleur résultat parmi les participants de la compétition.

Notons que pour certaines classes rares mais présentant de faibles variations linguistiques, hybrider notre système avec des règles serait probablement bénéfique.

4 Conclusion

Notre participation au défi fouille de texte de cette année s'est soldée par de bons résultats sur la tâche 3. Une analyse plus détaillée devra nous permettre de comprendre, notamment, pourquoi les types d'entités de la tâche 3.1 se sont moins bien prêtés à nos deux modèles, avec une F-mesure 5 points sous celle du meilleur participant, alors que nous obtenons les meilleurs résultats sur la tâche 3.2. De plus amples expérimentations seront également nécessaires pour estimer l'apport de

		Précision	Rappel	F1
LIMICS run 1	anatomie	0,6819	0,7074	0,6944
	dose	0,3611	0,2500	0,2955
	examen	0,6819	0,5667	0,6190
	mode	0,6458	0,3483	0,4526
	moment	0,6574	0,4303	0,5201
	substance	0,5134	0,3674	0,4283
	traitement	0,4798	0,3520	0,4061
	valeur	0,7746	0,6366	0,6989
	Total	0,6587	0,5673	0,6096
LIMICS run 2	anatomie	0,7674	0,7482	0,7577
	dose	0,6047	0,5000	0,5474
	examen	0,7947	0,8103	0,8024
	mode	0,6471	0,4944	0,5605
	moment	0,6629	0,7152	0,6880
	substance	0,8092	0,7316	0,7685
	traitement	0,6471	0,6513	0,6492
	valeur	0,8224	0,8148	0,8186
	Total	0,7635	0,7494	0,7564
LIMICS run 3	anatomie	0,8056	0,7250	0,7632
	dose	0,6486	0,4615	0,5393
	examen	0,8130	0,7980	0,8054
	mode	0,7455	0,4607	0,5694
	moment	0,6923	0,7091	0,7006
	substance	0,8375	0,7412	0,7864
	traitement	0,6960	0,6250	0,6586
	valeur	0,8313	0,7986	0,8146
	Total	0,7948	0,7330	**0,7626**
Meilleur DEFT	**Total**			**0,7626**
Médiane DEFT	**Total**			0,6151

TABLE 3 – Résultat de la tâche 3.2 (autres classes)

l'idée principale du modèle Iterative Greedy NER, c'est-à-dire la liberté laissée au modèle de trouver les meilleures combinaisons de mentions, indépendamment de leur niveau d'imbrication.

Remerciements

Nous remercions les organisateurs pour la création du corpus ainsi que pour l'organisation du défi.

Références

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. In *Actes de DEFT*.

GLOROT X. & BENGIO Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, p. 249–256.

GRABAR N., CLAVEAU V. & DALLOUX C. (2018). CAS : French corpus with clinical cases. In *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*, p. 122–128, Brussels, Belgium : Association for Computational Linguistics. DOI : 10.18653/v1/W18-5614.

JU M., MIWA M. & ANANIADOU S. (2018). A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, p. 1446–1459, New Orleans, Louisiana : Association for Computational Linguistics. DOI : 10.18653/v1/N18-1131.

KINGMA D. P. & BA J. L. (2015). Adam : A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

LAFFERTY J., MCCALLUM A. & PEREIRA F. C. N. (2001). Conditional random fields : Probabilistic models for segmenting and labeling sequence data. *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*, **8**(June), 282–289.

LAMPLE G., BALLESTEROS M., SUBRAMANIAN S., KAWAKAMI K. & DYER C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 260–270, San Diego, California : Association for Computational Linguistics. DOI : 10.18653/v1/N16-1030.

MARTIN L., MULLER B., SUÁREZ P. J. O., DUPONT Y., ROMARY L., DE LA CLERGERIE É. V., SEDDAH D. & SAGOT B. (2020). Camembert : a tasty french language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

ORTIZ SUÁREZ P. J., SAGOT B. & ROMARY L. (2019). Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures. In P. BAŃSKI, A. BARBARESI, H. BIBER, E. BREITENEDER, S. CLEMATIDE, M. KUPIETZ, H. LÜNGEN & C. ILIADI, Éds., *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*, Cardiff, United Kingdom : Leibniz-Institut für Deutsche Sprache. HAL : hal-02148693.

SRIVASTAVA N., HINTON G., KRIZHEVSKY A., SUTSKEVER I. & SALAKHUTDINOV R. (2014). Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, **15**(56), 1929–1958.

VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł. & POLOSUKHIN I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, **2017-Decem**, 5999–6009.

WENZEK G., LACHAUX M.-A., CONNEAU A., CHAUDHARY V., GUZMAN F., JOULIN A. & GRAVE E. (2019). Ccnet : Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv :1911.00359*.