Text Segmentation Architecture

Mattia Bertè* **Simone Giordani*** **Omar G. Younis***
{mattia.berte, simone.giordani2, omargallal.younis}@studio.unibo.it

1 Executive summary

Less than a third of high school seniors in the U.S. are proficient writers, according to the National Assessment of Educational Progress[1]. Low-income, Black, and Hispanic students fare even worse, with less than 15 percent demonstrating writing proficiency. Since good writing skills are important for success, unbiased support for writing could have a positive impact on society. One way to achieve this is via automated feedback tools, which often require a pipeline of tools to be deployed. We decided to contribute to the cause, building and releasing a model for text segmentation and argumentative element classification. Our model is an end-to-end neural network based on Transformers, capable of identifying the argumentative elements of a text and classifying them. We address the problem, proposing a new architecture inspired by successful ones for object detection in Computer Vision. Regarding the dataset, we used the one provided by the Feedback Prize competition on Kaggle[2]; we will describe it in Section 3.1.

2 Background

Text Segmentation is an underexplored area of research. The goal of this task is to identify argumentative elements from a document. Closely related is the *Argument Mining* task, from which we take inspiration. In the latter, the aim is also to extract a structure of argumentative elements, while in our task this is not necessary (even if an understanding of the structure of the text can help achieve better performances). One notable reference is (Ruiz-Dolz et al., 2021) which shows the effectiveness of transformer-based architectures for the Argument Mining tasks. (Galassi et al., 2021) proposed RESATTARG, a state-of-the-art architecture for Augmentative Mining. The main limitation is that it doesn't scale up well with large documents. We aim to overcome this issue using Longformer, (Beltagy et al., 2020).

Longformer are a particular type of transformer proposed to scale up for long documents. To do so, they use two types of attention, a local and a global one. The local attention is standard sliding-window attention, while the global one is a sparse pattern to attend to all the documents. With this trick, the scales linearly with the size of the sequence to process, making it suitable for our task. However, choosing the global attention pattern is a new implementation choice that must be tuned.

DETR is a state-of-the-art architecture for object detection proposed by (Carion et al., 2020). In a nutshell, it involves a backbone, typically a ResNet50 or a ResNet101 (He et al., 2015), that extracts the important feature of the image. These features are passed to an encoder-decoder transformer that extracts the box predictions. In its simplicity, DETR doesn't require a complex and hand-crafted pipeline for box extraction and it is highly parallelizable. However, it requires setting a maximum number of box predictions. For training, DETR uses two different losses. The first one is the cross-entropy loss for the classification of the boxes. The second one is a linear combination of the L_1 norm between the predicted center and length of the bounding boxes and the ground truth plus the Generalized Intersection-Over-Union between the true and the predicted boxes.

[1] https://nces.ed.gov/nationsreportcard/writing/
[2] https://www.kaggle.com/competitions/feedback-prize-2021

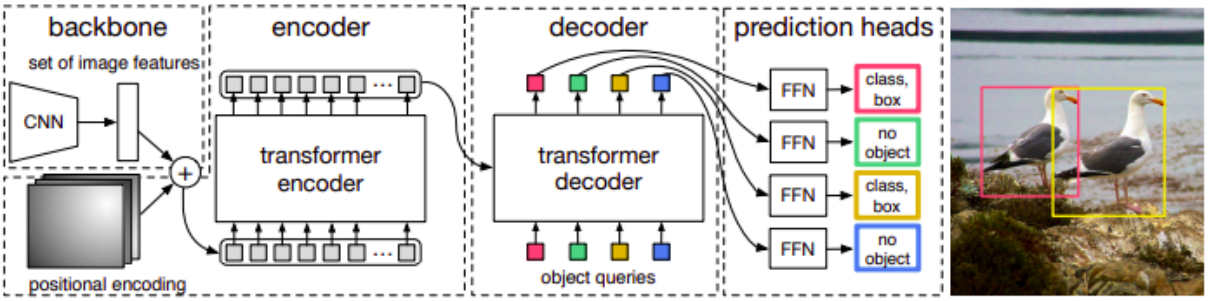


Figure 1: DETR architecture representation. The model predicts a fixed number of boxes and then classifies them with a special class to discard boxes.

To match the boxes with the ground truth, (Carion et al., 2020) use the Hungarian algorithm. So the overall loss is:

$$\mathcal{L}_{Hungarian}(y, \hat{y}) = \sum_{i=0}^{N} \left[-\lambda_{CE} \log p_{\hat{\sigma}(i)(c_i)} + 1_{\{c_i \neq \emptyset\}} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right] \tag{1}$$

where λ_{CE} is a hyperparameter and $\hat{\sigma}$ is the optimal assignment computed as:

$$\hat{\sigma} = \underset{\sigma \in \Theta}{\operatorname{argmin}} \sum_{i}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma_i}) \tag{2}$$

and $\mathcal{L}_{box}$ is:

$$\mathcal{L}_{box}(b_i, \hat{b}_{\sigma_i}) = \lambda_{iou} \mathcal{L}_{iou}(b_i, \hat{b}_{\sigma_i}) + \lambda_{L1} ||(b_i, \hat{b}_{\sigma_i})||_1 \tag{3}$$

with λ_{iou} and λ_{L1} hyperparameter.

3 System description

3.1 Dataset

The dataset contains more than 15k argumentative essays written by U.S students in grades 6-12. The essays were annotated by expert raters for elements commonly found in argumentative writing. As always in kaggle competition, the test set is private, you have access to a certain amount of evaluations of your model, however we decided to create also the test set from the whole dataset. Doing so we could analyze the error of our model on the test set, which wouldn't be possible otherwise.

As shown in figure 2, the documents have a quite spread distribution of lengths, with the majority having less than 1000 words. Instead, the number of argumentative units in each document follows a Gaussian distribution, as shown in figure 3.

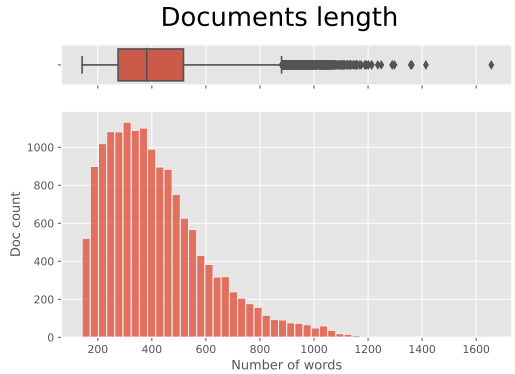


Figure 2: Number of words in each document

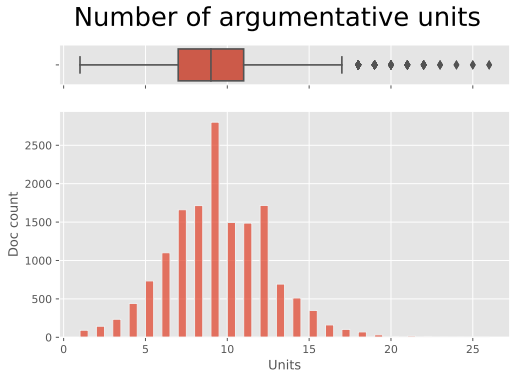


Figure 3: Number of argumentative units for each document

3.1.1 Annotations

The annotations report the following information:

- *discourse_start* - character position where discourse element begins in the essay response
- *discourse_end* - character position where discourse element ends in the essay response
- *discourse_type* - classification of discourse element (see 3.1.2)

Those kinds of annotation are similar to the ones commonly found in object detection tasks. In fact, we can make a parallelism between pixel position in the images of bounding boxes, and character position in the document of the discourse units (see section 3.2.1 for details). The major difference is that here the segmentation boxes are 1-dimensional, instead of the 2-D bounding boxes in image object detection. So we augment the annotations by including two new features:

- *box_center* - character position of the center of the discourse
- *box_length* - number of characters in the discourse

Those are the features that our model will train on and predict.

3.1.2 Classes

The discourses are classified using the following categorization:

- **Lead** - an introduction that begins with a statistic, a quotation, a description, or some other device to grab the reader's attention and point toward the thesis
- **Position** - an opinion or conclusion on the main question
- **Claim** - a claim that supports the position
- **Counterclaim** - a claim that refutes another claim or gives an opposing reason to the position
- **Rebuttal** - a claim that refutes a counterclaim
- **Evidence** - ideas or examples that support claims, counterclaims, or rebuttals.
- **Concluding Statement** - a concluding statement that restates the claims

Some parts of the essays will be unannotated (i.e., they do not fit into one of the classifications above).

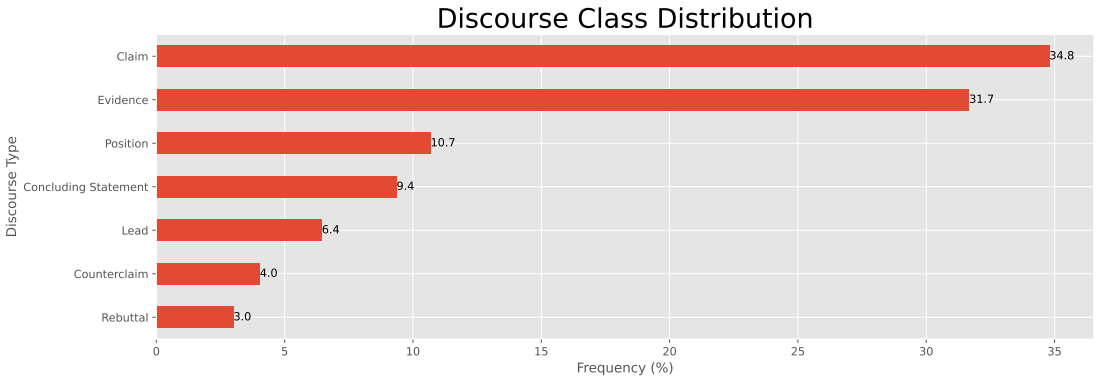


Figure 4: Distribution of the classes

As shown in figure 4 The classes *Claim* and *Evidence* are by far the most common in the dataset

In figure 5 we can see that on average the documents respect the structure of the argumentative essay. They start with the lead section, to introduce the thesis, followed by positions, claims and evidence to support the thesis. Then comes the counterclaims and the rebuttals, that are needed to give more credibility to the thesis, and then they end with the concluding statements. The central sections have a large standard deviation, as they can be mixed and even repeated multiple times in an essay. As we can expect, the larger sections are the Lead, the Concluding statements and the Evidence because they are the main parts of an argumentative essay, instead, the other sections are just needed to introduce the discussion.

We also evaluated how the words correlate with the classes. As correlation metrics, we take inspiration from the Tf-idf statistic, which is intended to reflect how important a word is to a document in a collection

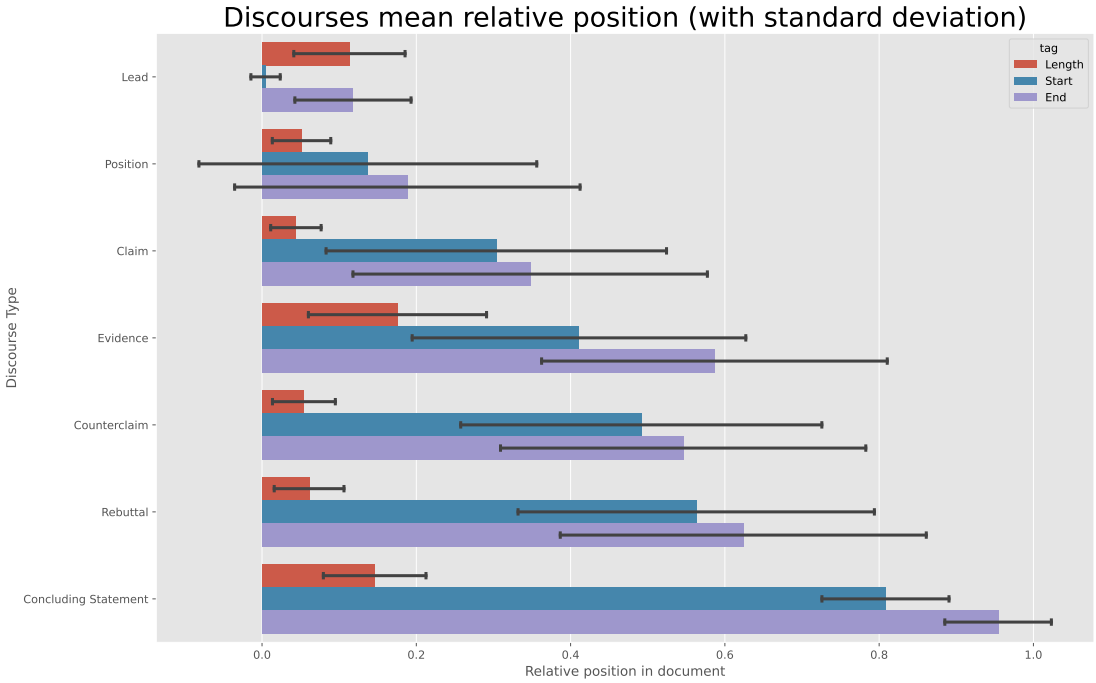


Figure 5: Average start position, end position and length (in characters) normalized with respect to the document length (with standard deviations)

or corpus. We evaluated the term frequency in the class as

$$\mathrm{tf_{c,t}} = \frac{f_{c,t}}{\sum_t' f_{c,t'}}$$

with $f_{c,t}$ as the counts of the occurrence of the term t in the class c. Then instead of the inverse-document-frequency term, we weighted the frequencies with an inverse-discourse-frequency in as

$$\mathrm{idf_t} = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

with D the set of all discourses in the dataset.

Some interesting correlations emerge. For example, in the *Counterclaim* class, the most correlated are *argue, say, although, may, might*, which are all words that introduce a hypothetical opposite opinion. In *Position* we can find words like *believe, idea, agree*, instead the top correlation for *Rebuttal* and *Concluding Statement* are respectively *however* and *conclusion*, as one might expect. Instead *Claim*, *Evidence* and *Lead* correlate with more generic words, maybe because in those classes there is more freedom in the content.

3.2 Architecture

Starting from the DETR proposal, we tried to adapt this architecture to our task. Even if CNNs still represent the SoTA in some computer vision tasks, they don't work so well in NLP tasks, where transformers became the dominant paradigm in the last years. Following these premises, we decide to replace the ResNet feature extractor in favour of an advanced version of transformers called Longformer. Actually, the Longformer architecture is nothing different from the BERT model, the innovative idea is related to the attention mechanism. The original Transformer model has a self-attention component with $O(n^2)$ time and memory complexity where n is the input sequence length, the Longformer model instead tries

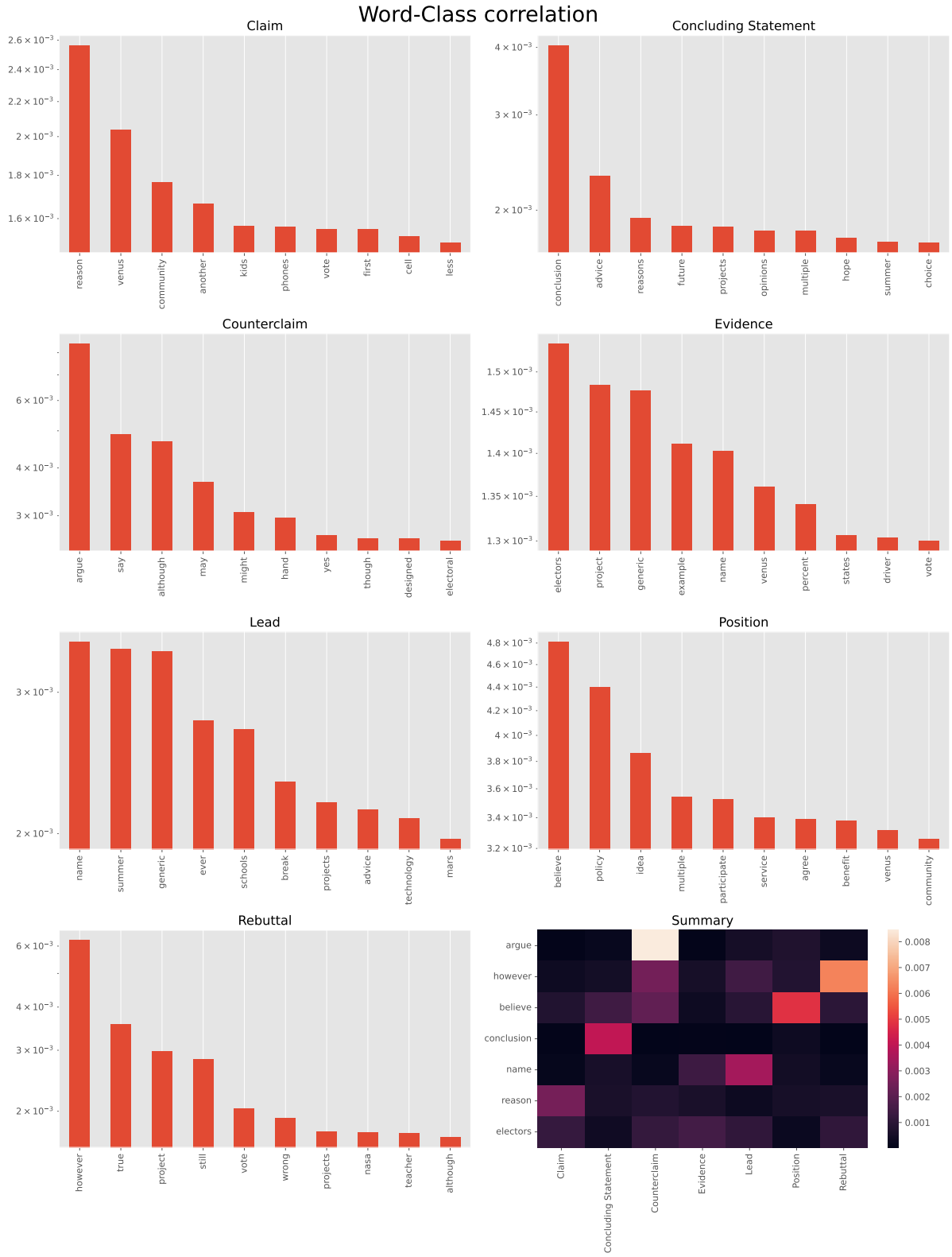
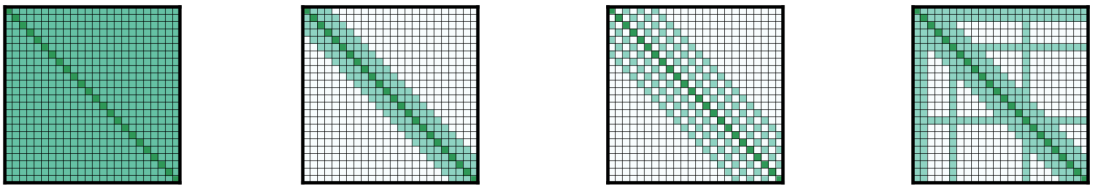


Figure 6: Correlation between words and classes.

to combine local attention with a sliding windows approach, whose complexity is $O(n)$, plus global attention related to preselected input locations in order to provide inductive bias to the model, since the number of these inputs is small related to the length of the text, the model complexity is still $O(n)$. Since our task requires dealing with long texts, in which each sentence has to be analyzed with respect

to the whole text, Longformer is a good trade-off. Following the same intuition, since the Longformer is present in the encoder-decoder variant, we decided to use it also for the second part of the architecture.



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 7: Comparison between Longformer attention mechanism and previous proposal.

3.2.1 From Object Detection to Text Segmentation

As said before we have made parallelism between object detection and text segmentation. The main difference is that in object detection the boxes are 2-D, instead in text segmentation they are 1-D. The regression head of DETR predicts 4 parameters that are the center and the dimensions of the bounding boxes, normalized in the range $(0, 1)$ with respect to the dimensions of the image. We changed it to predict only 2 parameters, the center and the length of the "segmentation boxes". We choose to consider those values normalized with respect to the position of the words in the document tokenized by the LongormerTokenizer, by huggingface[3], that we used during training.

With this in mind, we adapted all the operations that should have involved the 2-D bounding boxes, in order to make them work with our 1-dimensional "boxes". The main function to modify is the $\mathcal{L}_{iou}$ which evaluates the Intersection Over Union between bounding boxes, and it's used both in the Hungarian Matcher and in the Loss function. In particular, we used the Generalized Intersection Over Union, introduced by (Rezatofighi et al., 2019), that overcomes the issues of the plateau that IoU has in the case of nonoverlapping bounding boxes. GIoU in the general case is defined as follows: for two arbitrary convex shapes (volumes) $A, B \subseteq \mathbb{S} \in \mathbb{R}^n$, find the smallest convex shapes $C \subseteq \mathbb{S} \in \mathbb{R}^n$ enclosing both A and B. Then calculate a ratio between the volume (area) occupied by C excluding A and B and divide by the total volume (area) occupied by C. This represents a normalized measure that focuses on the empty volume (area) between A and B. Finally, GIoU is attained by subtracting this ratio from the IoU value. In summary:

$$GIoU = \frac{|A \cap B|}{|A \cup B|} - \frac{|C \setminus A \cup B|}{|C|}$$

$GIoU$ is both a metric and a loss, with $\mathcal{L}_{GIoU}(A, B) = 1 - GIoU(A, B)$. (Rezatofighi et al., 2019) show a consistent improvement in performance measures on popular object detection benchmarks by incorporating this generalized IoU as a loss into the state-of-the-art object detection frameworks.

To adapt this loss to our task, we considered to have boxes in the start-end format $b_i = (s_i, e_i)$ obtained from the predictions in format center-length with the equations $s_i = c_i - \frac{l_i}{2}$ and $e_i = c_i + \frac{l_i}{2}$ so it's guaranteed that $s_i \leq e_i$. We evaluate the $GIoU$ as follows:

1. $|b_i| = l_i = s_i - e_i$ - The area of the boxes is their length
2. $|b_i \cap b_j| = \max(0, \max(s_i, s_j) - \min(e_i, e_j))$ - The intersection of boxes is the length of the segment they have in common
3. $|b_i \cup b_j| = |b_i| + |b_j| - |b_i \cap b_j|$
4. $|C| = \min(s_i, s_j) - \max(e_i, e_j)$
5. $|C \setminus b_i \cup b_j| = |C| - |b_i \cup b_j|$ - The length of the segment that, if any, separates b_i and b_j. With perfect overlap, this factor is equal to 0

Since every token can belong to at most one argumentative part, we also experiment with a new loss $\mathcal{L}_{Overlap}$ that penalizes overlapping between the boxes issued by the network. Thus this loss is simply the mean of the *GIoU* between every couple of boxes issued by the network.

[3] https://huggingface.co/docs/transformers/model_doc/longformer#transformers.LongformerTokenizerFast

3.3 Evaluation metrics

To evaluate our model we used the metric described in the competition web page [4]. These metrics evaluate the overlap between ground truth and predicted word indices.

For each sample, all ground truths and predictions for a given class are compared. If the overlap between the ground truth and prediction is ≥ 0.5, and the overlap between the prediction and the ground truth ≥ 0.5, the prediction is a match and is considered a true positive. If multiple matches exist, the match with the highest pair of overlaps is taken. Any unmatched ground truths are false negatives and any unmatched predictions are false positives.

The final score is arrived at by calculating TP/FP/FN for each class, then taking the macro F1 score across all classes.

Annotations and predictions must be in word indices, that is calculated by using Python's `.split()` function and taking the indices of the words of the discourse unit in the resulting list. The two overlaps are calculated by taking the `set()` of each list of indices in a ground truth/prediction pair and calculating the intersection between the two sets divided by the length of each set.

The idea is similar to the Mean Average Precision (mAP) used commonly in object detection tasks, for example to evaluate the accuracy in the COCO dataset [5], where a threshold on the Intersection Over Union (IOU) is used to evaluate the matches between ground truth and predicted bounding boxes.

For example, if for the class *Claim* in a document there are ground truth annotations

$$[(1, 2, 3, 4, 5), (6, 7, 8), (21, 22, 23, 24, 25)]$$

and the model predicts

$$[(1, 2), (6, 7, 8)]$$

then:

- The first prediction would not have ≥ 0.5 overlap with either ground truth and would be a false positive.
- The second prediction would overlap perfectly with the second ground truth and be a true positive.
- The third ground truth would be unmatched, and would be a false negative.

4 Experimental setup

We trained our networks on a Tesla V100. The training takes about 30 min per epochs. We trained around 50 networks for tuning purposes, with an average of 2 epochs each, resulting in roughly 50 hours of training. We estimated the CO_2 emission of about 6.5 KG[6]. We decided to compensate it by planting an Avocado tree through Treedom [7], absorbing 500KG of CO_2 in 10 years.

We experimented multiple times varying the setup to understand the best way for training this model. For tackling the class imbalance problem, we implemented the focal loss technique (Lin et al., 2017). Then we tried also different initialization of weights and bias for the final layers and also for the encoder-decoder model when non-pretrained weights were used. We experimented also with fine-tuning of the features extractor backbone and with it frozen. As optimizer, we used *AdamW* and we dropped the learning rates for the second epochs by a factor of ten. We report the best hyperparameters that we found (Table 1):

heads lr	**transf. lr**	**hidden dim**	**num queries**	λ_{CE}	λ_{L1}	λ_{GIoU}	$\lambda_{Overlap}$	γ_{FL}
10^{-4}	10^{-5}	2048	40	1	1	0.5	0.5	2

Table 1: Table containing the best hyperparameters.

where λs are the coefficient of the diverse components of the loss as indicated by the subscript and γ_{FL} is the modulating term of the cross entropy (focal loss). For the best model, we also set the depth

[4] https://www.kaggle.com/competitions/feedback-prize-2021/overview/evaluation
[5] https://cocodataset.org/#detection-eval
[6] https://mlco2.github.io/impact/
[7] https://www.treedom.net/

for the classification head and the bounding box head to 3 and 5 layers respectively. We also initialize the bias of the last classification layer in order to match the prior distribution of classes. Instead, we don't apply weight decay nor dropout, since our model doesn't overfit the training data. We didn't see any benefit in applying or not gradient clipping. The model has 235M parameters and was trained for 2 epochs since training for more time doesn't improve its performance. In order to exploit the information obtained from the data set analysis, we decided to define a new strategy for the global attention. For each class we detected the most frequent word (as described in section 3.1.2) and associated with them the token of the global attention, to impose the network a certain focus on particularly relevant words (Table 2).

Class	Most freq. word	Class	Most freq. word
Claim	Reason	Lead	Name
Concluding statement	Conclusion	Position	Believe
Counterclaim	Argue	Rebuttal	However
Evidence	Electors		

Table 2: Each class with its correspondent most frequent word.

As can be seen, in some cases we can notice a relation between the meaning of a word and its classes, while in other cases we have words whose presence is strictly dependent on our data set (Lead/Name, Evidence/Electors). However for sake of fairness, we selected one word for each class. The model architecture used for this experiment is the same presented in Table 1

5 Results

As baseline we thought to use a simple detector which predict always the same class, 'Claim' which is the most frequent and detect an argumentative element every 151 words which is the median length. It can give a qualitative information on the hardness of the task and it is useful to show how we were able to improve this simplest classifier. For a standard classifier defining a simple baseline it's easy, it's not the same for an object detection problem, indeed the baseline we proposed isn't really informative.

Class	Precision	Recall	F1-score
Lead	0.000	0.000	0.000
Position	0.000	0.000	0.000
Evidence	0.000	0.000	0.000
Claim	0.001	0.001	0.001
Concluding Statement	0.000	0.000	0.000
Counterclaim	0.000	0.000	0.000
Rebuttal	0.000	0.000	0.000
Macro Avg	0.0001	0.0002	0.0002

Table 3: Scores of the baseline model.

Adding the global attention to particular words present a general improvement ($+6\%$ in f1-score). However we can notice a drop in performances related to the recall while an increasing in the precision. This probably means that the network lose a bit in the capability of classification but improve its performances in the identification of the position of the argumentative units. This technique could be in general a good idea, however as described in sec 4, some of the words we used don't present any real correlation with their class. So it is highly probable that in a different scenario using this words could lead to a poor results. The way in which we selected them is too task dependant, another possibility would be to use a list of words provided by a team of experts in order to have a more general capability.

Class	Precision	Recall	F1-score
Lead	0.428	0.721	0.537
Position	0.152	0.154	0.153
Evidence	0.516	0.561	0.538
Claim	0.129	0.311	0.183
Concluding Statement	0.088	0.919	0.161
Counterclaim	0.015	0.141	0.026
Rebuttal	0.014	0.196	0.027
Macro Avg	0.192	0.429	0.232

Table 4: Scores of the best model without global attention on most frequent words for each class.

Class	Precision	Recall	F1-score
Lead	0.438	0.715	0.543
Position	0.179	0.389	0.246
Evidence	0.371	0.661	0.476
Claim	0.119	0.256	0.163
Concluding Statement	0.535	0.615	0.572
Counterclaim	0.015	0.068	0.025
Rebuttal	0.010	0.066	0.017
Macro Avg	0.238	0.396	0.292

Table 5: Scores of the best model with global attention on most frequent words for each class.

6 Error analysis

One of the most common error and easiest to notice is the incapability of the model to understand which are the most common words or symbols that define the beginning and the end of a sentence, such as periods, commas or conjunction which usually are very helpful also for humans for understanding which kind of argumentative unit we can expect. A possible way to influence the model towards this direction, could be to use the global attention on all the conjunction. On one hand, we have that this approach would be very informative for the model since it would mimick also the way in which human annotators establish the groundtruth, on the other hand there would be the risk of increasing too much the computational complexity. Another common error is the overlapping between different detection, in some cases it is related just to consecutive detection, in other cases instead it is due to multiple detection of the same unit, this problem is pretty common in object detection task. A possible way to overcome this problem, would be to adopt technique such as non-maxima suppression, which consist in discarding all the detection which overlap more than a threshold and keeping just the one with the greatest score. Clearly also this approach doesn't come for free, it increases the computational complexity and moreover it takes in account only the best score, without any consideration for the rest. Below you can see an example the perfectly summarize the errors described above.

participate seagoing cowboys once. People might not have many reasons to support there statement about 'Why you [Claim]

participate seagoing cowboys once. People might not have many reasons to support there statement about 'Why you shouldn't be a Seagoing Cowboy', but I have at least some reasons as why you should join [Evidence]

In the end, another big difference between the object detection task and ours, is related to the proportion foreground/background and Argumentative/Non-Argumentative units. In the first case is common to have small number of object with a lot of background while in pour case, working with argumentative

essay, more or less all the proposition have to be classified.

7 Discussion

Even before starting, we were conscious that trying to tackle such a complex task as Argumentative Mining using a completely different approach, in particular one derived from object detection, would have been an hard challenge. However we were curious to explore new path and propose a different point of view for this kind of problems. It gave us the opportunity to work and understand not only the way in which transformers work, but also how they can be optimized for handling long texts, which by the way can be very useful for argumentative essays. This problem was very stimulating and challenging. Even if it's not new as described in section 2, the fact that doesn't exist a common solution which is able to perform well in any situation, urge us to analyze in deep the data set to find different way to improve our initial solution. Of course we wouldn't expect to tie the results of the top teams of the competition, we were more interested in proposing something new, or at least not so spread as more standard approaches. We hope it could offer a new perspective to Argumentative task and maybe to find a direction to merge CV and NLP with model capable of both tasks at the same time.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers.

Andrea Galassi, Marco Lippi, and Paolo Torroni. 2021. Multi-task attentive residual networks for argument mining.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. 2017. Focal loss for dense object detection.

Seyed Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. 2019. Generalized intersection over union: A metric and A loss for bounding box regression. *CoRR*, abs/1902.09630.

Ramon Ruiz-Dolz, Jose Alemany, Stella M. Heras Barbera, and Ana Garcia-Fornes. 2021. Transformer-based models for automatic identification of argument relations: A cross-domain evaluation. *IEEE Intelligent Systems*, 36(6):62–70, nov.