Travail pratique 3

Sujet 2 – Reconnaissance d'entités nommées (NER)

Travail présenté à Luc Lamontagne
IFT-7022

Réalisés par :

Gabriel Rieger Junqueira
111 192 282

Vincent Gaudreault
111 267 663

Liv Nguekap
111 242 831

18 décembre 2020

Introduction

Dans le cadre du travail pratique 3, nous avons choisi le Sujet 2 – Reconnaissance d'entités nommées (NER) – pour son importance et sa pertinence dans le monde d'aujourd'hui. La reconnaissance d'entités nommées a connu une croissance météorique au fil des dernières années, avec de nombreuses applications dans les domaines tels que : le journalisme – pour la classification des contenus d'articles, les moteurs de recherche – pour rendre les algorithmes plus efficaces, ou même encore dans le domaine du service à la clientèle – pour l'attribution automatique de différentes plaintes aux départements appropriés. Et ceci, juste pour en nommer quelques-uns. Il existe, en pratique, une grande variété d'applications de NER.

De ce fait, pour notre projet, nous avons implanté deux classifieurs sur lesquels on a mené nos expérimentations : un classifieur de modèle bidirectionnel LSTM et un classifieur de modèle bidirectionnel LSTM-CRF. Le but était, en effet, d'explorer et d'analyser les performances de ces deux algorithmes sur les jeux de données à notre disposition.

Dans la suite de notre rapport, nous discuterons du code que nous avons implémenté, des méthodes que nous avons utilisées pour entraîner, valider, et tester nos classifieurs (d'abord sur un corpus de résumés d'article scientifiques pour capter des mots clés désignant des méthodes, des tâches, des matériaux et des termes scientifiques, et ensuite sur un corpus de noms de maladies). Nous continuerons avec une discussion des résultats que nous avons obtenus et des difficultés que nous avons rencontrées. Enfin, nous offrirons notre conclusion.

Traitement des données

Cette partie a été divisée en trois tâches. D'abord, une modification directe dans les fichiers a été effectuée. Le défi a été de mettre les données au bon format pour qu'elles soient prises en charge dans les modèles. Les phrases étaient séparées par une ligne vide, celui-ci a été remplacé directement dans les fichiers par « \t\n » afin d'y ajouter une tabulation. Cela fait, Pandas a été capable de faire un *parsing* du fichier et créer des listes où chaque liste représente une phrase.

Pour préparer les données pour les modèles, les mots (caractéristiques) sont convertis en index et les étiquettes (classes) en un *one-hot-vector*. Le code transforme les mots en index pour les caractéristiques et les classes. Une fois cela fait, des tuples de données sont créés où l'on retrouve le mot indexé à la première position et son étiquette en deuxième position. Ces données sont des vecteurs avec le nombre maximum de mots à choisir, chaque vecteur doit avoir le même nombre d'éléments. Pour garantir cela le mot 'PAD' a été ajouté jusqu'à compléter la taille choisie. Finalement, les mots inconnus ont été remplacés par le mot 'UNK'.

Pour évaluer les résultats, il a été créé un code qui fait le processus inverse du prétraitement. Les prédictions en format de vecteurs de chiffres sont transformées en mots pour être comparées aux étiquettes réelles, cette transformation est appliquée seulement aux classes.

Modèle CRF [1]

Le CRF linéaire s'agit d'un modèle purement statistique. Pour utiliser ce dernier, les caractéristiques doivent être choisies par le développeur. Son entraînement semble relativement difficile étant donné que l'entraînement s'effectue via l'algorithme de Baum-Welch, soit un algorithme d'Expectation-Maximisation.

En règle générale, le modèle apprend en utilisant le contexte entre les mots des concepts de probabilité conditionnelle et de max vraisemblance et implémente des dépendances séquentielles entre les prédictions.

Il est comparable à un modèle MEMM qui suppose que les valeurs à apprendre sont connectées dans une chaîne de Markov. Contrairement au HMM, il n'assume pas que les caractéristiques soient indépendantes les unes par rapport aux autres. De plus, la constante de normalisation et la fonction de caractéristiques ont accès à l'entièreté de la séquence observée. Malgré sa bonne performance, il n'utilise pas l'apprentissage profond. Par souci de temps, il a été décidé de ne pas l'implémenter.

Modèle Bi-LSTM [2]

Il s'agit d'un modèle bidirectionnel d'apprentissage profond qui utilise le *softmax* comme fonction d'activation.

Comme données d'entrée, il reçoit le nombre des mots, le nombre d'étiquettes ainsi que le nombre maximal de mots par phrase.

Son architecture est formée d'une couche d'*embedding*, une couche de *dropout*, une couche bidirectionnelle LSTM et une couche dense comme le montre la figure suivante.

```
Model: "bi_lstm"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        multiple                  298300
_________________________________________________________________
dropout (Dropout)            multiple                  0
_________________________________________________________________
bidirectional (Bidirectional multiple                  120800
_________________________________________________________________
time_distributed (TimeDistri multiple                  2010
=================================================================
Total params: 421,110
Trainable params: 421,110
Non-trainable params: 0
```

Les métriques d'évaluation choisies ont été l'exactitude, la précision, le rappel et le F1-Score.

Modèle Bi-LSTM-CRF [3, 4]

Ce modèle est similaire à celui précédemment décrit. Il s'agit d'un modèle bidirectionnel LSTM d'apprentissage profond, avec l'ajout d'une couche CRF. Cela étant dit, le développement a été beaucoup plus difficile et a généré des résultats inattendus tout en augmentant le surapprentissage. Son architecture est formée des couches d'une couche d'*embedding*, une couche bidirectionnelle LSTM, une couche CRF et d'une couche dense tel que l'affiche la figure suivante.

```
Model: "bi_lstmcrf"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      multiple                  119320
_________________________________________________________________
bidirectional_1 (Bidirection multiple                  28400
_________________________________________________________________
time_distributed_1 (TimeDist multiple                  5050
_________________________________________________________________
crf (CRF)                    multiple                  630
=================================================================
Total params: 153,400
Trainable params: 153,400
Non-trainable params: 0
_________________________________________________________________
None
```

Un des comportements étranges est celui d'avoir la mesure « internal losses » toujours égale à zéro. Les hypothèses qui peuvent expliquer ce fait sont les suivantes :

- Une erreur dans l'architecture qui génère cette instabilité;
- Une erreur provenant de la couche CRF.

Cette couche ne se trouve pas directement dans Keras, mais dans une extension TensorFlow-Addons qui est encore instable. La *pull request* pour ce code est d'ailleurs encore active, de nombreux utilisateurs rapportent des problèmes quant à son utilisation.

Une autre approche probablement plus stable aurait été avec PyTorch [5]. Il a été découvert seulement après que tout le développement dans Keras soit fait.

Une autre difficulté rencontrée lors de l'implémentation de ce modèle est le fait qu'il n'était pas possible de sortir les métriques de façon automatique comme dans le premier cas, les seules métriques possibles dans ce cas ont été la perte pour chercher la meilleure performance et le F1-Score pour le comparer avec le premier modèle développé.

Modèle Bi-LSTM-CNN-CRF [6]

Finalement, le modèle susmentionné se présente comme une amélioration. En plus d'être bidirectionnel et d'avoir une couche CRF, son architecture repose aussi sur une couche de convolution. Cette structure permet de faire des prédictions caractères par caractères ce qui permet de mieux gérer les mots inconnus par exemple. Il a une complexité supérieure aux autres modèles présentés. Étant donné les problèmes d'implémentation du modèle précédent, aucun travail n'a été effectué sur ce dernier.

Résultats pour *science*

Table des résultats F1-Score en test pour le modèle Bi-LSTM. Une variation de 28% à 36.2% est observée.

	Max Length			
Batch Size	**50**	**60**	**70**	**80**
16	**33.1%**	**36.2%**	**35.0%**	**36.0%**
32	**34.8%**	**36.0%**	**34.0%**	**35.5%**
64	**32.8%**	**32.7%**	**28.0%**	**32.1%**

À la figure suivante, on peut remarquer une stabilisation après 30 époques.

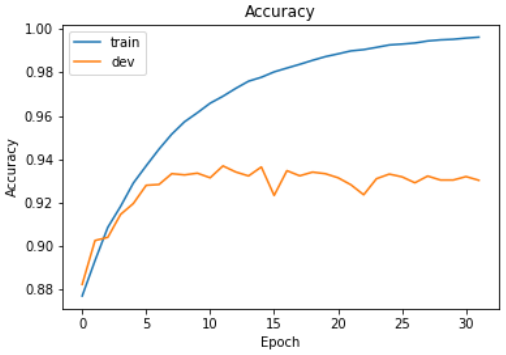


Table des résultats F1-Score en test pour le modèle Bi-LSTM-CRF. On peut remarquer que les résultats sont négligeables, peut-être en raison de l'instabilité de la couche CRF.

	Max Length			
Batch Size	**50**	**60**	**70**	**80**
16	**0.0%**	**1.0%**	**3.9%**	**0.0%**
32	**0.0%**	**0.9%**	**0.0%**	**0.0%**
64	**0.0%**	**0.0%**	**0.0%**	**0.0%**

Résultats pour *disease*

Table des résultats F1-Score en test pour le modèle Bi-LSTM. Une variation de 58.2% à 68.5% est observée.

	Max Length			
Batch Size	**50**	**60**	**70**	**80**
16	**65.3%**	**68.5%**	**63.1%**	**63.1%**
32	**64.6%**	**64.0%**	**66.7%**	**66.8%**
64	**66.8%**	**68.0%**	**58.2%**	**63.7%**

À la figure suivante, on peut remarquer une stabilisation après 25 époques.

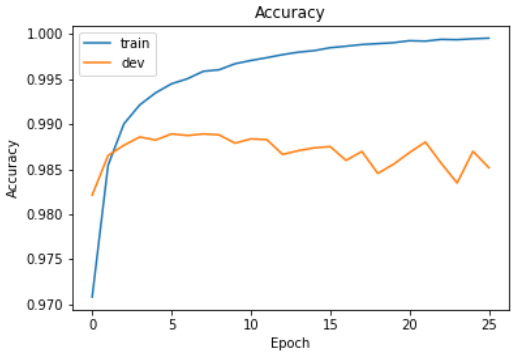


Table des résultats F1-Score en test pour le modèle Bi-LSTM-CRF. On peut remarquer que les résultats sont négligeables encore une fois, peut-être en raison de l'instabilité de la couche CRF.

	Max Length			
Batch Size	**50**	**60**	**70**	**80**
16	**0.0%**	**0.0%**	**0.0%**	**2.0%**
32	**3.4%**	**0.0%**	**2.1%**	**0.0%**
64	**2.1%**	**0.0%**	**0.0%**	**0.0%**

Analyse

Pour évaluer les résultats, il a été choisi d'utiliser l'exactitude, la précision, le rappel ainsi que le F1-Score qui servira aussi pour comparer les deux modèles développés.

Par rapport à la stabilisation, elle se situe après 25 à 30 époques pour le Bi-LSTM et 30 à 35 époques pour le Bi-LSTM-CFR. Dans chaque métrique choisie, on peut observer le surajustement en raison du fait d'avoir un jeu de données d'entraînement réduit et de l'utilisation d'un modèle assez complexe d'apprentissage profond pour faire les prédictions.

Comme attendu, les meilleurs résultats se donnent dans le plus petit lot testé (16 dans ce cas).

Contre-intuitivement, la performance ne s'améliore pas au fur et à mesure qu'on augmente la longueur maximale des séquences. Les meilleures performances sont pour une longueur de 60 mots et non 80 mots.

Fait intéressant à noter, les résultats pour les données de *science* correspondent à la moitié de ceux pour les données de *disease*. Ironiquement, le nombre de classes est aussi divisé par deux.

Parmi toutes les combinaisons essayées dans Bi-LSTM, les variations de performance ne sont pas très expressives, les résultats sont proches avec une différence maximale de 10%.

Pour Bi-LSTM-CFR, malheureusement il n'a pas été possible d'inférer quoi que ce soit, car les scores en test étaient majoritairement à zéro.

Conclusion

Notre projet avait pour but d'approfondir nos connaissances sur les logiciels d'extraction d'entités nommées. Nous souhaitions vraiment comprendre comment les utiliser en pratique. Pour ce faire, nous avons effectué des recherches approfondies sur l'Internet, étudié des articles scientifiques publiés sur le sujet qui offraient différentes approches et modèles NER à considérer. Et tout cela nous a permis d'implémenter notre propre code de manière à entraîner et à tester nos modèles.

La première étape de notre expérimentation a été un prétraitement des données, nécessaire pour rendre les données acceptables par nos modèles. Ceci ne fut pas une tâche triviale, vu la complexité du contenu des deux corpus.

L'étape suivante a été l'entraînement des modèles. Cette étape elle aussi a présenté de nombreux obstacles; particulièrement, la recherche des hyperparamètres pour optimiser les résultats de l'extraction des entités nommées a été perfide, car on a dû payer cher en temps et en ressources matérielles (GPU).

Finalement pour ce qui est de nos résultats, ceux que l'on a obtenus ne furent pas exactement à la hauteur de nos espérances, mais ils étaient quand même intéressants. Surtout que l'on a quand même réussi toutes nos étapes, et de plus notre code est fonctionnel et utilisable.

D'après les résultats que nous avons obtenus pour nos deux modèles, on observe que le modèle LSTM a de meilleures performances sur les deux jeux de données que le modèle LSTM-CRF. On constate aussi que les deux modèles ont de meilleures performances sur le corpus des noms des maladies que sur celui des résumés d'articles scientifiques. Ceci probablement dû à la différence entre le nombre de classes dans chaque jeu de données.

Par ailleurs, nos lacunes ont été dans les connaissances approfondies des algorithmes LSTM et LSTM-CRF nécessaires pour pouvoir améliorer nos résultats et aussi dans l'accès à des GPU pour réduire les temps de traitement des modèles pour chaque étape.

Bibliographie

[1] Charles SUTTON *et al*., « An Introduction to Conditional Random Fields », *arXiv*, [En ligne], 2010, https://arxiv.org/pdf/1011.4088.pdf.

[2] « Guide to sequence tagging with neural networks », *Depends on the definition*, [En ligne], 2017, https://www.depends-on-the-definition.com/guide-sequence-tagging-neural-networks-python/.

[3] « Sequence tagging with LSTM-CRFs », *Depends on the definition*, [En ligne], 2017, https://www.depends-on-the-definition.com/sequence-tagging-lstm-crf/.

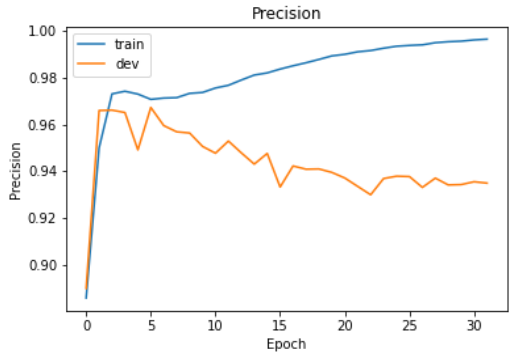
[4] Zhiheng HUANG *et al*., « Bidirectional LSTM-CRF Models for Sequence Tagging », *arXiv*, [En ligne], 2015, https://arxiv.org/pdf/1508.01991.pdf.

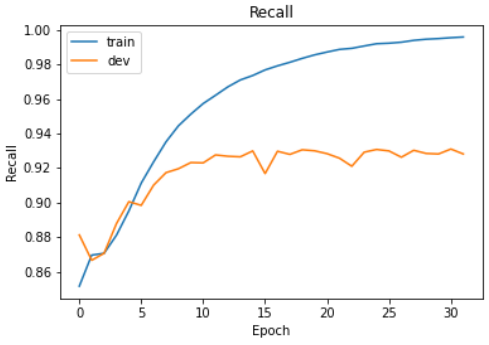
[5] « Advanced: making dynamic decisions and the BI-LSTM CRF », *PyTorch*, [En ligne], 2017, https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html.

[6] Xuezhe MA *et al*., « End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF », *arXiv*, [En ligne], 2016, https://arxiv.org/pdf/1603.01354.pdf.

Annexe A – Sommaire des résultats en test du modèle Bi-LSTM pour *science*

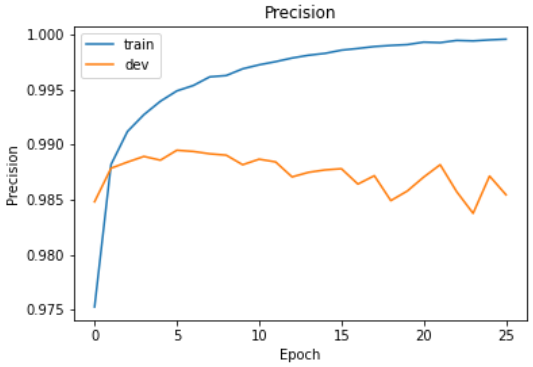
```
18/18 [==============================] - 4s 197ms/step
F1-score: 36.2%
                     precision    recall  f1-score   support

            Generic       0.55      0.60      0.58       241
           Material       0.21      0.41      0.28       158
             Method       0.41      0.45      0.43       427
             Metric       0.29      0.28      0.29        71
OtherScientificTerm       0.31      0.29      0.30       520
               Task       0.22      0.42      0.29       261

          micro avg       0.32      0.41      0.36      1678
          macro avg       0.33      0.41      0.36      1678
       weighted avg       0.35      0.41      0.37      1678
```

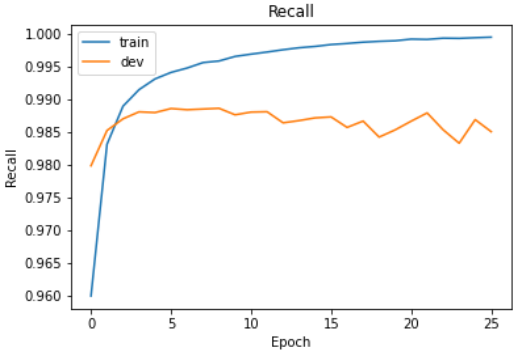




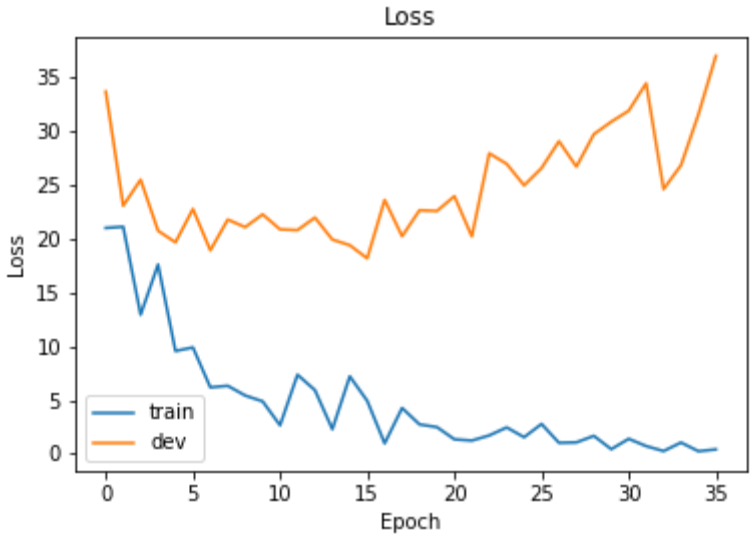
Annexe B – Sommaire des résultats en test du modèle Bi-LSTM pour *disease*

```
30/30 [==============================] - 1s 29ms/step
F1-score: 68.5%
              precision    recall  f1-score   support

      Disease       0.64      0.74      0.69       956

    micro avg       0.64      0.74      0.69       956
    macro avg       0.64      0.74      0.69       956
 weighted avg       0.64      0.74      0.69       956
```





Annexe C – Sommaire des résultats en test du modèle Bi-LSTM-CRF pour *science*



Annexe D – Sommaire des résultats en test du modèle Bi-LSTM-CRF pour *disease*