Jean-Christophe Mensonides, Sébastien Harispe,
Jacky Montmain, Véronique Thireau

Analyse automatique d'arguments et apprentissage multi-tâches : un cas d'étude

Volume 3, nº 3-4 (2022), p. 201-222.

<http://roia.centre-mersenne.org/item?id=ROIA_2022__3_3-4_201_0>

Revue Ouverte d'Intelligence Artificielle
Volume 3, nº 3-4, 2022, 201-222

Analyse automatique d'arguments et apprentissage multi-tâches : un cas d'étude

Jean-Christophe Mensonides[a], Sébastien Harispe[a], Jacky Montmain[a], Véronique Thireau[b]

[a] EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Ales, Alès, France
E-mail : jean-christophe.mensonides@mines-ales.fr, sebastien.harispe@mines-ales.fr, jacky.montmain@mines-ales.fr

[b] Université de Nîmes CHROME Rue du Dr Georges Salan Nîmes, France
E-mail : veronique.thireau@unimes.fr.

Résumé. — Nous proposons une étude sur l'analyse automatique d'arguments *via* des techniques d'apprentissage supervisé exploitant le paradigme de l'apprentissage multi-tâches. Nous définissons pour cela une approche multi-tâches à base d'apprentissage profond que nous évaluons sur un cas d'étude spécifique portant sur l'extraction d'arguments dans un corpus de dissertations. Les résultats obtenus permettent de discuter l'intérêt de définir un modèle multi-tâches unique – optimisé sur différents critères en tirant parti de la diversité des tâches d'apprentissage auxquelles il est confronté – par rapport à un ensemble de classifieurs entraînés de manière indépendante et spécifique. Nous montrons en particulier l'impact de l'ajout de tâches auxiliaires de bas niveau, telles que l'étiquetage morpho-syntaxique et l'analyse de dépendances grammaticales, pour l'obtention de classifieurs multi-tâches performants. Nous observons aussi que l'apprentissage multi-tâches permet l'obtention de modèles efficaces de performances semblables à l'état de l'art pour le cas d'étude traité.

Mots-clés. — Traitement Automatique du Langage Naturel, Extraction d'arguments, Apprentissage Multi-Tâches.

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'argumentation soit un champ étudié depuis longtemps dans des domaines tels que la philosophie, la linguistique ou l'intelligence artificielle [3, 4], l'extraction et l'analyse automatique d'arguments au sein de corpus textuels (aussi appelé *argument mining*) forment des axes de recherche relativement nouveaux [8, 24]. Un système d'analyse d'arguments 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 [26]. Nous nous limitons dans ce travail à une étude de la micro-structure argumentative, consistant à analyser la manière dont différents composants argumentatifs interagissent entre eux au sein d'un même texte.

Parmi les différentes approches proposées par l'état de l'art pour modéliser la problématique d'analyse d'arguments, Stab et Gurevych [51] proposent de distinguer trois types de composants argumentatifs : (i) les conclusions majeures, reflétant le point de vue global d'un auteur sur un 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. Une exemple est proposé en Figure 3.2. Un texte dont l'objectif est de défendre un point de vue peut alors être représenté par un graphe orienté dont les noeuds sont des composants argumentatifs et les arcs des relations de *support* ou d'*attaque* [12]. La génération automatique d'un tel graphe peut être décomposée en 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) Étiquetage des arcs existants comme relation de support ou d'attaque. L'analyse d'arguments vise l'automatisation de ces quatre tâches.

Dans cet article, nous proposons une méthode dont l'objectif est la résolution des tâches (1) à (4) mentionnées ci-dessus. Celles-ci sont traditionnellement traitées par des systèmes distincts qui reposent souvent par ailleurs sur l'utilisation de caractéristiques définies manuellement, *e.g.* en entraînant des modèles d'apprentissage automatique supervisé distincts qui exploitent parfois des informations pré-identifiées par les concepteurs des modèles. Nous nous intéressons dans cette étude à la mise en place et à l'évaluation d'une approche multi-tâches pour aborder l'analyse d'arguments – en apprentissage automatique supervisé, une approche multi-tâches se démarque d'une approche classique visant à entraîner un modèle sur une tâche unique, en cherchant à tirer parti de phases d'apprentissage multiples sur des tâches diverses mais connectées pour l'obtention d'un unique modèle performant. Plus précisément, nous souhaitons évaluer si un modèle unique traitant à la fois l'ensemble des tâches précitées, et des tâches bas niveau classiques en traitement automatique du langage naturel (*e.g.* étiquetage morpho-syntaxique, *chunking*), peut tirer parti des apprentissages croisés effectués pour chacune de ces tâches. Nous nous intéressons en particulier à savoir si une approche multi-tâches permettrait d'obtenir de meilleures performances que celles obtenues par un ensemble de modèles spécifiquement définis pour chacune des tâches. Nous souhaitons aussi considérer un cadre de définition de modèle multi-tâches qui ne repose pas sur l'utilisation de caractéristiques définies manuellement, à l'image des approches récentes de l'état de l'art basées sur des techniques d'apprentissage profond.

Répondre à ce double objectif – (i) définition d'un modèle unique pour l'ensemble des tâches, (ii) excluant l'utilisation de caractéristiques spécifiques définies manuellement – permettrait naturellement l'obtention de systèmes d'analyse d'arguments plus simples à utiliser, à maintenir, et éventuellement à spécialiser à des cadres applicatifs

spécifiques[(1)]. Le développement d'approches visant à considérer des problèmes complexes de manière plus globale, plutôt que comme une décomposition de tâches très spécifiques, est par ailleurs d'un intérêt tout particulier pour le domaine de l'Intelligence Artificielle.

Le papier est organisé de la manière suivante. La section 2 présente un panorama des travaux antérieurs réalisés en analyse d'arguments et en apprentissage multi-tâches ; des détails sur le positionnement de nos travaux sont exposés par la suite. La section 3 décrit le modèle multi-tâches que nous proposons. La section 4 présente les modalités d'entraînement du modèle et détaille différents aspects techniques associés (*e.g.*, fonctions d'erreur). La section 5 est consacrée aux expérimentations et résultats obtenus sur le jeu de données de Stab et Gurevych [51]. La section 6 synthétise nos résultats et présente différentes perspectives d'intérêt pour l'étude des modèles multi-tâches dans un contexte d'analyse d'arguments.

2. Travaux antérieurs

Nous proposons dans cette section une présentation de différents travaux dédiés à l'analyse d'arguments, ainsi qu'un éclairage sur l'apprentissage multi-tâches. La sélection de travaux que nous avons effectuée vise à présenter un aperçu aussi diversifié que concis de la littérature propre à ces thématiques. Les lecteurs intéressés par des études plus approfondies de l'état de l'art sur l'analyse d'arguments pourront faire référence aux différents états de l'art du domaine [34, 26, 8, 24]. Cabrio et Villata [8] proposent notamment une synthèse intéressante de l'état de l'art ; celle-ci présente entre autres les principales approches utilisées et les jeux de données du domaine. Lawrence et Reed proposent quant à eux une revue plus complète et récente du domaine [24].

L'aperçu de l'état de l'art sur l'analyse d'arguments proposé ci-après regroupe les travaux sur (1) la détection des composants argumentatifs, (2) la détermination de leur type, et (3) la prédiction et l'étiquetage des relations, ces deux dernières tâches étant généralement regroupées.

Détection des composants argumentatifs. 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.* [27] utilisent un Conditional Random Field (CRF) afin d'identifier des segments non argumentatifs au sein de dissertations ; Levy *et al.* [25] identifient les frontières d'unités textuelles représentant des conclusions supportant ou attaquant le sujet débattu dans des fils de discussions issus de Wikipedia, d'éditoriaux et de commentaires générés par des internautes ; Goudas *et al.* [17] identifient des phrases contenant des arguments avant de déterminer précisément leurs frontières au sein de médias sociaux à l'aide d'un CRF ; Sardianos *et al.* [44] déterminent les limites de composants argumentatifs au sein d'articles de presse

[(1)] En utilisant des techniques de spécialisation (*fine-tuning*) sur des modèles pré-entraînés.

à l'aide d'un CRF ; Stab et Gurevych [51] utilisent ce même modèle afin d'isoler les composants argumentatifs au sein de dissertations. La plupart des travaux reposent ainsi sur l'utilisation de méthodes nécessitant la définition *ad hoc* et manuelle de caractéristiques d'intérêt pour la classification – une liste de caractéristiques fréquemment utilisées par ces méthodes est rapportée par Cabrio et Villata [8], *e.g.* basée sur l'utilisation de lexiques ou de caractéristiques syntaxiques. Des approches dites profondes à base de réseaux de neurones permettent de s'affranchir de l'utilisation de caractéristiques définies manuellement tout en assurant des performances comparables aux approches précitées. À titre d'exemple, Ajjour *et al.* [1] utilisent des réseaux de neurones récurrents de type LSTM afin d'extraire des arguments issus de dissertations, d'éditoriaux et de commentaires générés par des internautes ; Eger *et al.* [14] utilisent un Bi-LSTM afin d'identifier des composants argumentatifs avec succès au sein de dissertations.

Détermination du type de composant. La tâche consistant à déterminer le type d'un composant argumentatif (prémisse, conclusion, etc.) a souvent été traitée comme un problème de classification de textes supervisée. Eckle-Kohler *et al.* [13] distinguent des prémisses et des conclusions au sein d'articles de presse à l'aide des modèles Naïve Bayes, Random Forest et Support Vector Machines (SVM) ; Park et Cardie [32] 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 ; Stab et Gurevych [51] classifient des composants argumentatifs en prémisses, conclusions intermédiaires et conclusions majeures dans des dissertations en utilisant ce même modèle ; Persing et Ng [36] utilisent un classifieur d'entropie maximale afin de déterminer le type de composants argumentatifs et Potash *et al.* [38] utilisent des réseaux de neurones récurrents dits « séquence à séquence » dans l'objectif d'inférer le type de composants argumentatifs.

Prédiction et étiquetage des relations entre composants. La prédiction de relations entre composants argumentatifs et l'étiquetage de ces relations (comme *support* ou *attaque*) sont des tâches qui peuvent être abordées comme des problèmes de classification supervisés. Stab et Gurevych [51] utilisent un SVM à cette fin ; Cabrio et Villata [7] ont recours à des techniques d'implication sémantique (*textual entailment*) ; Niculae *et al.* [31] expérimentent avec des réseaux de neurones récurrents et des SVM.

Un système complet d'analyse d'arguments implique donc la nécessité de traiter différentes tâches, certaines pouvant être relativement complexes à effectuer automatiquement [8] – l'analyse d'arguments est aujourd'hui un problème ouvert en demande d'approches novatrices pour permettre une amélioration des systèmes permettant une automatisation du traitement. Nous soulignons que les tâches sont traitées de manière isolée dans la littérature, et que la définition d'un système complet nécessite alors l'intégration des composantes définies pour résoudre chaque tâche mentionnée.

Approche multi-tâches et analyse d'arguments. Les approches mentionnées ci-avant se basent sur le paradigme de l'apprentissage supervisé classiquement rencontré dans la littérature en apprentissage automatique appliqué au traitement de textes. Les

modèles permettant de résoudre les différentes tâches sont entraînés de manière dédiée et isolée pour la résolution de chacune des tâches.

Nous cherchons dans cette étude à proposer une approche permettant d'aborder le problème d'analyse d'arguments de manière plus globale, au travers de la définition d'un système unique capable de résoudre les différentes tâches. Nous faisons un parallèle avec des systèmes biologiques et émettons l'hypothèse qu'un système traitant l'ensemble de ces tâches pourrait obtenir de meilleures performances qu'un ensemble de systèmes indépendants.

Ce type d'approches, appelé apprentissage multi-tâches en apprentissage automatique, cherche à traiter différents problèmes partageant généralement implicitement ou explicitement des caractéristiques communes [9]. Ruder [42] énonce les raisons pour lesquelles ce type de modèle peut être efficace du point de vue de l'apprentissage automatique : l'utilisation de plusieurs bases d'apprentissage différentes induit une augmentation du nombre d'exemples disponibles pendant la phase d'entraînement, ce qui est un facteur facilitant l'obtention de modèles performants (en considération du fait que les tâches partagent certaines propriétés). De plus, en cherchant à résoudre différentes tâches de manière simultanée, le modèle se voit en quelque sorte contraint d'identifier des caractéristiques utiles pour l'ensemble des tâches à traiter, ce qui limite la modélisation du bruit et permet *in fine* une meilleure capacité de généralisation, i.e. limitant ainsi le risque de sur-spécialisation. Différents travaux proposent des résultats empiriques permettant de souligner l'intérêt d'aborder certaines tâches complexes *via* une approche multi-tâches. Hashimoto *et al.* [18] obtiennent des résultats compétitifs en utilisant un modèle unique pour traiter différentes tâches en traitement automatique du langage naturel. Yang *et al.* [52] 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 ; Søgaard et Goldberg [48] montrent que de la connaissance *a priori* peut aussi être exprimée lors de la définition d'un modèle multi-tâches en hiérarchisant l'ordre des tâches à apprendre. Plus spécifiquement, en analyse d'arguments, Schulz *et al.* [46] ont conduit des expériences d'ablation de jeu de données tout en incorporant des jeux de données annexes (mais relativement similaires) sur une tâche de détection de composants argumentatifs. 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 [29, 2, 5]. Plusieurs résultats récents soulignent cependant l'apport de ce type d'approches pour l'analyse de structures argumentatives [49, 16, 48].

Dans ce contexte, nous proposons de définir et d'étudier une approche multi-tâches capable de traiter les tâches de détection des composants argumentatifs, détermination du type de composant, ainsi que la prédiction et l'étiquetage des relations entre composants.

3. Une approche multi-tâches de l'analyse d'arguments

Nous proposons un modèle ayant pour objectif la synthèse automatique de textes explicitant des argumentaires en graphes structurés de composants argumentatifs. Ce

processus suit un découpage en quatre étapes similaire à celui proposé par Stab et Gurevych [51], à savoir : (1) déterminer les frontières des composants argumentatifs, (2) déterminer le type de chaque composant argumentatif, (3) déterminer l'existence d'arcs entre chaque paire ordonnée de composants argumentatifs au sein d'un même paragraphe et (4) étiqueter chaque arc comme relation de support ou d'attaque.

Dans la continuité des travaux proposés par Hashimoto *et al.* [18] nous optons pour un modèle multi-tâches s'affranchissant de la définition de caractéristiques définies manuellement. Plus particulièrement, nous utilisons des techniques issues de l'apprentissage profond et entraînons un modèle effectuant, en plus des quatre tâches susmentionnées, de l'étiquetage morpho-syntaxique (EMS), du *chunking* et de l'analyse de dépendances grammaticales (ADG). La Table 3.1 synthétise les différents acronymes utilisés afin de nommer les différentes couches utilisées. Une illustration de l'architecture du modèle est proposée en Figure 3.1. Les différentes couches utilisées sont présentées ci-dessous.

TABLE 3.1. Correspondance des acronymes utilisés pour nommer les différentes couches

Acronyme	Nom étendu
EMS	Étiquetage morpho-syntaxique
ADG	Analyse de dépendances grammaticales
DelCA	Délimitation des composants argumentatifs
DetCA	Détermination des composants argumentatifs
DetArcs	Détermination des arcs

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. e_t est la concaténation de deux composantes :

- d'une part la représentation vectorielle du mot w_t. Ces représentations sont pré-entraînées de manière non-supervisée avec un modèle *skipgram* [28]. Elles sont continuellement optimisées durant l'entraînement du modèle. Les mots pour lesquels nous ne disposons pas de représentations vectorielles pré-entraînées sont transformés en un mot spécial <*UNK*>.
- d'autre part une représentation vectorielle construite comme la moyenne arithmétique des n-grams de caractères composants le mot w_t [6]. Par exemple pour n=3, le mot « article » est représenté comme la moyenne des représentations vectorielles des n-grams suivants :

  <ar, art, rti, tic, cle, le>

  où < et > sont des caractères spéciaux représentant respectivement le début et la fin d'un mot. Cette méthode permet de calculer une représentation

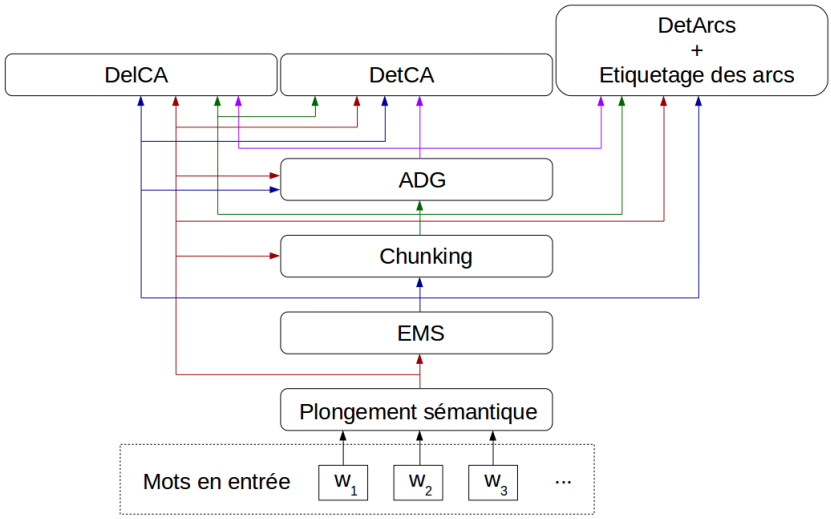

FIGURE 3.1. Aperçu de l'architecture utilisée couche par couche.

vectorielle pour chaque mot, y compris ceux non rencontrés dans le jeu de données de pré-entraînement. Les représentations vectorielles des n-grams de caractères sont pré-entraînées avec un modèle *skipgram*.

3.2. ÉTIQUETAGE MORPHO-SYNTAXIQUE

La seconde couche du modèle correspond à une tâche d'étiquetage morpho-syntaxique (EMS), consistant à assigner à chaque mot w_t en entrée du système une étiquette morpho-syntaxique (*e.g.*, nom commun, verbe, déterminant, etc.).

Nous exploitons le contexte « passé » et « futur » d'une séquence de N éléments $[x_1, x_1, ..., x_N]$, en construisant un encodage bi-directionnel h_t à l'instant t par un encodage séquentiel « à l'endroit » $\overrightarrow{h_t}$ (*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 » $\overleftarrow{h_t}$ (*e.g.* à l'instant $t = 1$, l'entrée est x_N, à l'instant $t = 2$, l'entrée est x_{N-1}, etc.). L'encodage est réalisé à l'aide d'un réseau de neurones récurrent Long Short-Term Memory (LSTM) [20]. Les éléments $[x_1, x_1, ..., x_N]$ sont substitués par leur représentation vectorielle :

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

Nous notons $[\overrightarrow{h_t^{(1)}}; \overleftarrow{h_t^{(1)}}]$ la concaténation des vecteurs $\overrightarrow{h_t^{(1)}}$ et $\overleftarrow{h_t^{(1)}}$. L'exposant (i) utilisé dans les notations, *e.g.* $h_t^{(i)}$, sert à identifier la couche associée.

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

$$p(y_t^{(1)} = k|h_t^{(1)}) = \frac{\exp\left(\left[W_{sm}^{(1)} fc_t^{(1)} + b_{sm}^{(1)}\right]_k\right)}{\sum_{c \in c_1} \exp\left(\left[W_{sm}^{(1)} fc_t^{(1)} + b_{sm}^{(1)}\right]_c\right)} \tag{3.1}$$

$$fc_t^{(1)} = relu(W_{fc}^{(1)} h_t^{(1)} + b_{fc}^{(1)}) \tag{3.2}$$

Avec W et b matrices et vecteurs de paramètres[(2)], *relu* la fonction Unité de Rectification Linéaire [30] (ici appliquée à chaque valeur du vecteur), et c_1 l'ensemble des classes possibles pour l'étiquette EMS. On note $[\cdot]_i$ la valeur du vecteur associée à la classe i.

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 :

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

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

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

Avec $h_t^{(1)}$ l'état caché obtenu à l'instant t pour la tâche EMS et $y_t^{(\text{EMS})}$ la représentation vectorielle pondérée de l'étiquette EMS. En suivant l'approche proposée par Hashimoto *et al.* [18], $y_t^{(\text{EMS})}$ est défini comme suit :

$$y_t^{(\text{EMS})} = \sum_{j=1}^{\text{card}(c_1)} p(y_t^{(1)} = j|h_t^{(1)}) l_j \tag{3.3}$$

où l_j est la représentation vectorielle de la j-ème étiquette EMS. Les représentations vectorielles des étiquettes sont pré-entraînées avec GloVe [35].

La probabilité d'assigner une étiquette chunk à un mot est ensuite calculée de manière similaire à celle utilisée pour les étiquettes EMS (équations 3.1 et 3.2), mais avec un ensemble de paramètres propres à la couche chunking.

[(2)] les indices sm et fc font référence aux notions de *softmax* et de couche *fully connected*.

3.4. Analyse de dépendances grammaticales (ADG)

L'analyse de dépendances grammaticales consiste à construire, pour chaque phrase, un arbre de dépendances grammaticales. Dans un tel arbre, chaque mot caractérise un mot parent par l'intermédiaire d'un arc portant une étiquette (*e.g.*, auxiliaire, adjectif qualificatif, ponctuation, etc.). Nous calculons les états cachés relatifs à l'analyse de dépendances grammaticales de la manière suivante :

$$\overrightarrow{h_t^{(3)}} = \overrightarrow{\text{LSTM}}\left(\left[e_t; h_t^{(1)}; y_t^{(\text{EMS})}, h_t^{(2)}; y_t^{(\text{chunk})}\right]\right)$$
$$\overleftarrow{h_t^{(3)}} = \overleftarrow{\text{LSTM}}\left(\left[e_t; h_t^{(1)}; y_t^{(\text{EMS})}, h_t^{(2)}; y_t^{(\text{chunk})}\right]\right)$$
$$h_t^{(3)} = \left[\overrightarrow{h_t^{(3)}}; \overleftarrow{h_t^{(3)}}\right]$$

où y_t est la représentation vectorielle pondérée de l'étiquette chunk, calculée de manière similaire à celle de l'étiquette EMS (équation 3.3).

Afin de déterminer le mot parent du mot w_t , nous appliquons la fonction de score suivante [18] :

$$score(t, j) = h_t^{(3)} \cdot \left(W_{DAG} h_j^{(3)}\right)$$

où j est l'indice du mot parent candidat et W_{DAG} est une matrice de paramètres. Un vecteur de paramètres supplémentaire $h_{N+1}^{(3)}$ est introduit afin de représenter la racine de l'arbre.

La probabilité d'assigner w_j comme parent de w_t est obtenue par normalisation des scores :

$$p(j|h_t^{(3)}) = \frac{\exp(score(t, j))}{\sum_{m=1, m\neq t}^{N+1} \exp(score(t, m))} \tag{3.4}$$

La probabilité d'assigner une étiquette à chaque relation est ensuite calculée de la manière suivante :

$$p\left(y_t^{(3)} = k \mid h_t^{(3)}, h_j^{(3)}\right) = \frac{\exp\left(\left[W_{sm}^{(3)} fc_t^{(3)} + b_{sm}^{(3)}\right]_k\right)}{\sum_{c\in c_3} \exp\left(\left[W_{sm}^{(3)} fc_t^{(3)} + b_{sm}^{(3)}\right]_c\right)} \tag{3.5}$$

$$fc_t^{(3)} = relu\left(W_{fc}^{(3)}\left[h_t^{(3)}; h_j^{(3)}\right] + b_{fc}^{(3)}\right) \tag{3.6}$$

avec W et b matrices et vecteurs de paramètres et c_3 l'ensemble des étiquettes possibles.

3.5. 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 traitons cette tâche comme un problème de segmentation de texte supervisée dont les étiquettes suivent un IOB-tagset [40] : le premier mot de chaque composant argumentatif porte l'étiquette *Begin*, le reste des mots dudit composant argumentatif porte l'étiquette *In*, et les mots n'appartenant pas à un composant argumentatif portent l'étiquette *Out*.

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

$$\overrightarrow{h_t^{(4)}} = \overrightarrow{\text{LSTM}}\left(\left[e_t; h_t^{(1)}; y_t^{(\text{EMS})}; h_t^{(2)}; y_t^{(\text{chunk})}; h_t^{(3)}\right]\right)$$
$$\overleftarrow{h_t^{(4)}} = \overleftarrow{\text{LSTM}}\left(\left[e_t; h_t^{(1)}; y_t^{(\text{EMS})}; h_t^{(2)}; y_t^{(\text{chunk})}; h_t^{(3)}\right]\right)$$
$$h_t^{(4)} = \left[\overrightarrow{h_t^{(4)}}; \overleftarrow{h_t^{(4)}}\right]$$

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

3.6. 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 illustrée en Figure 3.2.

[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 3.2. Un extrait d'une dissertation issu du corpus Argument Annotated Essays (version 2) [51]. Les passages soulignés par un trait continu constituent des prémisses, ceux en italique 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 :

$$\overrightarrow{h_{it}} = \overrightarrow{\text{LSTM}}\left(\left[e_{it}; h_{it}^{(1)}; y_{it}^{(\text{EMS})}; h_{it}^{(2)}; y_{it}^{(\text{chunk})}, h_t^{(3)}\right]\right)$$
$$\overleftarrow{h_{it}} = \overleftarrow{\text{LSTM}}\left(\left[e_{it}; h_{it}^{(1)}; y_{it}^{(\text{EMS})}; h_{it}^{(2)}; y_{it}^{(\text{chunk})}, h_t^{(3)}\right]\right)$$
$$h_{it} = \left[\overrightarrow{h_{it}}; \overleftarrow{h_{it}}\right]$$

où it représente l'instant t du segment s_i.

Nous appliquons une opération de max-pooling sur h_{it} afin d'obtenir une représentation vectorielle sh_i de taille fixe pour chaque segment. Nous pouvons ensuite construire une représentation vectorielle de chaque dissertation à partir des états cachés des segments :

$$\overrightarrow{h_j^{(5)}} = \overrightarrow{\text{LSTM}}(sh_i), i \in [1, L]$$
$$\overleftarrow{h_j^{(5)}} = \overleftarrow{\text{LSTM}}(sh_i), i \in [L, 1]$$
$$h_j^{(5)} = \left[\overrightarrow{h_j^{(5)}}; \overleftarrow{h_j^{(5)}}\right]$$

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.

3.7. Détermination des arcs (DetArcs)

L'objectif de cette tâche est de déterminer, pour chaque paire ordonnée $(c_i, c_j), i \neq j$ de composants argumentatifs, s'il existe une relation de *support* ou d'*attaque* de c_i vers c_j. Pour cela nous construisons un arbre de dépendances entre segments avec une méthode similaire à celle employée pour l'analyse de dépendances grammaticales (équation 3.4).

3.8. Etiquetage des arcs

L'objectif de cette tâche est de déterminer la nature de chaque arc existant parmi *support* ou *attaque*. Nous utilisons une méthode similaire à celle employée pour l'étiquetage de l'analyse de dépendances grammaticales (équations 3.5 et 3.6).

4. Entraînement du modèle

Nous entraînons le modèle en alternant les couches à chaque *epoch*[3] dans l'ordre suivant : EMS, chunking, ADG, DelCA, DetCA, DetArcs et étiquetage des arcs. Les détails de l'entraînement de chaque couche sont explicités ci-dessous.

[3] Une *epoch* correspond à un cycle d'apprentissage sur l'ensemble complet des données concernant la tâche à traiter.

4.1. Couche EMS

Nous notons $\theta_{\text{EMS}} = (W_{\text{EMS}}, b_{\text{EMS}}, \theta_e)$ 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 :

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

Avec $p(y_t^{(1)} = k \mid h_t^{(1)})$ la probabilité d'assigner la bonne étiquette k au mot w_t de la séquence de mots s, $\|W_{\text{EMS}}\|^2$ est la somme des normes L2 des matrices de paramètres de W_{EMS} et $\left\|\theta_e - \theta'_e\right\|^2$ 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'*epoch* précédente.

4.2. Couche chunking

Nous notons $\theta_{\text{chunk}} = (W_{\text{chunk}}, b_{\text{chunk}}, E_{\text{EMS}}, \theta_e)$ 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 :

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

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 ADG

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

chunk. La fonction de coût est définie de la manière suivante :

$$J^{(3)} = -\sum_{s}\sum_{t} \log p\left(\alpha \mid h_t^{(3)}\right) p\left(\beta \mid h_t^{(3)}, h_\alpha^{(3)}\right) + \lambda \left\|W_{\text{ADG}}\right\|^2 + \delta \left\|\theta_{\text{chunk}} - \theta'_{\text{chunk}}\right\|^2$$

Avec $p(\alpha \mid h_t^{(3)})$ la probabilité de déterminer le bon parent w_α au mot w_t et $p(\beta \mid h_t^{(3)}, h_\alpha^{(3)})$ la probabilité d'assigner la bonne étiquette au couple (w_t, w_α).

4.4. Couche DelCA

Nous utilisons la fonction de coût suivante pour optimiser la couche DelCA :

$$J^{(4)} = -\sum_{d}\sum_{t} \log p\left(y_t^{(4)} = k \mid h_t^{(4)}\right) + \lambda \left\|W_{\text{DelCA}}\right\|^2 + \delta \left\|\theta_{\text{ADG}} - \theta'_{\text{ADG}}\right\|^2$$

Avec $p(y_t^{(4)} = k \mid h_t^{(4)})$ la probabilité d'assigner la bonne étiquette k au mot w_t de la dissertation d.

4.5. Couche DetCA

Nous utilisons la fonction de coût suivante pour optimiser la couche DetCA :

$$J^{(5)} = -\sum_{d}\sum_{i} log\, p(y_i^{(5)} = k | sh_i^{(5)}) + \lambda \left\|W_{\text{DetCA}}\right\|^2 + \delta \left\|\theta_{\text{ADG}} - \theta'_{\text{ADG}}\right\|^2$$

Avec $p(y_i^{(5)} = k \mid sh_i^{(5)})$ la probabilité d'assigner la bonne étiquette k au segment s_i de la dissertation d.

4.6. Couches DetArcs et étiquetage des arcs

Nous utilisons la fonction de coût suivante pour optimiser les couches DetArcs et étiquetage des arcs :

$$J^{(6)} = -\sum_{d}\sum_{i} \log p\left(\alpha \mid sh_i^{(5)}\right) p\left(\beta \mid sh_i^{(5)}, sh_\alpha^{(5)}\right) + \lambda \left\|W_{\text{DetArcs}}\right\|^2 + \delta \left\|\theta_{\text{ADG}} - \theta'_{\text{ADG}}\right\|^2$$

Avec $p(\alpha \mid sh_i^{(5)})$ la probabilité de déterminer le bon segment parent s_α au segment s_i et $p(\beta \mid sh_i^{(5)}, sh_\alpha^{(5)})$ la probabilité d'assigner la bonne étiquette au couple (sh_i, sh_α).

5. Expérimentations et résultats

5.1. Données d'entraînement

Nous utilisons le corpus issu de la tâche partagée CoNLL-2000 [43] avec les étiquettes associées pour entraîner les couche EMS et chunking. Nous utilisons le jeu de données Universal Dependencies EnglishWeb Treebank [47] pour entraîner la couche ADG.

Pour les couches DelCA, DetCA, DetArcs et étiquetage des arcs nous utilisons le corpus Argument Annotated Essays (version 2) [51] 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.

Ce corpus contient 3 832 prémisses, 1 506 conclusions intermédiaires et 751 conclusions majeures. Les arcs des graphes sont porteurs d'une étiquette *support* dans 3 613 cas (94 %) et d'une étiquette *attaque* dans 219 cas (6 %). Ces 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).

Le degré de concordance de trois annotateurs a été mesuré pour un sous-ensemble de 80 dissertations. Sur une tâche consistant à identifier la présence de chaque type de composant argumentatif au sein d'une phrase, les annotateurs ont obtenu un coefficient κ de Fleiss [15][(4)] de 0,833 pour les prémisses, 0,635 pour les conclusions intermédiaires et 0,877 pour les conclusions majeures. Sur la tâche de détermination de la nature des arcs entre composants argumentatifs, ils ont respectivement obtenu $\kappa = 0{,}737$ et $\kappa = 0{,}708$ pour les relations d'attaque et de support. Landis et Koch [23] estiment que l'accord inter-annotateurs est important pour $\kappa > 0{,}61$ et presque parfait pour $\kappa > 0{,}81$.

Afin de pouvoir comparer nos résultats à ceux reportés par Stab et Gurevych [51], nous limitons la tâche DetArcs aux paires ordonnées $(c_i, c_j), i \neq j$ de composants argumentatifs appartenant à un même paragraphe.

Nous conservons le découpage entraînement/test fourni en utilisant 10 % des données d'entraînement pour créer un corpus de validation d'hyper-paramètres.

5.2. Hyper-paramètres utilisés

Nous précisons ci-dessous les hyper-paramètres considérés dans notre étude. Des informations sur des aspects techniques non discutés pourront être obtenues à partir du code de calcul utilisé pour nos expérimentations.[(5)]

[(4)] Le κ de Fleiss permet de mesurer, pour un nombre d'annotateurs supérieur à deux, le degré d'accord entre ces derniers.

[(5)] `https://github.com/lgi2p/ROIA-2020-Mensonides-al`

5.2.1. *Optimisation*

Nous entraînons le modèle en alternant les couches, suivant l'ordre suivant : EMS, chunking, DelCA, DetCA, DetArcs. Chaque couche est entraînée pendant une *epoch* avant de passer à la couche suivante. Nous utilisons Adam [22] comme algorithme d'apprentissage tel que suggéré par la littérature [41]. Le coefficient d'apprentissage est commun à toutes les couches et fixé à 10^{-3} au début de l'entraînement puis diminué de moitié toutes les cinq *epoch*. Afin de limiter le problème d'explosion du gradient, nous redimensionnons sa norme avec une stratégie de *gradient clipping* [33]. Nous suivons Hashimoto *et al.* [18] et appliquons un *gradient clipping* de $\min(3, \textit{profondeur})$, où *profondeur* représente le nombre de LSTM impliqués dans la couche entraînée.

5.2.2. *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 LSTM, comme préconisé par Saxe *et al.* [45]. Les autres matrices de paramètres sont initialisées avec des valeurs issues d'une loi normale $\mathcal{N}(0, \sqrt{\frac{2}{n_{in}}})$, où n_{in} représente le nombre d'entrées dans la couche concernée, comme proposé par He *et al.* [19]. Les vecteurs de biais sont initialisés en tant que vecteurs nuls.

5.2.3. *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 100. Les états cachés des LSTM sont de dimension 200 pour toutes les couches du modèle.

5.2.4. *Régularisation*

Nous fixons les coefficients λ à 10^{-6} pour les matrices de paramètres des LSTM et 10^{-5} pour les autres matrices de paramètres [18]. Le coefficient de régularisation successif δ est fixé à 10^{-2} pour les couches EMS, chunking et ADG. Il est fixé à 10^{-3} pour les autres couches. Nous utilisons Dropout [50] sur toutes les couches, avec un taux de neurones affectés de 40 %.

5.3. Résultats obtenus

Nous présentons les résultats obtenus lors de nos expérimentations sur le jeu de test pour les tâches DelCA, DetCA, DetArcs et Etiquetage des arcs en Table 5.1. Plusieurs versions du modèle décrit ci-dessus sont présentées.

La colonne « Modèle multi-tâches avec auxiliaires » fait référence aux performances atteintes par un modèle multi-tâches avec un unique jeu de paramètres : les performances reportées correspondent à celles obtenues pour une même *epoch*. Lors de l'entraînement, les fonctions de coût $J^{(1)}$, $J^{(2)}$, $J^{(3)}$, $J^{(4)}$, $J^{(5)}$ et $J^{(6)}$ ont été successivement minimisées. L'*epoch* a été choisie telle que la somme des fonctions de coûts

TABLE 5.1. Macros f1-scores obtenus sur les différentes tâches

Tâches	Modèle multi-tâches avec auxiliaires	Modèle multi-tâches avec auxiliaires – différentes *epoch*	Modèle multi-tâches sans auxiliaires	Modèles mono-tâche
DelCA	0,867	**0,890**	0,787	0,876
DetCA	0,795	0,795	0,782	0,795
DetArcs	0,726	0,734	0,697	0,714
Étiquetage des arcs	0,683	**0,743**	0,499	0,742

$J^{(4)} + J^{(5)} + J^{(6)}$ (celles propres à l'analyse d'arguments) soit minimale sur le jeu de validation.

La colonne « Modèle multi-tâches avec auxiliaires – différentes *epoch* » correspond à un modèle similaire à celui de la première colonne mais pour lequel le jeu de paramètres n'est pas unique : les performances reportées correspondent à celles obtenues à différentes *epoch*. L'*epoch* choisie est celle minimisant la fonction de coût d'intérêt pour la tâche concernée sur le jeu de validation (*e.g.*, pour DelCA seule $J^{(4)}$ est considérée).

La colonne « Modèle multi-tâches sans auxiliaires » fait référence aux performances atteintes par un modèle multi-tâches avec un unique jeu de paramètres pour lequel l'optimisation des fonctions $J^{(1)}, J^{(2)}$ et $J^{(3)}$ ont été omises. L'objectif est de pouvoir observer l'impact de l'ajout de tâches auxiliaires telles que l'étiquetage morpho-syntaxique sur les performances d'un modèle multi-tâches.

La colonne « Modèle mono-tâche » présente les résultats obtenus lorsque seule la fonction de coût associée à la tâche d'intérêt a été optimisée, permettant d'observer l'intérêt de se placer dans un contexte d'apprentissage multi-tâches.

La Table 5.2 présente les résultats reportés dans la littérature. À noter que l'approche par Stab et Gurevych [51] utilise de nombreuses caractéristiques définies par des experts comme entrées de leurs modèles. Celles d'Ajjour *et al.* [1] et Potash *et al.* [38] s'appuient moins largement sur ce type de caractéristiques, mais conservent des informations sur la structure et sur l'emplacement des composants argumentatifs à traiter.

TABLE 5.2. F1-scores reportés dans la littérature d'intérêt

Tâche	F1-scores obtenus par Stab et Gurevych [51]	F1-scores obtenus par Potash *et al.* [38]	F1-scores obtenus par Ajjour *et al.* [1]
DelCA	0,867	Non-traité	0,885
DetCA	0,826	**0.849**	Non-traité
DetArcs	0,751	**0,767**	Non-traité
Etiquetage des arcs	0,680	Non-traité	Non-traité

Nous pouvons observer que les résultats obtenus par notre approche sont la plupart du temps comparables avec ceux reportés dans la littérature. Le modèle multi-tâches avec auxiliaires, lorsque le jeu de paramètres utilisé est choisi tel qu'il minimise les fonctions de coûts associés aux tâches DelCA et Etiquetage des arcs, sont particulièrement compétitifs.

L'impact de l'ajout de tâches auxiliaires peut être observé en comparant les performances reportées pour un modèle multi-tâches à jeu de paramètres unique avec auxiliaires et pour un modèle multi-tâches à jeu de paramètres unique sans auxiliaires (Table 5.1, colonnes 1 et 3). L'optimisation des fonctions de coût associées aux tâches auxiliaires permet d'obtenir une amélioration relative des performances respectives de 10 %, 2 %, 4 % et 37 % sur les tâches DelCA, DetCA, DetArcs et d'étiquetage des arcs. On observe donc, dans le contexte de l'évaluation réalisée, un intérêt à l'intégration de tâches auxiliaires dans un contexte d'apprentissage multi-tâches relatif à l'analyse d'arguments.

Nous observons aussi une amélioration des performances lorsque, pour chaque tâche à considérer, une version du modèle multi-tâches est spécifiquement sélectionnée pour celle-ci (Table 5.1, colonne 2). On préférera donc, lorsque les conditions le permettent, utiliser autant de jeux de paramètres qu'il y a de tâches à résoudre plutôt qu'un unique modèle paramétrique. La comparaison des résultats des modèles multi-tâches à plusieurs jeux de paramètres et des modèles mono-tâches (Table 5.1, colonnes 2 et 4) permettent aussi d'observer un accroissement des performances, bien que modeste.

Sur les tâches DelCA et d'étiquetage des arcs, les performances obtenues sont sensiblement meilleures que celles reportées par Stab et Gurevych [51]. En revanche, l'approche proposée par Potash *et al.* [38] montre des performances supérieures sur les tâches DetCa et DetArcs – nous rappelons que ce modèle exploite des caractéristiques structurelles et positionnelles que l'on sait intéressantes pour traiter des tâches de traitement automatique du langage complexes [21]. Il semblerait que l'approche multi-tâches proposée ne permette pas l'identification d'indicateurs aussi informatifs pour la résolution des tâches précitées.

5.3.1. *Analyse de l'erreur*

La Table 5.3 présente la matrice de confusion du modèle unique avec auxiliaires pour la tâche DetCA sur le jeu de test. Bien que le modèle distingue correctement les prémisses des conclusions majeures, une part importante des erreurs est imputée aux conclusions intermédiaires : sur 304 conclusions intermédiaires, 83 (27 %) sont classifiées comme prémisses et 26 (9 %) sont classifiées comme conclusions majeures. De manière intéressante, le même phénomène est observé par Stab & Gurevych [51], qui suggèrent que lorsque des prémisses forment une chaîne (*e.g.*, une prémisse supporte une seconde prémisse, la seconde prémisse supporte une conclusion), le modèle peine à déterminer correctement la nature des composants argumentatifs en milieu de chaîne. Si l'on se réfère au degré de concordance des annotateurs, on peut observer qu'identifier les conclusions intermédiaires est la tâche la plus ardue pour des agents

humains ($\kappa = 0{,}635$ pour les conclusions intermédiaires contre $\kappa = 0{,}833$ et $\kappa = 0{,}877$ pour les prémisses et conclusions majeures).

TABLE 5.3. Matrice de confusion du modèle unique avec auxiliaires sur la tâche DetCa.

		Prédiction		
		Prémisse	Conclusion intermédiaire	Conclusion majeure
Vérité	Prémisse	702	104	3
	Conclusion intermédiaire	83	195	26
	Conclusion majeure	2	11	140

La Table 5.4 présente la matrice de confusion du modèle unique avec auxiliaires pour la tâche DetArcs sur le jeu de test. La distribution des classes étant largement biaisée en faveur de « Non existence d'un lien », le modèle a tendance à sur-considérer celle-ci au détriment de l'autre. En effet lorsque deux composants argumentatifs au sein d'un même paragraphe sont liés, le modèle n'est capable de le déterminer que dans 55 % des cas. Des approches visant à entraîner les modèles avec un jeu de données ayant une représentation équilibrée des deux classes pourraient être bénéfiques pour répondre à cette limitation. Une telle stratégie amènerait cependant une réduction drastique du nombre de données labéllisées et n'est pas compatible avec le type de modèles testés dans notre étude (compter une réduction de la taille du jeu de données de 1100 à 184 données labéllisées dans le cas de la constitution d'un jeu de données parfaitement équilibré).

TABLE 5.4. Matrice de confusion du modèle unique avec auxiliaires sur la tâche DetArcs.

		Prédiction	
		Non existence d'un lien	Existence d'un lien
Vérité	Non existence d'un lien	3 736	377
	Existence d'un lien	367	442

La Table 5.5 présente la matrice de confusion du modèle unique avec auxiliaires pour la tâche d'étiquetage des arcs sur le jeu de test[(6)]. On peut observer un phénomène de biais dans la distribution des classes en faveur des relations de support, ce qui

[(6)] Il y a plus de liens à étiqueter en Table 6 que de liens existants entre composants argumentatifs au sein d'un même paragraphe en Table 5. Cela est dû au fait que les conclusions intermédiaires, bien que parfois non présentes dans le même paragraphe que les conclusions majeures, possèdent forcément un attribut d'attaque ou de support envers ces conclusions majeures.

explique les performances du modèle sur la classe sous-représentée. Une remarque semblable à celle proposée ci-dessus pour la tâche DetArcs peut être formulée.

TABLE 5.5. Matrice de confusion du modèle unique avec auxiliaires sur la tâche d'étiquetage des arcs.

		Prédiction	
		Support	Attaque
Vérité	Support	979	42
	Attaque	57	35

6. CONCLUSION

Nous avons proposé une méthode d'extraction et d'analyse automatique d'arguments à partir de textes bruts basée sur (i) une approche d'apprentissage automatique supervisé multi-tâches, (ii) non dépendante de caractéristiques manuellement définies par des experts. Celle-ci obtient des performances comparables à celles des systèmes de l'état de l'art sur la plupart des tâches d'intérêt. Ces résultats intéressants soulignent la possibilité de développer un système complet d'analyse d'arguments reposant sur un modèle unique, qui serait plus simple à maintenir et potentiellement à spécialiser pour des contextes applicatifs spécifiques. Des études complémentaires, notamment sur une variété plus large de jeux de données du domaine, sont naturellement nécessaires pour discuter plus largement l'intérêt de l'apprentissage multi-tâches dans les différents contextes d'analyse d'arguments supervisés retrouvés dans la littérature (*e.g.* variantes de modélisation de la notion d'argument, langues, quantité de données annotées). L'évaluation de l'impact du choix et de l'ordre des tâches auxiliaires considérées nous semble particulièrement important. Il serait aussi intéressant d'observer si le couplage de l'approche multi-tâches proposée avec des méthodes d'apprentissage par transfert plus traditionnelles, *e.g.* à base de modèles de langue récents [11, 37, 39], apporterait une plus-value. Les résultats obtenus invitent dans tous les cas à l'étude des approches multi-tâches, encore peu étudiées dans la littérature, pour la résolution de problèmes complexes abordés comme une succession de tâches spécifiques.

REMERCIEMENTS

Ces travaux ont bénéficié d'un accès aux moyens de calcul de l'IDRIS au travers de l'allocation de ressources 2020-AD011011309 attribuée par GENCI. Nous remercions les équipes de l'IDRIS pour leur disponibilité et professionnalisme. Nous souhaitons aussi remercier les relecteurs et les organisateurs de la revue ROIA qui ont par leurs retours constructifs contribué aux travaux exposés dans cet article.

Bibliographie

[1] Y. Ajjour, W.-F. Chen, J. Kiesel, H. Wachsmuth & B. Stein, « Unit segmentation of argumentative texts », in *Proceedings of the 4th Workshop on Argument Mining*, Association for Computational Linguistics, 2017, p. 118-128.

[2] H. M. Alonso & B. Plank, « When is multitask learning effective ? Semantic sequence prediction under varying data conditions », `https://arxiv.org/abs/1612.02251`, 2016.

[3] L. Amgoud & H. Prade, « Using arguments for making and explaining decisions », *Artificial Intelligence* **173** (2009), nº 3-4, p. 413-436.

[4] ———, « Can AI models capture natural language argumentation ? », *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)* **6** (2012), nº 3, p. 19-32.

[5] J. Bingel & A. Søgaard, « Identifying beneficial task relations for multi-task learning in deep neural networks », `https://arxiv.org/abs/1702.08303`, 2017.

[6] P. Bojanowski, E. Grave, A. Joulin & T. Mikolov, « Enriching word vectors with subword information », *Transactions of the Association for Computational Linguistics* **5** (2017), p. 135-146.

[7] E. Cabrio & S. Villata, « A natural language bipolar argumentation approach to support users in online debate interactions », *Argument & Computation* **4** (2013), nº 3, p. 209-230.

[8] ———, « Five Years of Argument Mining : a Data-driven Analysis », in *IJCAI*, 2018, p. 5427-5433.

[9] R. Caruana, « Multitask learning », *Machine learning* **28** (1997), nº 1, p. 41-75.

[10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk & Y. Bengio, « Learning phrase representations using RNN encoder-decoder for statistical machine translation », `https://arxiv.org/abs/1406.1078`, 2014.

[11] J. Devlin, M.-W. Chang, K. Lee & K. Toutanova, « Bert : Pre-training of deep bidirectional transformers for language understanding », `https://arxiv.org/abs/1810.04805`, 2018.

[12] P. M. Dung, « On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games », *Artificial intelligence* **77** (1995), nº 2, p. 321-357.

[13] J. Eckle-Kohler, R. Kluge & I. Gurevych, « On the role of discourse markers for discriminating claims and premises in argumentative discourse », in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2015, p. 2236-2242.

[14] S. Eger, J. Daxenberger & I. Gurevych, « Neural end-to-end learning for computational argumentation mining », `https://arxiv.org/abs/1704.06104`, 2017.

[15] J. L. Fleiss, « Measuring nominal scale agreement among many raters », *Psychological bulletin* **76** (1971), nº 5, p. 378-382.

[16] A. Galassi, M. Lippi & P. Torroni, « Argumentative link prediction using residual networks and multi-objective learning », in *Proceedings of the 5th Workshop on Argument Mining*, Association for Computational Linguistics, 2018, p. 1-10.

[17] T. Goudas, C. Louizos, G. Petasis & V. Karkaletsis, « Argument extraction from news, blogs, and social media », in *Hellenic Conference on Artificial Intelligence*, Lecture Notes in Computer Science, vol. 8445, Springer, 2014, p. 287-299.

[18] K. Hashimoto, C. Xiong, Y. Tsuruoka & R. Socher, « A joint many-task model : Growing a neural network for multiple nlp tasks », `https://arxiv.org/abs/1611.01587`, 2016.

[19] K. He, X. Zhang, S. Ren & J. Sun, « Delving deep into rectifiers : Surpassing human-level performance on imagenet classification », in *Proceedings of the IEEE international conference on computer vision*, 2015, p. 1026-1034.

[20] S. Hochreiter & J. Schmidhuber, « Long short-term memory », *Neural computation* **9** (1997), nº 8, p. 1735-1780.

[21] A. Karanikolos & I. Refanidis, « Encoding Position Improves Recurrent Neural Text Summarizers », in *Proceedings of the 3rd International Conference on Natural Language and Speech Processing*, Association for Computational Linguistics, 2019, p. 142-150.

[22] D. P. Kingma & J. Ba, « Adam : A method for stochastic optimization », `https://arxiv.org/abs/1412.6980`, 2014.

[23] J. R. Landis & G. G. Koch, « The measurement of observer agreement for categorical data », *biometrics* (1977), p. 159-174.

[24] J. Lawrence & C. Reed, « Argument Mining : A Survey », *Computational Linguistics* **45** (2020), nº 4, p. 765-818.

[25] R. Levy, Y. Bilu, D. Hershcovich, E. Aharoni & N. Slonim, « Context dependent claim detection », in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics : Technical Papers*, Dublin City University and Association for Computational Linguistics, 2014, p. 1489-1500.

[26] M. Lippi & P. Torroni, « Argumentation mining : State of the art and emerging trends », *ACM Transactions on Internet Technology (TOIT)* **16** (2016), nº 2, p. 10.

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

[28] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado & J. Dean, « Distributed representations of words and phrases and their compositionality », in *Advances in neural information processing systems*, 2013, p. 3111-3119.

[29] L. Mou, Z. Meng, R. Yan, G. Li, Y. Xu, L. Zhang & Z. Jin, « How transferable are neural networks in nlp applications ? », `https://arxiv.org/abs/1603.06111`, 2016.

[30] V. Nair & G. E. Hinton, « Rectified linear units improve restricted boltzmann machines », in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, p. 807-814.

[31] V. Niculae, J. Park & C. Cardie, « Argument mining with structured SVMs and RNNs », `https://arxiv.org/abs/1704.06869`, 2017.

[32] J. Park & C. Cardie, « Identifying appropriate support for propositions in online user comments », in *Proceedings of the first workshop on argumentation mining*, 2014, p. 29-38.

[33] R. Pascanu, T. Mikolov & Y. Bengio, « On the difficulty of training recurrent neural networks », in *International conference on machine learning*, 2013, p. 1310-1318.

[34] A. Peldszus & M. Stede, « Joint prediction in MST-style discourse parsing for argumentation mining », in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, p. 938-948.

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

[36] I. Persing & V. Ng, « End-to-end argumentation mining in student essays », in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 2016, p. 1384-1394.

[37] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee & L. Zettlemoyer, « Deep contextualized word representations », `https://arxiv.org/abs/1802.05365`, 2018.

[38] P. Potash, A. Romanov & A. Rumshisky, « Here's my point : Joint pointer architecture for argument mining », `https://arxiv.org/abs/1612.08994`, 2016.

[39] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei & I. Sutskever, « Language models are unsupervised multitask learners », *OpenAI Blog* **1** (2019), nº 8.

[40] L. A. Ramshaw & M. P. Marcus, « Text chunking using transformation-based learning », in *Natural language processing using very large corpora*, Springer, 1999, p. 157-176.

[41] S. Ruder, « An overview of gradient descent optimization algorithms », `https://arxiv.org/abs/1609.04747`, 2016.

[42] ———, « An overview of multi-task learning in deep neural networks », `https://arxiv.org/abs/1706.05098`, 2017.

[43] E. F. Sang & S. Buchholz, « Introduction to the CoNLL-2000 shared task : Chunking », `https://arxiv.org/abs/cs/0009008`, 2000.

[44] C. Sardianos, I. M. Katakis, G. Petasis & V. Karkaletsis, « Argument extraction from news », in *Proceedings of the 2nd Workshop on Argumentation Mining*, 2015, p. 56-66.

[45] A. M. Saxe, J. L. McClelland & S. Ganguli, « Exact solutions to the nonlinear dynamics of learning in deep linear neural networks », `https://arxiv.org/abs/1312.6120`, 2013.

[46] C. Schulz, S. Eger, J. Daxenberger, T. Kahse & I. Gurevych, « Multi-task learning for argumentation mining in low-resource settings », `https://arxiv.org/abs/1804.04083`, 2018.

[47] N. Silveira, T. Dozat, M.-C. De Marneffe, S. R. Bowman, M. Connor, J. Bauer & C. D. Manning, « A Gold Standard Dependency Corpus for English. », in *LREC*, 2014, p. 2897-2904.

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

[49] W. Song, Z. Song, L. Liu & R. Fu, « Hierarchical multi-task learning for organization evaluation of argumentative student essays », in *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, IJCAI, 2020, p. 3875-3881.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever & R. Salakhutdinov, « Dropout : a simple way to prevent neural networks from overfitting », *The journal of machine learning research* **15** (2014), nº 1, p. 1929-1958.

[51] C. Stab & I. Gurevych, « Parsing argumentation structures in persuasive essays », *Computational Linguistics* **43** (2017), nº 3, p. 619-659.

[52] Z. Yang, R. Salakhutdinov & W. Cohen, « Multi-task cross-lingual sequence tagging from scratch », `https://arxiv.org/abs/1603.06270`, 2016.

Abstract. — We present a method performing automatic extraction and analysis of arguments from raw texts in a multi-task learning framework. The results obtained show that training a single model on different tasks can lead to good performances. We explore the impact of adding low-level auxiliary tasks, such as Part-Of-Speech tagging and dependency parsing, on a model's ability to handle more complex tasks. Our experiments show that multi-task learning can lead to competitive results when performing automatic argument mining.

Keywords. — Natural Language Processing, Argument mining, Multi-Task Learning.

Manuscrit reçu le 19 novembre 2018, révisé le 4 novembre 2019, accepté le 2 octobre 2020.