Réseaux de Neurones Récurrents Multi-tâches pour l'Analyse Automatique d'Arguments

Jean-Christophe Mensonides[1] Sébastien Harispe[1]
Jacky Montmain[1] Véronique Thireau[2]

[1] LGI2P, IMT Mines Ales, Univ Montpellier, Ales, France
[2] Université de Nîmes, CHROME, Rue du Dr Georges Salan, Nîmes, France

jean-christophe.mensonides@mines-ales.fr

Résumé

Dans cet article nous proposons une méthode d'extraction et d'analyse automatique d'arguments à partir de textes bruts, en nous affranchissant de l'utilisation de caractéristiques manuellement définies par des experts. Nous présentons un modèle multi-tâches faisant appel à des techniques d'apprentissage profond, composé de plusieurs couches de réseaux de neurones récurrents. Plus particulièrement, nous tirons parti de paramètres entraînés sur des tâches simples, comme l'étiquetage morpho-syntaxique ou le chunking, afin d'obtenir un modèle capable de traiter des tâches plus complexes nécessitant une compréhension fine du texte.

Mots Clef

Traitement Automatique du Langage Naturel, Extraction d'arguments, Réseaux de neurones récurrents, Apprentissage profond.

Abstract

In this article we propose a method performing automatic extraction and analysis of arguments from raw texts, without using handcrafted features. We introduce a multi-task deep learning model stacking several layers of recurrent neural networks. Specifically, we make use of weight parameters trained on simple tasks, such as Part-Of-Speech tagging and chunking, in order to obtain a model able to handle more complex tasks that require a detailed understanding of the text.

Keywords

Natural Language Processing, Argument mining, Recurrent neural networks, Deep learning.

1 Introduction

L'argumentation est un ensemble de techniques visant à faire adhérer un interlocuteur à un point de vue qui lui est présenté, en construisant un raisonnement à base d'arguments. Bien que l'étude de l'argumentation soit un champ étudié depuis longtemps dans des domaines tels que la philosophie ou la linguistique, l'extraction et l'analyse automatique d'arguments au sein de corpus textuels (aussi appelé *argument mining*) forment des axes de recherche relativement nouveaux. Un système d'argument mining a pour objectif la génération automatique d'un graphe d'arguments à partir de textes non structurés, et peut généralement être divisé en une séquence d'étapes comportant notamment la détection d'arguments et la modélisation des liens unissant ces derniers [1]. Nous nous limitons à une étude de la micro-structure argumentative, consistant à analyser la manière dont différents composants argumentatifs interragissent entre eux au sein d'un même texte.

De manière plus spécifique, Stab et Gurevych [2] ont proposé le corpus Argument Annotated Essays (version 2), contenant 402 dissertations extraites de essayforum.com. La structure argumentative de chaque dissertation a été manuellement annotée suivant un modèle de graphe orienté acyclique connexe, dans lequel les noeuds représentent des composants argumentatifs et les arcs des liens entre ces derniers. Le schéma d'annotation utilisé permet de distinguer trois types de composants argumentatifs : (i) les conclusions majeures, reflétant le point de vue global de l'auteur sur le sujet disserté, (ii) les conclusions intermédiaires, représentant des affirmations qui ne pourraient être acceptées sans justifications complémentaires, et (iii) les prémisses, servant de justifications aux conclusions intermédiaires avancées. Les arcs du graphe sont porteurs d'une étiquette "support" ou "attaque" selon que le composant argumentatif source corrobore ou réfute la cible. Les arcs ne peuvent exister que a) d'une prémisse vers une autre prémisse, b) d'une prémisse vers une conclusion (majeure ou intermédiaire), et c) d'une conclusion intermédiaire vers une autre conclusion (majeure ou intermédiaire).

Afin d'obtenir automatiquement un graphe synthétisant la structure argumentative d'une dissertation, Stab et Gurevych [2] ont proposé une chaîne de traitement constituée de quatre étapes : (1) Délimitation des frontières des composants argumentatifs, (2) Détermination du type

de chaque composant argumentatif, (3) Détermination de l'existence d'un arc entre chaque paire ordonnée de composants argumentatifs, et (4) Etiquetage des arcs existants comme relation de support ou d'attaque.

Dans cet article, nous nous concentrons sur l'étude des tâches (1) et (2), en cherchant à nous affranchir de l'utilisation de caractéristiques définies manuellement par des experts. La section 2 présente un panorama des travaux antérieurs réalisés sur des tâches similaires à (1) et (2). La section 3 décrit le modèle que nous avons mis en place pour traiter les deux tâches évoquées ci-dessus. La section 4 présente les modalités d'entraînement du modèle. La section 5 est consacrée aux expérimentations que nous avons menées et aux résultats obtenus. La section 6 propose des directions et perspectives pour nos futures recherches.

2 Travaux antérieurs

La détection de composants argumentatifs consiste à déterminer les frontières séparant les unités textuelles porteuses d'arguments du reste du texte. Cette tâche est généralement considérée comme un problème de segmentation de texte supervisée au niveau du mot. Les modèles exploitant l'aspect séquentiel des mots, inhérent à la construction d'une argumentation convaincante, sont particulièrement adaptés et utilisés : Madnani et al (2012) utilisent un Conditional Random Field (CRF) afin d'identifier des segments non argumentatifs au sein de dissertations [3], Levy et al (2014) identifient les frontières d'unités textuelles représentant des conclusions supportant ou attaquant le sujet débattu dans des fils de discussions issus de Wikipedia [4], Ajjour et al (2017) utilisent des réseaux de neurones récurrents de type Long Short-Term Memory (LSTM) afin d'extraire des arguments issus de dissertations, d'éditoriaux et de commentaires générés par des internautes [5], Goudas et al (2014) identifient des phrases contenant des arguments avant de déterminer précisément leurs frontières au sein de médias sociaux à l'aide d'un CRF [6], Sardianos et al (2015) déterminent les limites de composants argumentatifs au sein d'articles de presse à l'aide d'un CRF [7], Stab et Gurevych (2017) utilisent un CRF afin d'isoler les composants argumentatifs au sein de dissertations [2], Eger et al (2017) ont recouru à des techniques d'apprentissage profond [8].

La tâche consistant à déterminer le type d'un composant argumentatif (prémisse, conclusion, etc.) a souvent été traité comme un problème de classification de texte supervisée. Eckle-Kohler et al (2015) distinguent des prémisses et des conclusions au sein d'articles de presse à l'aide de Naive Bayes, Random Forest et Support Vector Machine (SVM) [9], Park et Cardie (2014) utilisent un SVM pour déterminer à quel point des affirmations sont justifiées au sein de commentaires d'internautes relatifs à de nouveaux projets de législation [10], Stab et Gurevych (2017) classifient des composants argumentatifs en prémisses, conclusions intermédiaires et conclusions majeures dans des dissertations en utilisant un SVM [2], Persing et Ng (2016) utilisent un classifieur d'entropie maximale afin de déterminer le type de composants argumentatifs [11], Potash et al (2016) utilisent des réseaux de neurones récurrents dits "séquence à séquence" dans l'objectif d'inférer le type de composants argumentatifs [12].

L'étude de modèles multi-tâches, capables de traiter plusieurs problèmes différents en partageant un sous-ensemble commun de paramètres, a fait l'objet d'un engouement récent au sein de la communauté du traitement automatique du langage. Ce type de modèles est bio-inspiré : un être humain est capable de réaliser une multitude de tâches différentes et peut exploiter, quand cela est nécessaire, son savoir-faire acquis concernant la résolution d'un type de problème pour apprendre plus vite à résoudre d'autres types de problèmes. Ruder (2017) énonce les raisons pour lesquelles ce type de modèle est efficace d'un point de vue apprentissage automatique [13] : l'utilisation de plusieurs corpus différents induit une augmentation implicite du nombre d'exemples disponibles pendant la phase d'entraînement. De plus, le modèle doit rechercher des caractéristiques utiles pour l'ensemble des tâches à traiter, ce qui limite la modélisation du bruit dans les données et permet une meilleur généralisation.

Søgaard et Goldberg (2016) montrent qu'induire de la connaissance a priori dans un modèle multi-tâches en hiérarchisant l'ordre des tâches à apprendre permet d'obtenir de meilleurs performances [14]. Yang et al (2016) ont montré qu'entraîner un modèle multi-tâches et multi-langues permettait d'améliorer les performances sur des problèmes où les données ne sont que partiellement annotées [15], Hashimoto et al (2017) obtiennent des résultats compétitifs sur la majorité des tâches d'un même modèle [16]. Le bénéfice d'un modèle multi-tâches n'est cependant pas garanti, et dépend notamment de la distribution des données relatives aux différents problèmes traités (Mou et al (2016) [17], Alonso et Plank (2017) [18], Bingel et Søgaard (2017) [19]).

3 Modèle proposé

Nous proposons un modèle ayant pour objectif 1) de déterminer les frontières de composants argumentatifs présents dans un ensemble de dissertations et 2) de déterminer le type de chaque composant argumentatif dans lesdites dissertations. Nous nous inspirons du travail de Hashimoto et al [16] et optons pour un modèle multi-tâches s'affranchissant de la définition de caractéristiques manuellement définies. Plus particulièrement, nous utilisons des techniques issues de l'apprentissage profond et entraînons un modèle capable d'effectuer de l'étiquetage morpho-syntaxique (EMS), du chunking, de la détection de limites de composants argumentatifs et de la classification de com-

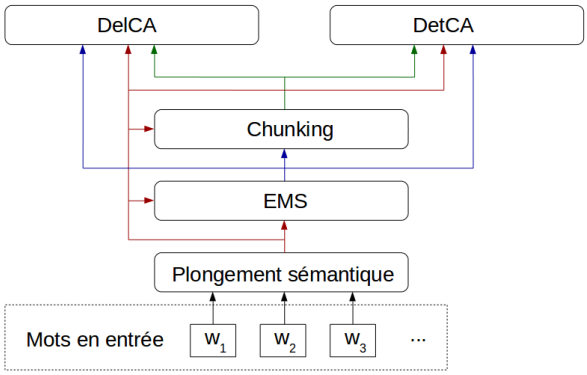

FIGURE 1 – Aperçu de l'architecture utilisée couche par couche. EMS, DelCA et DetCA sont respectivement des acronymes pour Etiquetage Morpho-Syntaxique, Délimitation des Composants Argumentatifs et Détermination du type des Composants Argumentatifs.

posants argumentatifs. Une illustration de l'architecture du modèle est proposée en Figure 1. Les différentes couches utilisées sont présentées ci-dessous.

3.1 Plongement sémantique

Nous utilisons une première couche de plongement sémantique assignant une représentation vectorielle e_t à chaque mot w_t donné en entrée du système. Nous utilisons Glove [20] afin d'obtenir un ensemble de représentations vectorielles entraînées de manière non-supervisée [1]. Les représentations vectorielles de mots sont continuellement optimisées au cours de l'entraînement du modèle sur les différentes tâches explicitées ci-dessous. Les mots pour lesquels nous ne disposons pas de représentation vectorielle pré-entraînée sont transformés en un mot spécial <*UNK*>.

3.2 Etiquetage morpho-syntaxique

La seconde couche du modèle correspond à une tâche d'étiquetage morpho-syntaxique (EMS), consistant à assigner pour chaque mot w_t en entrée du système une étiquette morpho-syntaxique (e.g, nom commun, verbe, déterminant, etc.). Nous utilisons un Gated Recurrent Unit (GRU) [21] bi-directionnel afin d'encoder les séquences de mots en entrée du système.

GRU est un réseau de neurones récurrent utilisant un mécanisme de déclenchement sans utilisation de cellule mémoire séparée. A l'instant t, GRU calcule l'état caché h_t de la manière suivante :

$$h_t = (1 - z_t)n_t + z_t h_{(t-1)}$$

avec

$$n_t = tanh(W_n x_t + b_n + r_t(W_{hn} h_{(t-1)} + b_{hn}))$$

$$r_t = \sigma(W_r x_t + b_r + W_{hr} h_{(t-1)} + b_{hr})$$

$$z_t = \sigma(W_z x_t + b_z + W_{hz} h_{(t-1)} + b_{hz})$$

où x_t représente l'entrée à l'instant t, r_t, z_t et n_t sont respectivement les portes de réinitialisation, d'entrée et de nouveauté, σ représente la fonction sigmoïde, et W et b sont des matrices et vecteurs de paramètres.

En vue d'exploiter le contexte "passé" et "futur" d'un élément d'une séquence de N éléments $[x_1, x_1, ..., x_N]$, nous pouvons construire un encodage bi-directionnel par concaténation des états cachés obtenus par un encodage séquentiel "à l'endroit" (e.g, à l'instant $t = 1$, l'entrée est x_1, à l'instant $t = 2$, l'entrée est x_2, etc.) et un encodage "à l'envers" (e.g, à l'instant $t = 1$, l'entrée est x_N, à l'instant $t = 2$, l'entrée est x_{N-1}, etc.) :

$$\overrightarrow{h_t} = \overrightarrow{GRU}(x_t), t \in [1, N]$$

$$\overleftarrow{h_t} = \overleftarrow{GRU}(x_t), t \in [N, 1]$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$$

Nous utilisons les représentations vectorielles des mots constituant l'exemple en cours comme entrée de la couche EMS :

$$\overrightarrow{h_t^{(1)}} = \overrightarrow{GRU}(e_t)$$

$$\overleftarrow{h_t^{(1)}} = \overleftarrow{GRU}(e_t)$$

$$h_t^{(1)} = [\overrightarrow{h_t^{(1)}}; \overleftarrow{h_t^{(1)}}]$$

Ensuite pour chaque instant t, nous calculons la probabilité d'assigner l'étiquette k au mot w_t de la manière suivante :

$$p(y_t^{(1)} = k|h_t^{(1)}) = \frac{exp(W_{sm_{(1)}} fc_t^{(1)} + b_{sm_{(1)}})}{\sum_{c_1} exp(W_{sm_{(1)}} fc_t^{(1)} + b_{sm_{(1)}})} \quad (1)$$

$$fc_t^{(1)} = relu(W_{fc_{(1)}} h_t^{(1)} + b_{fc_{(1)}}) \quad (2)$$

Avec W et b matrices et vecteurs de paramètres, $relu$ la fonction Unité de Rectification Linéaire [22], et c_1 l'ensemble des classes possibles pour l'étiquette EMS.

3.3 Chunking

Le chunking consiste à assigner une étiquette chunk (chunk nom, chunk verbe, etc.) à chaque mot. Nous calculons les états cachés relatifs au chunking en exploitant ce que le modèle a appris pour la tâche EMS :

$$\overrightarrow{h_t^{(2)}} = \overrightarrow{GRU}([e_t; h_t^{(1)}; y_t^{(EMS)}])$$

$$\overleftarrow{h_t^{(2)}} = \overleftarrow{GRU}([e_t; h_t^{(1)}; y_t^{(EMS)}])$$

$$h_t^{(2)} = [\overrightarrow{h_t^{(2)}}; \overleftarrow{h_t^{(2)}}]$$

Avec $h_t^{(1)}$ l'état caché obtenu à l'instant t pour la tâche EMS et $y_t^{(EMS)}$ la représentation vectorielle pondérée de

1. Le modèle pré-entraîné est issu de https ://nlp.stanford.edu/projects/glove/

l'étiquette EMS. En suivant Hashimoto et al. [16], $y_t^{(EMS)}$ est définit comme suit :

$$y_t^{(EMS)} = \sum_{j=1}^{card(c_1)} p(y_t^{(1)} = j | h_t^{(1)}) l(j) \quad (3)$$

où $l(j)$ est une représentation vectorielle de la j-ème étiquette EMS. Les représentations vectorielles des étiquettes sont pré-entraînées avec GloVe.

La probabilité d'assigner une étiquette chunk à un mot est ensuite calculée de manière similaire à celle pour les étiquettes EMS (équations (1) et (2)), mais avec un ensemble de paramètres propres à la couche chunking.

3.4 Délimitation des composants argumentatifs (DelCA)

L'objectif de cette tâche est de déterminer, au mot près, les frontières de chaque composant argumentatif au sein d'une dissertation. Nous suivons Stab et Gurevych [2] et traitons cette tâche comme un problème de segmentation de texte supervisée dont les étiquettes suivent un IOB-tagset [23] : le premier mot de chaque composant argumentatif porte l'étiquette "Arg-B", les mots restant dudit composant argumentatif portent l'étiquette "Arg-I", et les mots n'appartenant pas à un composant argumentatif portent l'étiquette "O".

Chaque dissertation est traitée comme une unique séquence de mots que nous encodons de la manière suivante :

$$\overrightarrow{h_t^{(3)}} = \overrightarrow{GRU}([e_t; h_t^{(1)}; y_t^{(EMS)}; h_t^{(2)}; y_t^{(chunk)}])$$

$$\overleftarrow{h_t^{(3)}} = \overleftarrow{GRU}([e_t; h_t^{(1)}; y_t^{(EMS)}; h_t^{(2)}; y_t^{(chunk)}])$$

$$h_t^{(3)} = [\overrightarrow{h_t^{(3)}}; \overleftarrow{h_t^{(3)}}]$$

où $y_t^{(chunk)}$ est la représentation vectorielle pondérée de l'étiquette chunk, calculée de manière similaire à celle de l'étiquette EMS (équation (3)).

La probabilité d'assigner une étiquette à un mot est ensuite calculée de manière similaire à celle pour les étiquettes EMS, mais avec un ensemble de paramètres propres à la couche DelCA.

3.5 Déterminer le type des composants argumentatifs (DetCA)

L'objectif de cette tâche est de déterminer le type de chaque composant argumentatif parmi prémisse, conclusion intermédiaire et conclusion majeure. Nous traitons cette tâche comme un problème d'étiquetage de segment. Nous considérons qu'un segment peut être la séquence des mots appartenant à un même composant argumentatif ou la séquence des mots appartenant à une même portion de texte continue dont les mots n'appartiennent pas à un composant argumentatif. La notion de segment est illustré en Figure 2.

[S1] The greater our goal is, the more competition we need. [S2] Take Olympic games which is a form of competition for instance, it is hard to imagine how an athlete could win the game without the training of his or her coach, and the help of other professional staffs such as the people who take care of his diet, and those who are in charge of the medical care [S3] . The winner is the athlete but the success belongs to the whole team. Therefore [S4] without the cooperation, there would be no victory of competition [S5] .
Consequently, no matter from the view of individual development or the relationship between competition and cooperation we can receive the same conclusion that ***[S6] a more cooperative attitudes towards life is more profitable in one's success***.

FIGURE 2 – Un extrait d'une dissertation extrait du corpus. Les passages soulignés par un trait continu constituent des prémisses, ceux soulignés par un trait discontinu constituent des conclusions intermédiaires, et les passages en gras sont des conclusions majeures. Les numéros des segments [S#] sont rajoutés à titre indicatif. Le premier segment correspond à la portion du début du texte jusqu'à la première prémisse. Le second segment correspond à la première prémisse. Le troisième segment correspond à la portion non surlignée entre la première prémisse et la première conclusion intermédiaire, etc.

Nous encodons chaque segment $s_i, i \in [1, L]$ de la manière suivante :

$$\overrightarrow{h_{it}} = \overrightarrow{GRU}([e_{it}; h_{it}^{(1)}; y_{it}^{(EMS)}; h_{it}^{(2)}; y_{it}^{(chunk)}])$$
$$\overleftarrow{h_{it}} = \overleftarrow{GRU}([e_{it}; h_{it}^{(1)}; y_{it}^{(EMS)}; h_{it}^{(2)}; y_{it}^{(chunk)}])$$
$$h_{it} = [\overrightarrow{h_{it}}; \overleftarrow{h_{it}}]$$

où it représente l'instant t du segment s_i.

Afin que le modèle se concentre davantage sur les marqueurs potentiellement importants (comme "I firmly believe that" ou "we can receive the same conclusion that") nous utilisons un mécanisme d'attention [24], nous permettant de surcroît de synthétiser l'information portée par les états cachés d'un segment en un vecteur de taille fixe :

$$u_{it} = tanh(W_{att} h_{it} + b_{att})$$
$$\alpha_{it} = \frac{exp(u_{it}^\intercal u_{att})}{\sum_t exp(u_{it}^\intercal u_{att})}$$
$$sh_i = \sum_t \alpha_{it} h_{it}$$

Avec W_{att}, b_{att} et u_{att} respectivement matrices, biais et vecteurs de paramètres.

Nous encodons ensuite la dissertation à partir des états cachés synthétiques sh_i des segments :

$$\overrightarrow{h_j^{(4)}} = \overrightarrow{GRU}(sh_i), i \in [1, L]$$

$$\overleftarrow{h_j^{(4)}} = \overleftarrow{GRU}(sh_i), i \in [L, 1]$$

$$h_j^{(4)} = [\overrightarrow{h_j^{(4)}}; \overleftarrow{h_j^{(4)}}]$$

La probabilité d'assigner une étiquette à un segment est ensuite calculée de manière similaire à celle pour les étiquettes EMS, mais avec un ensemble de paramètres propres à la couche DetCA.

4 Entraînement du modèle

Nous entraînons le modèle en alternant les couches à chaque "epoch" dans l'ordre suivant : EMS, chunking, DelCA, DetCA. Afin d'évaluer la pertinence d'implémenter un modèle multi-tâches, nous avons entraîné une version du modèle en omettant l'optimisation des couches EMS et chunking (nommée "w/o EMS & chunking") et une version du modèle en optimisant l'ensemble des couches (nommée "w/ EMS & chunking"). Les détails de l'entraînement de chaque couche sont explicités ci-dessous.

4.1 Couche EMS

Nous suivons Hashimoto et al. [16] et notons $\theta_{EMS} = (W_{EMS}, b_{EMS}, \theta_e)$ l'ensemble des paramètres intervenant dans la couche EMS. W_{EMS} représente l'ensemble des matrices de paramètres de la couche EMS, b_{EMS} l'ensemble des biais de la couche EMS et θ_e l'ensemble des paramètres de la couche de plongement sémantique des mots. La fonction de coût est définie par :

$$J^{(1)} = -\sum_s \sum_t log\, p(y_t^{(1)} = k|h_t^{(1)}) + \lambda \left\|W_{EMS}\right\|^2 + \delta \left\|\theta_e - \theta'_e\right\|^2$$

Avec $p(y_t^{(1)} = k|h_t^{(1)})$ la probabilité d'assigner la bonne étiquette k au mot w_t de la séquence de mots s, $\lambda \left\|W_{EMS}\right\|^2$ est la régularisation L2 et $\delta \left\|\theta_e - \theta'_e\right\|^2$ un régularisateur successif. λ et δ sont des hyper-paramètres.

Le régularisateur successif a pour vocation de stabiliser l'entraînement en empêchant θ_e d'être trop modifié spécifiquement par la couche EMS. θ_e étant partagé par l'ensemble des couches du modèle, des modifications trop importantes apportées par l'entraînement de chaque couche empêcherait le modèle d'apprendre convenablement. θ'_e est l'ensemble des paramètres intervenant dans la couche de vectorisation des mots à l'époch précédente.

4.2 Couche chunking

Nous notons $\theta_{chunk} = (W_{chunk}, b_{chunk}, E_{EMS}, \theta_e)$ l'ensemble des paramètres intervenant dans la couche chunking. W_{chunk} et b_{chunk} sont respectivement les matrices de paramètres et biais de la couche chunking, incluant ceux de θ_{EMS}. E_{EMS} est l'ensemble des paramètres caractérisant la représentation vectorielle des étiquettes EMS. La fonction de coût est définie de la manière suivante :

$$J^{(2)} = -\sum_s \sum_t log\, p(y_t^{(2)} = k|h_t^{(2)}) + \lambda \left\|W_{chunking}\right\|^2 + \delta \left\|\theta_{EMS} - \theta'_{EMS}\right\|^2$$

Avec $p(y_t^{(2)} = k|h_t^{(2)})$ la probabilité d'assigner la bonne étiquette k au mot w_t de la séquence de mots s. θ'_{EMS} est l'ensemble des paramètres de la couche EMS obtenus avant d'entamer l'"epoch" courante d'entraînement de la couche chunking.

4.3 Couche DelCA

Notons $\theta_{DelCA} = (W_{DelCA}, b_{DelCA}, E_{EMS}, Echunk, \theta_e)$ l'ensemble des paramètres intervenant dans la couche DelCA, avec W_{DelCA} et b_{DelCA} respectivement matrices de paramètres et biais de la couche DelCA, incluant ceux de la couche chunking et EMS. E_{chunk} est l'ensemble des paramètres caractérisant la représentation vectorielle des étiquettes de la couche chunking. La fonction de coût est définie de la manière suivante :

$$J^{(3)} = -\sum_d \sum_t log\, p(y_t^{(3)} = k|h_t^{(3)}) + \lambda \left\|W_{DelCA}\right\|^2 + \delta \left\|\theta_{chunk} - \theta'_{chunk}\right\|^2$$

Avec $p(y_t^{(3)} = k|h_t^{(3)})$ la probabilité d'assigner la bonne étiquette k au mot w_t de la dissertation d. θ'_{chunk} est l'ensemble des paramètres de la couche chunking obtenus avant d'entamer l'"epoch" courante d'entraînement de la couche DelCA.

4.4 Couche DetCA

Notons $\theta_{DetCA} = (W_{DetCA}, b_{DetCA}, E_{EMS}, Echunk, \theta_e)$ l'ensemble des paramètres intervenant dans la couche DetCA, avec W_{DetCA} et b_{DetCA} respectivement matrices de paramètres et biais de la couche DetCA, incluant ceux de la couche chunking et EMS. La fonction de coût est définie de la manière suivante :

$$J^{(4)} = -\sum_d \sum_i log\, p(y_i^{(4)} = k|sh_i^{(4)}) + \lambda \left\|W_{DetCA}\right\|^2 + \delta \left\|\theta_{chunk} - \theta'_{chunk}\right\|^2$$

Avec $p(y_i^{(4)} = k|sh_i^{(4)})$ la probabilité d'assigner la bonne étiquette k au segment s_i de la dissertation d.

5 Expérimentations et résultats

5.1 Hyper-paramètres et données utilisées

Optimisation. Nous entraînons le modèle en alternant les couches, suivant l'ordre suivant : EMS, chunking, DelCA, DetCA. Chaque couche est entraînée pendant une "epoch" avant de passer à la couche suivante. Nous utilisons Adam [25] comme algorithme d'apprentissage, avec

$\beta_1 = 0.9$, $\beta_2 = 0.999$ et $\epsilon = 10^{-8}$. Le coefficient d'apprentissage est commun à toutes les couches et fixé à 10^{-3} au début de l'entraînement, puis multiplité par 0.75 toutes les 10 "epoch". Afin de limiter le problème d'explosion du gradient, nous redimensionnons sa norme avec une stratégie de gradient clipping [26]. Nous suivons [16] et appliquons un gradient clipping de $min(3.0, profondeur)$, où $profondeur$ représente le nombre de GRU impliquées dans la couche entraînée.

Initialisation des paramètres. Afin de faciliter la propagation du gradient lors de l'entraînement, nous utilisons des matrices orthogonales générées aléatoirement comme états initiaux pour les matrices de paramètres des GRU, comme préconisé par Saxe et al. [27]. Les autres matrices de paramètres sont initialisées avec des valeurs issues d'une loi normal $\mathcal{N}(0, \sqrt{2/n_{in}})$, où n_{in} représente le nombre de neurones entrant dans la couche concernée, comme proposé par He et al [28]. Les vecteurs de biais sont initialisés en tant que vecteurs nuls.

Dimensions vectorielles utilisées. La représentation vectorielle utilisée pour les mots en entrée du système et les représentations vectorielles des étiquettes EMS et chunking sont de dimension 50. Les états cachés des GRU sont de dimension 100 pour toutes les couches du modèle.

Régularisation. En suivant [16], nous fixons les coefficients λ à 10^{-6} pour les matrices de paramètres des GRU et 10^{-5} pour les autres matrices de paramètres. Le coefficient de régularisation successif δ est fixé à 10^{-2} pour toutes les couches. Nous utilisons aussi Dropout [29] sur toutes les couches, avec taux de neurones affectés de 0.2.

Données d'entraînement pour les couches EMS et chunking. Nous utilisons le corpus issu de la tâche partagée CoNLL-2000 [30] avec les étiquettes associées pour entraîner les couches EMS et chunking.

Données d'entraînement pour les couches DelCA et DetCA. Nous utilisons le corpus Argument Annotated Essays (version 2) partagé par Stab et Gurevych [2] en suivant le découpage entraînement/test fourni pour l'entraînement des couches DelCA et DetCA.

Arrêt de l'entraînement. Dans un cas d'entraînement uni-tâche, une pratique généralement adoptée est d'arrêter l'entraînement du modèle peu avant le surapprentissage. Dans le cas de notre modèle, il n'est pas évident de déterminer le meilleur moment pour arrêter l'entraînement, puisque le modèle peut surapprendre sur une tâche particulière, mais pas sur les autres. Ainsi, nous arrêtons l'entraînement du modèle lorsqu'il surapprend sur les couches DelCA et DetCA, et reportons les meilleurs résultats obtenus pour chaque tâche avant le surapprentissage de celle-ci.

DetCa simple. Nous nommons DetCa simple la tâche DetCa avec la modification suivante : tous les segments des dissertations correspondant à des composants argumentatifs sont traités comme ne comportant qu'un unique mot spécial <*VIDE*>. L'hypothèse est que cette transformation forcera le modèle à se concentrer sur le contexte entourant les composants argumentatifs, et l'empêchera donc de se surentraîner en considérant les mots à l'intérieur des composants.

Tâche	w/o EMS & chunking	w/ EMS & chunking
DelCA	0.5934	0.8688
DetCA	0.7464	0.7105
DetCA simple	0.7529	0.7911

TABLE 1 – Macros f1-scores obtenus sur les différentes tâches.

Tâche	F1-score obtenus en [2]	F1-score humain
DelCA	0.867	0.886
DetCA	0.826	0.868

TABLE 2 – F1-scores obtenus sur les tâches DelCA et DetCA par Stab et Gurevych [2] et des agents humains.

5.2 Résultats obtenus

Les résultats obtenus sur les données de test pour les tâches DelCA, DetCA et DetCA simple sont présentés en Table 1. La colonne "w/o EMS & chunking" fait référence à la version du modèle pour laquelle l'optimisation des couches EMS et chunking a été omise. La colonne "w/ EMS & chunking" fait référence à la version du modèle pour laquelle l'optimisation des couches EMS et chunking a été réalisée. Nous prenons comme référence les performances atteintes par des agents humains [2] ainsi que les résultats présentés par Stab et Gurevych [2], illustrés en Table 2.

Evaluation générale des performances. Nous obtenons un macro f1-score de 0.8688 sur DelCA avec la version "w/ EMS & chunking". Ces résultats sont obtenus sans définition de caractéristiques manuelles et sont comparables à ceux enregistrés en [2] ; ils atteignent 98,06% de la performance humaine. Concernant la classification des composants argumentatifs, nous obtenons un macro f1-score de 0.7911 avec DetCA simple pour la version "w/ EMS & chunking", ce qui représente 95,8% des performances obtenues en [2] et 91,1% de la performance humaine.

Pertinence de DetCA simple. Selon nous, les mots formant un composant argumentatif ne sont pas réellement caractéristiques de sa classe, et en se concentrant dessus, le modèle peut être amené à modéliser du bruit l'empêchant de généraliser correctement. En revanche, le contexte dans lequel apparaissent les composants semble très important. Par exemple, des mots tels que "we can receive the same conclusion that" semblent indiquer que l'auteur va annoncer une conclusion intermédiaire ou majeure. Cela peut expliquer la différence de performances entre DetCA et DetCA simple, notamment pour la version "w/ EMS

2. La performance humaine correspond à la moyenne des résultats obtenus par des annotateurs humains, tels que présentés en [2]

& chunking", avec respectivement un f1-score de 0.7105 contre 0.7911, soit une amélioration de 11,3%.

Pertinence du modèle multi-tâches. Les macro f1-scores sur les tâches DelCA et DetCA simple sont respectivement de 0.5934 et 0.7529 pour la version "w/ EMS & chunking" et de 0.8688 et 0.7911, soit des améliorations de 46,4% et 5,1%. Ces résultats permettent donc de valider l'intérêt d'entraîner un modèle multi-tâches et incitent à l'ajout de tâches auxiliaires supplémentaires.

6 Travaux à venir et perspectives

Les résultats obtenus sont encourageants et pourraient sûrement être améliorés, notamment avec une recherche plus d'approfondie d'hyper-paramètres optimaux. La différence de performances entre les versions du modèle "w/ EMS & chunking" et "w/o EMS & chunking" portent à croire qu'implémenter davantage de tâches auxiliaires pourrait être bénéfique. Une piste serait d'introduire une couche modélisant un arbre de dépendances syntaxiques en complément de la couche chunking, comme effectué en [16].

En vue d'implémenter un système complet d'argument mining tel que présenté par Stab et Gurevych [2], nous prévoyons d'implémenter des couches permettant la génération automatique de graphes d'arguments. A cette fin il est nécessaire de déterminer s'il existe un arc entre chaque paire ordonnée de composants argumentatifs, ainsi que d'inférer l'étiquette portée par ledit arc.

7 Conclusion

Cet article a présenté une méthode d'extraction et d'analyse automatique d'arguments à partir de textes bruts. L'utilisation de techniques d'apprentissage profond nous permet de nous affranchir de la définition de caractéristiques manuellement définies. Par ailleurs, l'amélioration des performances de notre système par l'exploitation de paramètres optimisés sur des tâches auxiliaires met en avant l'intérêt de l'utilisation d'un modèle multi-tâches. Nous avons comme perspective la complétion de la chaîne de traitement existante en vue d'obtenir un système capable de synthétiser une dissertation par modélisation automatique d'un graphe d'arguments.

Références

[1] M. Lippi et P. Torroni, Argumentation mining : State of the art and emerging trends, *ACM Transactions on Internet Technology (TOIT)*, 16(2), p.10, 2016.

[2] C. Stab et I. Gurevych, Parsing argumentation structures in persuasive essays, *Computational Linguistics*, 43(3), pp.619-659, 2017.

[3] N. Madnani, M. Heilman, J. Tetreault et M. Chodorow, Identifying high-level organizational elements in argumentative discourse, *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pp. 20-28, Association for Computational Linguistics, 2012.

[4] R. Levy, Y. Bilu, D. Hershcovich, E. Aharoni et N. Slonim, Context dependent claim detection, *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics : Technical Papers*, pp. 1489-1500, 2014.

[5] Y. Ajjour, W.F. Chen, J. Kiesel, H. Wachsmuth et B. Stein, Unit Segmentation of Argumentative Texts, *Proceedings of the 4th Workshop on Argument Mining*, pp. 118-128, 2017.

[6] T. Goudas, C. Louizos, G. Petasis et V. Karkaletsis, Argument extraction from news, blogs, and social media, *Hellenic Conference on Artificial Intelligence*, pp. 287-299, Springer, Cham, 2014.

[7] C. Sardianos, I.M. Katakis, G. Petasis et V. Karkaletsis, Argument extraction from news, *Proceedings of the 2nd Workshop on Argumentation Mining*, pp. 56-66, 2015.

[8] S. Eger, J. Daxenberger et I. Gurevych, Neural End-to-End Learning for Computational Argumentation Mining, *arXiv preprint arXiv :1704.06104*, 2017.

[9] J. Eckle-Kohler, R. Kluge et I. Gurevych, On the role of discourse markers for discriminating claims and premises in argumentative discourse, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2236-2242, 2015.

[10] J. Park et C. Cardie, Identifying appropriate support for propositions in online user comments. *Proceedings of the 1st Workshop on Argumentation Mining*, pp. 29-38, 2014.

[11] I. Persing et V. Ng, End-to-End Argumentation Mining in Student Essays, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, Association for Computational Linguistics, pages 1384–1394, 2016.

[12] P. Potash, A. Romanov et A. Rumshisky, Here's My Point : Joint Pointer Architecture for Argument Mining, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1364-1373, 2017.

[13] S. Ruder, An overview of multi-task learning in deep neural networks, *CoRR*, abs/1706.05098, 2017.

[14] A. Søgaard et Y. Goldberg, Deep multi-task learning with low level tasks supervised at lower layers, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, Vol. 2, pp.231-235, 2016.

[15] Z. Yang, R. Salakhutdinov et W. Cohen, Multi-task cross-lingual sequence tagging from scratch, *arXiv preprint arXiv :1603.06270*, 2016.

[16] K. Hashimoto, C. Xiong, Y. Tsuruoka et R. Socher, A joint many-task model : Growing a neural network for multiple nlp tasks, *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[17] L. Mou, Z. Meng, R. Yan, G. Li, Y. Xu, L. Zhang et Z. Jin, How transferable are neural networks in nlp applications ?, *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 479–489, 2016.

[18] H.M Alonso et B. Plank, When is multitask learning effective ? Semantic sequence prediction under varying data conditions, *15th Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

[19] J. Bingel et A. Søgaard, Identifying beneficial task relations for multi-task learning in deep neural networks, *arXiv preprint arXiv :1702.08303*, 2017.

[20] J. Pennington, R. Socher et C. Manning, Glove : Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532-1543, 2014.

[21] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk et Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, *arXiv preprint arXiv :1406.1078*, 2014.

[22] V. Nair et G.E. Hinton, Rectified linear units improve restricted boltzmann machines, *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807-814, 2010.

[23] L.A. Ramshaw et M.P. Marcus, Text chunking using transformation-based learning, *Natural language processing using very large corpora*, pp. 157-176, Springer, Dordrecht, 1999.

[24] D. Bahdanau, K. Cho et Y. Bengio, Neural machine translation by jointly learning to align and translate, *ICLR*, 2015.

[25] D.P. Kingma et J. Ba, Adam : A method for stochastic optimization, *ICLR*, 2015.

[26] R. Pascanu, T. Mikolov et Y. Bengio, On the difficulty of training recurrent neural networks, *Proceedings of The 30th International Conference on Machine Learning*, pp. 1310–1318, 2013.

[27] A.M. Saxe, J.L. McClelland, S. Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[28] K. He, X. Zhang, S. Ren et J. Sun, Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification, *ICCV*, 2015.

[29] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever et R. Salakhutdinov, Dropout : a simple way to prevent neural networks from overfitting, *Journal of machine learning research*, 15(1) :1929–1958, 2014.

[30] E.F.T.K. Sang, S. Buchholz, Introduction to the CoNLL-2000 shared task : chunking, *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning*, Lisbon, Portugal, vol. 7, 2000, pp. 127–132, 2000.