Automatic Detection and Classification of Argument Components using Multi-task Deep Neural Network

Jean-Christophe Mensonides, Sébastien Harispe, Jacky Montmain
LGI2P, IMT Mines Ales,
Ales, France
firstname.lastname@mines-ales.fr

Véronique Thireau
CHROME, Université de Nîmes,
Nîmes, France
veronique.thireau@unimes.fr

Abstract

In this article we propose a novel method for automatically extracting and classifying argument components from raw texts. We introduce a multi-task deep learning framework exploiting weight parameters trained on auxiliary simple tasks, such as Part-Of-Speech tagging or chunking, in order to solve more complex tasks that require a fine-grained understanding of natural language. Interestingly, our results show that the use of advanced deep learning techniques framed in a multi-task setting enables competing with state-of-the-art systems that depend on handcrafted features.

1 Introduction

Argumentation consists in a set of methods aiming at making an interlocutor adhere to an introduced point of view or conclusion -or at least to increase its adherence to the latter. In its simplest form, argumentation is a reasoning process selecting and structuring premises to attack or defend a given conclusion. Although the study of argumentation is well established in fields such as Logic, Philosophy, or Linguistics, automatic extraction and analysis of arguments from natural language, commonly referred as Argument Mining, is a relatively new research area. Advances in Argument Mining are of major importance for Natural Language Understanding and Processing, in particular due to their several direct applications and relationships with other tasks, e.g. fake news detection, knowledge base enrichment and population, source trustworthiness estimation. To date, most advanced argument mining systems aims at generating argument-based graphs identifying and structuring premises, claims and conclusion from raw texts (Cabrio and Villata, 2018). They can usually be split into sequences of subtasks including argument detection and argument linking (Lippi and Torroni, 2016). This article focuses on analyzing argumentative micro-structure, i.e., how different argumentative components interact with each other within a single text.

Three types of argument components are often used in the annotation scheme considered in Argument mining: (i) major claims, reflecting the author's standpoint on the debated topic, (ii) claims, which are statements needing further justifications to be accepted, and (iii) premises, which are justifications used in order to make a claim stand (Stab and Gurevych, 2017). Those components are further structured into a directly acyclic graph in which nodes account for argumentative components and edges account for oriented links between them. Directed edges in the graph are labeled either as a *support* or *attack* relationship, and are only allowed a) from a premise to another premise, b) from a premise to a (major) claim, and c) from a claim to a (major) claim. Such an annotation schema is adopted by (Stab and Gurevych, 2017) in the Argument Annotated Essays corpus (version 2) composed of 402 manually annotated student essays taken from essayforum.com.

A pipeline of treatments is generally applied to automatically obtain the graph reflecting the underlying argumentative structure of an essay. The following intuitive decomposition involving four subtasks is moreover often considered in practice (Stab and Gurevych, 2017): (1) argument components identification, (2) argument components classification, (3) assessment of the existence of directed edges between argument components, and (4) tagging of the existing directed edges either as a support or as an attack relation-

ship. This article will focus on solving subtasks (1) and (2). It presents a novel multi-task approach to argument mining that does not require handcrafted features as input. We are in particular interested by evaluating if recent deep learning techniques, such as recurrent neural network mixed with multi-task learning, can compete with traditional approaches based on handcrafted features.

The paper is organized as follows. Section 2 introduces an overview of related work proposed on tasks similar to (1) and (2). Section 3 introduces our approach to solve those two tasks. Section 4 describes the training details of the proposed model. Section 5 describes our experiments and results. Section 6 provides directions and perspectives for future works.

2 Related work

Argument components detection consists in determining the boundaries separating the textual units carrying arguments from the rest of the text. This task is generally considered as a supervised text segmentation problem at word level. Models exploiting the sequential aspect of texts, inherent in the construction of a convincing argumentation, seem particularly adapted and are often used. (Madnani et al., 2012) used a CRF (Conditional Random Field) to identify non-argumentative segments within dissertations. (Levy et al., 2014) identified the boundaries of textual units detailing conclusions which were supporting or attacking topics discussed in threads from Wikipedia. (Ajjour et al., 2017) used LSTM (Long short-term memory, recurrent neural network) to extract arguments from essays, editorials, and from user-generated comments. (Goudas et al., 2014) first identified sentences containing arguments and then detected their boundaries within social media using a CRF. (Sardianos et al., 2015) determined argument components boundaries in news articles using also using a CRF. Similarly, (Stab and Gurevych, 2017) used a CRF to extract argument components in essays. (Eger et al., 2017) leveraged deep learning techniques to extract arguments from raw texts.

Determining the type of argument components (premise, conclusion, etc.) has often been treated as a supervised text classification problem. (Eckle-Kohler et al., 2015) distinguished premises and conclusions in news articles using Naive Bayes, Random Forest and SVM (Support Vector Machine). (Park and Cardie, 2014) also used a SVM to determine the extent to which claims are justified in citizen's comments related to possible new legislation projects. (Stab and Gurevych, 2017) classified argumentative components into premises, claims and major claims in essays using a SVM. (Persing and Ng, 2016) used maximum entropy classification to determine the type of argument components. (Potash et al., 2016) used sequence-to-sequence recurrent neural networks to infer the type of argument components.

Multi-tasks models are able to handle several different problems by sharing a subset of shared parameters. They have been subject to recent interest within the Natural Language Processing community (Hashimoto et al., 2016; Søgaard and Goldberg, 2016; Eger et al., 2017; Yang et al., 2016). This type of models is bio-inspired: human beings are able to carry out a multitude of different tasks and can exploit, when necessary, knowledge related to different types of problems, making the learning of new tasks faster and easier. (Ruder, 2017) states the reasons why this type of model is effective from a machine learning point of view: the use of several different corpora induces an implicit increase in the number of examples available during the training phase. In addition, the model has to look for characteristics which may be useful for all the tasks to be processed, which limits the noise modeling and thus, leads to a better generalization.

(Søgaard and Goldberg, 2016) showed that inducing *a priori* knowledge in a multi-task model, by ordering the tasks to be learned, leads to better performance. (Yang et al., 2016) have shown that driving a multi-task and multi-language model can improve performance on problems where data is only partially annotated. (Hashimoto et al., 2016) obtained competitive results on several different tasks with a single model. However, we should note that there is no guarantee on the benefits of using multi-task models, and that their success depends on the data distribution related to the various problems treated (Mou et al., 2016; Alonso and Plank, 2016; Bingel and Søgaard, 2017). (Schulz et al., 2018) proposed a multi-task framework to perform end-to-end argument mining. The result they obtained are very promising. In this paper, we are interested in leveraging auxiliary informa-

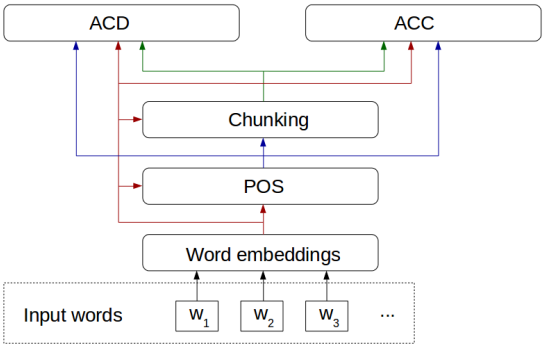

Figure 1: Overview of the model architecture (Layer-wise). POS, ACD and ACC are respectively acronyms for Part-Of-Speech, Argument Components Detection and Argument Components Classification.

tions such as Part-Of-Speech and Chunking tags in a multi-task learning setup, in order to perform argument component detection and classification.

3 Proposed approach

We propose a model which aims at 1) determining argument components boundaries within a set of essays and 2) determining the type of each argument component within the latter essays. We draw inspiration from the work of (Hashimoto et al., 2016) and opt for a multi-task model without the definition of handcrafted features. Specifically, we use deep learning techniques and develop a model that performs Part-Of-Speech (POS) tagging, chunking, argument components boundaries detection, and argument components classification. An overview of the model architecture is given in Figure 1. The different layers used in the architecture are described below.

3.1 Word embeddings

We first use a word embedding layer, assigning a vector representation e_t to each word w_t given in input to the system. We use GloVe (Pennington et al., 2014) to obtain a set of pre-trained embeddings in an unsupervised fashion[1]. Note that, word embeddings are continually optimized while training the model on the different tasks described below. Out-of-vocabulary words are mapped to a special $<UNK>$ token.

[1]We used pre-trained embeddings from https://nlp.stanford.edu/projects/glove/.

3.2 POS tagging

The second layer of the model corresponds to a POS tagging task, consisting in assigning a POS tag (noun, verb, adjective, etc.) to each word w_t given in input to the system. We use a bi-directional Gated Recurrent Unit (GRU) (Cho et al., 2014) to encode input word sequences.

GRU is a recurrent neural network using a gating mechanism and avoiding the use of a separate memory cell. At each time step t, GRU computes the hidden state h_t as follows:

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

with

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

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

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

where x_t is the input at time step t, r_t, z_t and n_t are respectively the reset, update and new gates, σ is the sigmoid function, and W and b are matrice and vector parameters.

In order to exploit the past and future contexts of an element from a sequence of N inputs $[x_1, x_1, ..., x_N]$, we construct a bi-directional encoding by concatenating the hidden states obtained with a forward encoding (e.g, at time step $t = 1$, the input is x_1, at time step $t = 2$, the input is x_2, etc.) and a backward encoding (e.g, at time step $t = 1$, the input is x_N, at time step $t = 2$, the input is x_{N-1}, etc.):

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

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

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

We use word embeddings as input of the POS tagging layer:

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

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

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

Then for each time step t we compute the probability of assigning the label k to the word w_t as follows:

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

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

where W and b are parameter matrices and vectors, $ReLU$ is the Rectified Linear Unit function (Nair and Hinton, 2010), and c_1 is the set of possible POS tags.

3.3 Chunking

Chunking is a task aiming at assigning a chunk tag (noun phrase, verb phrase, etc.) to each word. We compute hidden chunking states by exploiting what the model has learned for the POS tagging task:

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

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

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

where $h_t^{(1)}$ is the hidden state obtained at time step t on the POS tagging task and $y_t^{(POS)}$ is the weighted POS label embedding. Following (Hashimoto et al., 2016), $y_t^{(POS)}$ is defined as follows:

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

where $l(j)$ is an embedding of the j-th POS tag. POS tag embeddings are pre-trained using GloVe.

The probability to assign a chunk tag to each word is then computed in a similar way to the one for POS tags (eq. (1) and (2)), but with a set of parameters specific to the chunking layer.

3.4 Argument Components Detection (ACD)

Argument components detection aims at delimiting the boundaries of each argument component within essays at the word level. We follow (Stab and Gurevych, 2017) and treat this task as a supervised text segmentation problem, where labels follow an IOB-tagset (Ramshaw and Marcus, 1999): the first word of each argument component carries an "Arg-B" label, remaining words of said argument component bear an "Arg-I" tag, and the words not belonging to any argument component bear an "O" tag.

Each essay is considered as a single word sequence which we encode as follows:

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

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

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

where $y_t^{(chunk)}$ is the weighted chunk label embedding, computed in a similar way as the one for POS labels (eq. (3)).

The probability to assign a chunk tag to a word is then computed in a similar way as the one for POS labels, but with a set of parameters specific the ACD layer.

3.5 Argument Components Classification (ACC)

Argument components classification aims at determining the type of each argument component between premise, claim and major claim. We treat this task as a segment labeling problem. We consider that a segment can be the sequence of words belonging to a same argument component or can be the sequence of words belonging to a same portion of continuous text whose words do not belong to an argument component. The notion of segment is illustrated in Figure 2.

We encode each segment $s_i, i \in [1, L]$ as follows:

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

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

$$h_{it} = [\overrightarrow{h_{it}}; \overleftarrow{h_{it}}]$$

where it is the time step t for the segment s_i.

[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: Excerpt from an essay of the corpus illustrating the notion of segments. Text regions underlined by a solid line are premises, those underlined by a dashed line are claims, and the bold regions are major claims. Segment numbers [S#] were added as indications. The first segment is the region from the beginning of the text to the first premise. The second segment corresponds to the first premise. The third segment is the not underlined region between the first premise and the first claim, and so on.

In order to help the model focusing on the most important markers (such as "I firmly believe that" or "we can receive the same conclusion that") we use an attention mechanism (Bahdanau et al., 2014), which in addition allows us to synthesize the information carried by segments hidden states into fixed size vectors:

$$u_{it} = tanh(W_{att} h_{it} + b_{att})$$

$$\alpha_{it} = \frac{exp(u_{it}^{\intercal} u_{att})}{\sum_t exp(u_{it}^{\intercal} u_{att})}$$

$$sh_i = \sum_t \alpha_{it} h_{it}$$

where W_{att}, b_{att} and u_{att} are respectively parameter matrices, bias and vectors.

Then we encode each essay using the synthetic segments hidden states sh_i:

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

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

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

The probability of assigning a label to each segment is then computed similarly to that for POS tags, but with a set of parameters specific to the ACC layer.

4 Model Training

For each epoch of training, we optimized the model's parameters for each layer. That is, at each epoch we trained the layers in the following order: POS tagging, chunking, ACD and ACC. In order to assess the relevance of implementing a multi-task model, we trained two versions of the model: a version where we trained every layer (referred as "w/ POS & chunking"), and a version for which we voluntary omitted to train the POS tagging and chunking layers (referred as "w/o POS & chunking"). The training details of each layer are described below.

4.1 POS tagging layer

Following (Hashimoto et al., 2016), we denote $\theta_{POS} = (W_{POS}, b_{POS}, \theta_e)$ the set of parameters involved in the POS tagging layer. W_{POS} represents the set of parameter matrices for the POS tagging layer, b_{POS} the set of biases of the POS tagging layer, and θ_e the set of parameters of the words embedding layer. The cost function is defined as:

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

where $p(y_t^{(1)} = k | h_t^{(1)})$ is the probability of assigning the right label k to the word w_t of the word sequence s, $\lambda \left\| W_{POS} \right\|^2$ is the L2 regularization term and $\delta \left\| \theta_e - \theta'_e \right\|^2$ is a secondary regularization term. λ and δ are hyperparameters.

The secondary regularization term aims at stabilizing the training by preventing θ_e from being too specifically optimized to fit the POS tagging task. Indeed, since θ_e is shared across all layers of the model, excessive modifications of its parameters would prevent the model from learning efficiently. θ'_e is the set of parameters involved in the word embedding layer at last epoch.

4.2 Chunking layer

We denote $\theta_{chunk} = (W_{chunk}, b_{chunk}, E_{POS}, \theta_e)$ the set of parameters involved in the chunking

layer. W_{chunk} et b_{chunk} are respectively parameter matrices and bias of the chunking layer, including those of θ_{POS}. E_{POS} is the set of parameters characterizing the POS label embeddings. The cost function is defined as follows:

$$J^{(2)} = -\sum_{s}\sum_{t} log\, p(y_t^{(2)} = k|h_t^{(2)}) \\ +\lambda \left\|W_{chunking}\right\|^2 + \delta \left\|\theta_{POS} - \theta'_{POS}\right\|^2$$

with $p(y_t^{(2)} = k|h_t^{(2)})$ the probability of assigning the right label k to the word w_t of the word sequence s. θ'_{POS} is the set of parameters of the POS tagging layer right before the training of the chunking layer for the current epoch.

4.3 ACD layer

We denote θ_{ACD} the set of parameters involved in the ACD layer, with $\theta_{ACD} = (W_{ACD}, b_{ACD}, E_{POS}, E_{chunk}, \theta_e)$. W_{ACD} and b_{ACD} are respectively parameter matrices and bias of the ACD layer, including those of the chunking and POS tagging layers. E_{chunk} is the set of parameters characterizing the chunk label embeddings. The cost is defined as follows:

$$J^{(3)} = -\sum_{d}\sum_{t} log\, p(y_t^{(3)} = k|h_t^{(3)}) \\ +\lambda \left\|W_{ACD}\right\|^2 + \delta \left\|\theta_{chunk} - \theta'_{chunk}\right\|^2$$

with $p(y_t^{(3)} = k|h_t^{(3)})$ the probability of assigning the right label k to the word w_t of the essay d. θ'_{chunk} is the set of parameters of the chunking layer right before the training of the ACD layer for the current epoch.

4.4 ACC layer

We denote θ_{ACC} the set of parameters involved in the ACC layer, with $\theta_{ACC} = (W_{ACC}, b_{ACC}, E_{POS}, E_{chunk}, \theta_e)$. W_{ACC} and b_{ACC} are respectively parameter matrices and bias of the ACC layer, including those of the chunking and POS tagging layers. The cost function is defined as follows:

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

with $p(y_i^{(4)} = k|sh_i^{(4)})$ is the probability of assigning the right label k to the segment s_i of the essay d.

5 Experiments and discussion

5.1 Hyperparameters and training corpora

5.1.1 Optimization

We trained the model alternating, for each epoch, the layers to be trained in the following order: POS tagging, chunking, ACD and ACC. We used Adam (Kingma and Ba, 2014) as learning algorithm, with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The learning rate was shared across all layers and was fixed to 10^{-3} at the beginning of the training, and then multiplied by 0.75 every 10 epochs. In order to limit the gradient exploding problem, we used a gradient clipping strategy (Pascanu et al., 2013). We followed (Hashimoto et al., 2016) and used a clipping value of $min(3.0, depth)$, where $depth$ stands for the number of bi-GRU involved in the trained layer.

5.1.2 Parameters initialization

In order to smooth the backpropagation of the gradient during the training phase, we used random orthogonal matrices as initial weights for parameter matrices of every GRU, as suggested by (Saxe et al., 2013). The remaining parameter matrices were initialized with values drawn from a gaussian $\mathcal{N}(0, \sqrt{\frac{2}{n_{in}}})$, with n_{in} being the number of input neurons in the layer, as proposed by (He et al., 2015). Bias vectors were initialized as zero vectors.

5.1.3 Vector dimensions used

We used 50 dimensional vectors for the words and labels embeddings. We used 100 dimensional vectors for the hidden states of every GRU in the model.

5.1.4 Regularization

Following (Hashimoto et al., 2016), we used $\lambda = 10^{-6}$ for the parameter matrices of every GRU and $\lambda = 10^{-5}$ for the remaining parameter matrices. The secondary regularization term rate δ was set to 10^{-2} for each layer. We also used Dropout (Srivastava et al., 2014) on every layer, with a probability to affect neurons set to 0.2.

5.1.5 POS tagging and chunking corpora

We used the corpora from the shared task CoNLL-2000 (Sang and Buchholz, 2000) and the associated labels to train the POS tagging and chunking layers.

5.1.6 ACD and ACC corpora

We used the Argument Annotated Essays (version 2) corpora released by (Stab and Gurevych, 2017) following the train/test split given to train the ACD and ACC layers.

5.1.7 Training termination criteria

For a mono-task model, a common practice is to stop the training right before overfitting. However, it is not clear to determine when to stop the training when dealing with multi-task models since it is possible that the model overfits only on a subset of the target tasks. Thus, we decided to stop the training when the model overfitted both on the ACD and the ACC tasks on a held-out validation set[2]. The reported results are the best we obtained for each task on the test set, before overfitting on the validation set. Note that hyperparameters can be chosen so that the model overfits roughly at the same time on the ACD and ACC tasks.

5.1.8 Simple ACC

We denote Simple ACC the ACC task with the following modification: every segment corresponding to an argument component was treated as a single special token <*EMPTY*>. We hypothesize that this transformation will prevent the model from focusing on words inside argument components, but rather on its context, thus allowing a better generalization process.

5.2 Results and discussion

We report the obtained results on the test data for the tasks ACC, ACD and Simple ACC in Table 1. The column "w/o POS & chunking" refers to the model version for which we omitted the training of the POS tagging and chunking layers, while the column "w/ POS & chunking" refers to the model version optimized for every tasks. As a baseline, we use the human performances[3] and the results reported by (Stab and Gurevych, 2017), shown in Table 2.

[2]We randomly sampled 10% essays from the training data to build the validation set.

[3]Human performance corresponds the average performances reached by human agents, as presented in (Stab and Gurevych, 2017).

Table 1: Macros f1-scores obtained on the ACC, ACD and Simple ACC tasks.

Task	w/o POS & chunking	w/ POS & chunking
ACD	0.5922	0.8870
ACC	0.6950	0.7257
Simple ACC	0.7670	0.7980

Table 2: F1-scores reported on the ACD and ACC tasks by Stab and Gurevych (Stab and Gurevych, 2017) and human agents.

Task	F1-score from (Stab and Gurevych, 2017)	Human f1-score
ACD	0.867	0.886
ACC	0.826	0.868

5.2.1 General performance discussion

We obtained a macro f1-score of 0.8870 on ACD with the "w/ POS & chunking" model version, reaching human performance. This result was obtained without using handcrafted features, and is comparable to the one reported in (Stab and Gurevych, 2017). Regarding the ACC task, we obtained a macro f1-score of 0.7980 with Simple ACC for the "w/ POS & chunking" version, representing 96.6 % of the performance obtained in (Stab and Gurevych, 2017) and 91.9% of the human performance.

5.2.2 Simple ACC assessment

We consider that the words composing argument components are not really relevant to determine if they are major claims, claims or premises. Hence, we hypothesize that focusing on those words will lead to model noise. Conversely, the context surrounding argument components should be a good indicator: sequences such as "we can receive the same conclusion that" seem to be strong indicators that the upcoming argument component is not a premise. This could explain the gap in performance obtained between ACC and Simple ACC, more particularly for the "w/ POS & chunking" version, with respective f1-scores of 0.7257 and 0.7980 (9.96% performance increase).

5.2.3 Multi-task framework assessment

Regarding the tasks ACD and Simple ACC, we obtained f1-scores of 0.5922 and 0.7670 for the

"w/o POS & chunking" version and of 0.8870 and 0.7980 for the "w/ POS & chunking", representing respectively a 49.78% and a 4.1% performance gain. Those results show the benefits of using a multi-task framework and suggests that more subtasks could be added to the model.

6 Upcoming work and perspectives

The results we got are encouraging and could probably be improved, particularly by analyzing optimal hyperparameters in a deeper way. The performance difference between the "w/ POS & chunking" and "w/o POS & chunking" models tends to show that implementing more auxiliary tasks could be beneficial. One exploration way could be to insert a dependency parsing layer on top of the chunking layer, as done in (Hashimoto et al., 2016).

In order to implement a complete argument mining system, as introduced by (Stab and Gurevych, 2017), we plan to implement layers which enable to automatically generate argument graphs. Therefore, it is necessary to determine if a directed edge exists between each ordered pair of argument components, and also to label those edges either as support or as attack relationships.

7 Conclusion

This article introduced the use of a novel model based on a multi-task framework for automatically extracting and classifying argument components from raw texts. Interestingly, our results show that the use of advanced deep learning techniques enables competing with state-of-the-art systems that depend on handcrafted features. The variation of performance between the model exploiting auxiliary tasks (POS tagging and chunking) and a version skipping those tasks clearly promotes the added-value of a multi-task framework.

References

Yamen Ajjour, Wei-Fan Chen, Johannes Kiesel, Henning Wachsmuth, and Benno Stein. 2017. Unit segmentation of argumentative texts. In *Proceedings of the 4th Workshop on Argument Mining*, pages 118–128.

Héctor Martínez Alonso and Barbara Plank. 2016. When is multitask learning effective? semantic sequence prediction under varying data conditions. *arXiv preprint arXiv:1612.02251*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303*.

Elena Cabrio and Serena Villata. 2018. Five years of argument mining: a data-driven analysis. In *IJCAI*, pages 5427–5433.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Judith Eckle-Kohler, Roland Kluge, and Iryna Gurevych. 2015. 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*, pages 2236–2242.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. *arXiv preprint arXiv:1704.06104*.

Theodosis Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. 2014. Argument extraction from news, blogs, and social media. In *Hellenic Conference on Artificial Intelligence*, pages 287–299. Springer.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10.

Nitin Madnani, Michael Heilman, Joel Tetreault, and Martin Chodorow. 2012. 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*, pages 20–28. Association for Computational Linguistics.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? *arXiv preprint arXiv:1603.06111*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the first workshop on argumentation mining*, pages 29–38.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Isaac Persing and Vincent Ng. 2016. 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*, pages 1384–1394.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2016. Here's my point: Joint pointer architecture for argument mining. *arXiv preprint arXiv:1612.08994*.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Erik F Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. *arXiv preprint cs/0009008*.

Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. 2015. Argument extraction from news. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 56–66.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multi-task learning for argumentation mining in low-resource settings. *arXiv preprint arXiv:1804.04083*.

Anders Søgaard and Yoav Goldberg. 2016. 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)*, pages 231–235.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.