RKadiyala at SemEval-2024 Task 8: Black-box Word-level text boundary detection in partially machine generated texts

Ram Mohan Rao Kadiyala
University of Maryland , College Park
rkadiyal@terpmail.umd.edu

Abstract

With increasing usage of generative models for text generation and widespread use of machine generated texts in various domains, being able to distinguish between human written and machine generated texts is a significant challenge. While existing models and proprietary systems focus on identifying whether given text is entirely human written or entirely machine generated, few systems provide insights at sentence or paragraph level at likelihood of being machine generated at a non reliable accuracy level. This paper introduces a novel approach for identifying which part(s) of a given text are machine generated at a word level while comparing results from different approaches and methods. We present a comparison with proprietary systems , performance of our model on unseen domains' and generators' texts. The finding reveal significant improvements in detection accuracy and comparison on other aspects of detection capabilities. Finally we discuss potential avenues for improvement and implications of our work.

1 Introduction

With rapid advancements and usage of AI models for text generation , being able to distinguish machine generated texts from human generated texts is gaining importance. While existing models and proprietary systems like GLTR (Gehrmann et al., 2019), ZeroGPT (ZeroGPT), GPTZero (Tian and Cui, 2023), GPTKit (GptKit), Open AI detector , etc.. focus on detecting whether a given text is entirely AI written or entirely human written , there was less advancement in detecting which parts of a given text are AI written in a partially machine generated text. While some of the above mentioned systems provide insights into which parts of the given text are likely AI generated , these are often found to be unreliable and having an accuracy close or worse than random guessing. There is also a rise in usage of AI to spread fake news and misinformation along with using AI models to modify wikepedia articles (Vice, 2023). Our proposed model focuses solely on detecting word level text boundary in partially machine generated texts. This paper also discusses implications of findings , comparisons with different models and approaches , comparision with existing proprietary systems with relevant metrics , other findings regarding AI generated texts. The official submission is DeBERTa-CRF , several other models have been tested for comparison. With new, better, and diverse AI models coming into existence, having a model that can make accurate predictions on unseen domains and unseen generator texts can be useful for practical scenarios.

2 Dataset

The dataset used is part of M4 Dataset(Wang et al., 2023) consisting of texts each of which are partially human written and partially machine generated sourced from peerread reviews and outfox student essays (Koike et al., 2023) all of which are in English. The generators used were GPT-4 , ChatGPT , LLaMA2 7/13/70B (Touvron et al., 2023). Table 1 shows the source , generator used and data split of the dataset.

Set	Count	Sources	Generators
Train	3649	PeerRead	ChatGPT
Dev	505	PeerRead	ChatGPT
Test	11123	PeerRead	LLaMA2 7/13/70B
		OUTFOX	LLaMA2 7/13/70B
		OUTFOX	GPT-4

Table 1: Dataset sources and split

3 Baseline

The provided baseline uses finetuned longformer over 10 epochs. The baseline classifies tokens individually as human or machine generated and then

maps the tokens to words to identify the text boundary between machine generated and human written texts. The final predictions are the labels of words after whom the text boundary exists. The detection criteria is first change from 0 to 1 or vice versa. We have tried one more approach by considering the change only if consecutive tokens are the same. The baseline model achieved an MAE of 3.53 on the Development set which consists of same source and generator as the training data. The model had an MAE of 21.535 on the test set which consists of unseen domains and generators.

4 Proposed Model

We have built several models out of which DeBERTa-CRF was used as the official submission. We have finetuned DeBERTa(He et al., 2023), SpanBERT(Joshi et al., 2020), Longformer(Beltagy et al., 2020), Longformer-pos (longfomer with training only on position embeddings), each of them again along with Conditional Random Fields (CRF)(McCallum, 2012) with different text boundary identification logic by training on just the training dataset and after hyperparameter tuning , the predictions have been made on both development and test sets. The primary metric used was Mean Average Error of predicted word labels of the text boundaries. Some of the plots and information couldn't be added due to page limits and are availible in the `documentation website`. [1] along with the `code used`. [2]. a hypothetical example in Figure 1 demonstrates how the model works. The tokens are classified at first and mapped to words. In cases where part of a word is classified at human and rest as machine (incase of longer words), the word as a whole is classified as machine generated.

4.1 Our system

We have used 'deberta-v3-base' along with CRF using Adam(Kingma and Ba, 2017) optimizer over 30 epochs with a learning rate of 2e-5 and a weight decay of 1e-2 to prevent overfitting. other models that have been used are 'Spanbert-base-cased', 'longformer-base-4096', 'longformer-base-4096-extra.pos.embd.only' which is similar to Longformer but pretrained to preserve and freeze weights from RoBERTa(Liu et al., 2019) and train

[1]more information available at : `https:www.rkadiyala.com/papers`

[2]Code availible at : `https://github.com/1024-m/NAACL-2024-SemEval-TASK-8C`

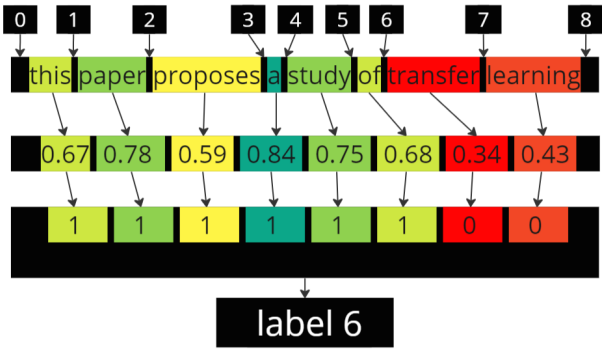

Figure 1: A visual example of working of the model

on only the position embeddings. The large variants of these have also been tested however the base variants have achieved better performance on both the development and testing datasets. predictions have been made on both the development and testing datasets by training on just the training dataset. Two approaches were used when detecting text boundary 1) looking for changes in token predictions i.e from 1 to 0 or 0 to 1, 2) looking for change to consecutive tokens i.e 1 to 0,0 or 0 to 1,1

4.2 Results

The results from using different models with the two approaches on the development set can be seen in Table 2. These models have been trained over 30 epochs and the best results were added among the several attempts with varying hyperparameters. while the provided baseline has been trained on 10 epochs using longformer. These models have

approach →	**I**	**II**
Model ↓	**MAE**	
DeBERTa	3.217	3.174
DeBERTa-CRF	**2.311**	**2.192**
SpanBERT	6.593	5.918
SpanBERT-CRF	4.855	4.519
Longformer	3.52	2.878
Longformer-CRF	2.782	2.41
Longformer.pos	3.296	3.075
Longformer.pos-CRF	2.613	2.406
Longformer (baseline)	3.53	3.287

Table 2: Performance of different models and approaches on development set

then been used to make predictions on the test set without further training or changes using the set of hyperparameters that produced the best results for each on the development set, these results can be

seen in Table 3

approach →	I	II
Model ↓	MAE	
DeBERTa	22.031	19.347
DeBERTa-CRF	20.074	**18.538**
SpanBERT	28.406	25.229
SpanBERT-CRF	24.283	20.97
Longformer	27.985	23.177
Longformer-CRF	20.941	18.943
Longformer.pos	23.219	19.502
Longformer.pos-CRF	20.223	**18.542**
baseline	21.535	19.898

Table 3: Performance of different models and approaches on test set

5 Comparison with proprietary systems

Some of the proprietary systems built for the purpose of detecting machine generated text provide insights into what parts of the text input is likely machine generated at a sentence / paragraph level. Many of the popular systems like GPTZero, GPTkit, etc.. are found to to less reliable for the task of detecting text boundary in partially machine generated texts. Of the existing models only ZeroGPT was found to produce a reliable level of accuracy. For the purpose of accurate comparision percentage accuracy of classifying each sentence as human / machine generated is used as ZeroGPT does detection at a sentence level.

5.1 Results comparision

Since the comparison is being done at a sentence level, In cases where actual boundary lies inside the sentence, calculation of metrics is done on the remaining sentences, and when actual boundary is at the start of a sentence , all sentences were taken into consideration. With regard to predictions, A sentence prediction is deemed correct only when a sentence that is entirely human written is predicted as completely human written and vice versa. The two metrics used were average sentence accuracy which is average of percentage of sentences correctly calculated in each input text, and overall sentence accuracy which is percentage of sentences in the entire dataset accurately classified. The results on the development and test sets are as shown in Table 4. Since ZeroGPT's API doesnt cover sentece level predictions , they have been manually calculated over the development set and can be found here. [3]. Since its difficult to do the same on 12000 items of the test set , a small section of 500 random samples were used for comparision and were found to perform similar to the development set with a 15-20 percent lower accuracy than the proposed models.

Dev set		
Model	**Accuracy**	**Avg. Accuracy**
DeBERTa-CRF	**0.9883**	**0.9848**
Longformer.pos-CRF	0.9806	0.9778
ZeroGPT	0.8086	0.7976
Test set		
Model	**Accuracy**	**Avg. Accuracy**
DeBERTa-CRF	**0.9969**	**0.9974**
Longformer.pos-CRF	0.9889	0.9901

Table 4: Performance at sentence level across Development and Test Sets

6 Conclusion

The metrics from Table 4 demonstrate the proposed model's performance on both seen domain and generator data (dev set) along with unseen domain and generator data (test set) , hinting at wider applicability. While there was a drop in accuracy at a word level, there was an increase in sentence level accuracy.

6.1 Strenghts and Weaknesses

It was observed that the proprietary systems used for comparison struggled with shorter texts. i.e when the input text has fewer sentences, the predictions were either that the input text is fully human written or fully machine generated leading to comparatively low accuracy. The average accuracy of sentence level classification for each text length of our model and ZeroGPT can be seen in Figure 2 , Figure 3 respectively. the proposed model overcomes this issue by providing more accurate results even on short text inputs. The sentence level accuracy did vary considerably while comparing cases where the actual text boundary is at the end of sentence and those where it is mid sentence. The results can be seen in Table 5.

[3]ZeroGPT annotations available at : https://docs.google.com/spreadsheets/d/1DOgAZBWQ3G6JtslQwgg9tJiX1WyZt0ajMrr2I9-yfHU/edit?usp=sharing

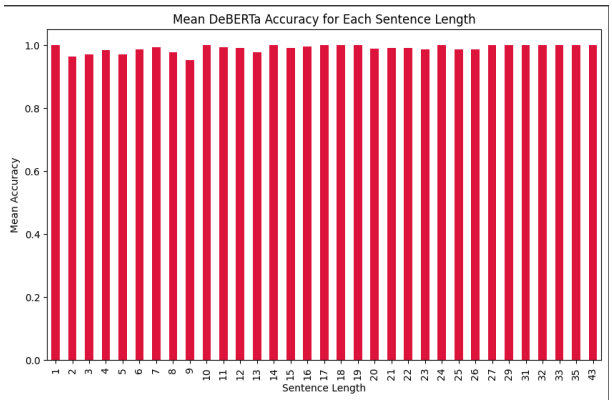

Figure 2: Average sentence accuracy VS number of sentences in given text : DeBERTa-CRF

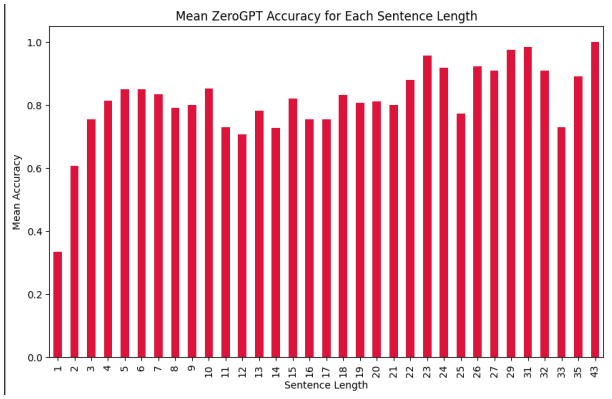

Figure 3: Average sentence accuracy VS number of sentences in given text : ZeroGPT

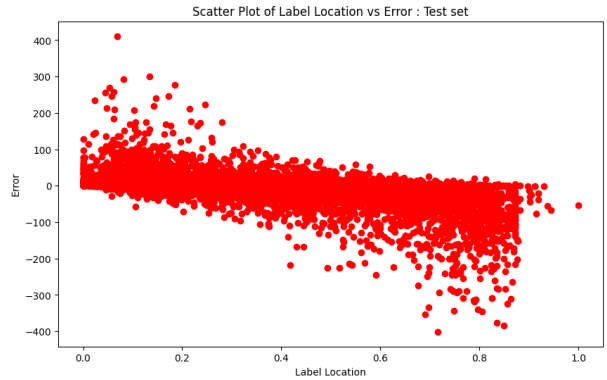

Figure 4: Text boundary location VS MAE : DeBERTa

6.2 Possible Improvements

DeBERTa performed better when text boundaries are in the first half of the given text, while longformer had better performance when the text boundary is in the other half. The cases where there was a significantly bigger MAE , atleast one of two (DeBERTa and Longformer) had made a very close prediction. There is a possibility that an ensemble of both might have a better performance. Further,

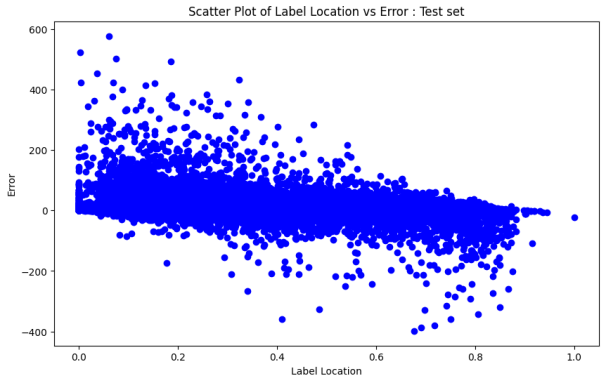

Figure 5: Text boundary location VS MAE : Longformer

Model ↓	mid sent..	end of sent..
DeBERTa-CRF	0.9835	**0.9972**
Longformer.pos-CRF	0.9765	**0.9901**
ZeroGPT	0.7942	**0.8296**

Table 5: Performance of models based on text boundary placement

the POS tags of the words pre and post text boundary were examined to findout what led to some cases having higher MAE. Figure 6 and Figure 7 display the count of data samples in train set and median MAE of those in test set for each POS tags combination pre and post split. The cases where the median MAE was higher (i.e 30 or above) had none or very few samples in the training set. Excluding those cases the new MAE was less than half of what it previously was. Adding more data that covers all cases of pre-split and post-split POS tag words might lead to better results. At a sentence level the accuracy was close to 100 percent excluding the above mentioned samples.

6.3 Possible Extensions and Applications

The need to detect AI generated content is also prevailent over other languages too. While the current model utilizes just the english language data, gathering multilingual data and having a multilingual model might also be of great use. With the growth of misinformation and fake news using bots on social media handles(Zellers et al., 2019), being able to detect AI generated texts is of great importance. As most of the texts i.e posts , comments etc.. are shorter in length and difficult to detect, An extension of the current work by training on social media data may yield a good result as demonstarted in Figure 2 and Figure 3. Some of the other findings are availible in the Appendix A.

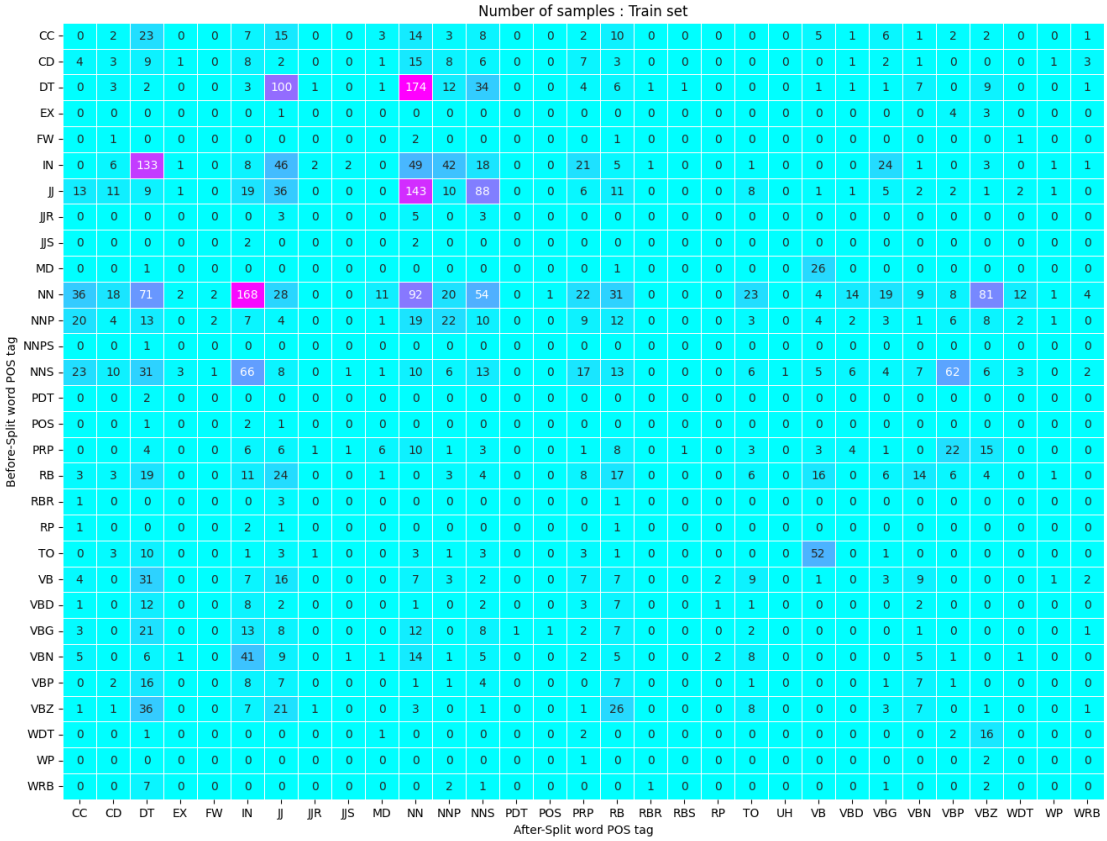

Figure 6: Train set count for each pre and post text boundary POS tag combination

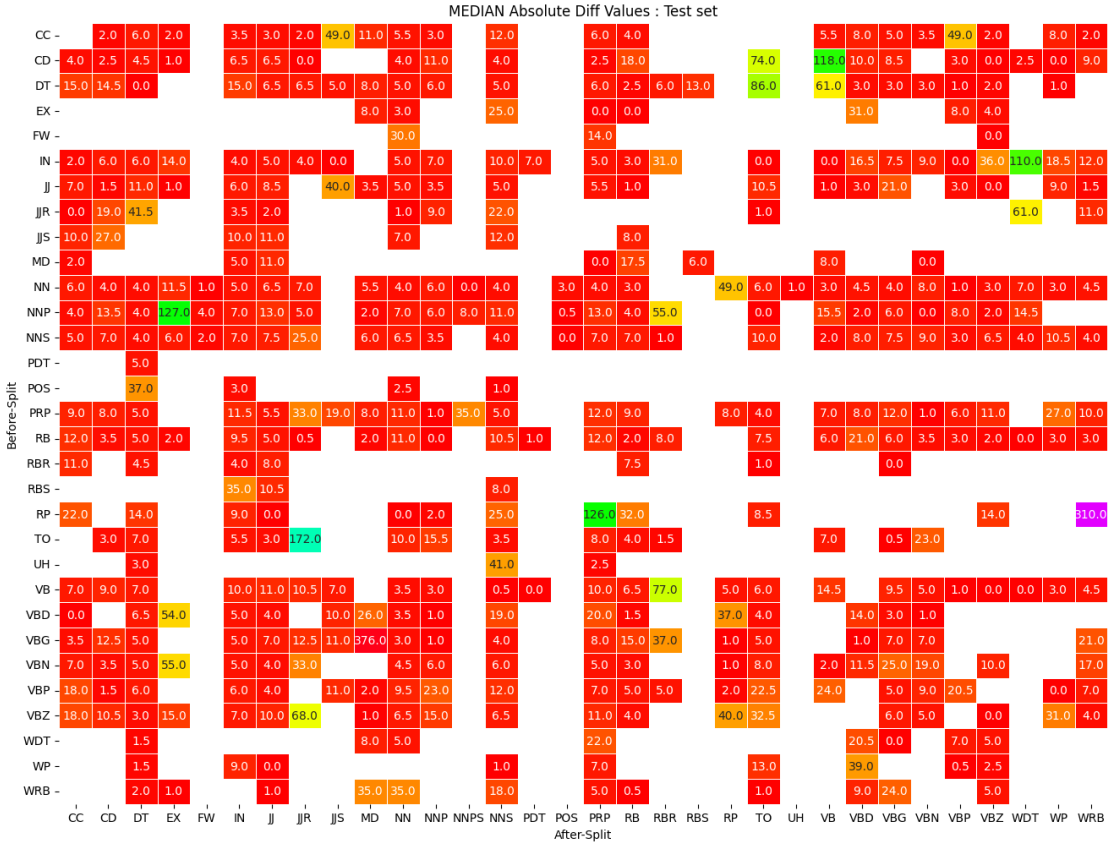

Figure 7: Test set median MAE for each pre and post text boundary POS tag combination

References

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

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. GLTR: statistical detection and visualization of generated text. *CoRR*, abs/1906.04043.

GptKit. GPTKit: A Toolkit for Detecting AI Generated Text. https://gptkit.ai/. Accessed: 2024-02-12.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. In *Transactions of the Association for Computational Linguistics*, volume 8, pages 64–77. MIT Press.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Ryuto Koike, Masahiro Kaneko, and Naoaki Okazaki. 2023. Outfox: Llm-generated essay detection through in-context learning with adversarially generated examples.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Andrew McCallum. 2012. Efficiently inducing features of conditional random fields.

Edward Tian and Alexander Cui. 2023. Gptzero: Towards detection of ai-generated text using zero-shot and supervised methods.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Vice. 2023. AI Is Tearing Wikipedia Apart. https://www.vice.com/en/article/v7bdba/ai-is-tearing-wikipedia-apart. Accessed: 2024-02-12.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Alham Fikri Aji, and Preslav Nakov. 2023. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. *arXiv:2305.14902*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

ZeroGPT. Zerogpt : Reliable chat gpt, gpt4 & ai content detector. https://www.zerogpt.com.

A Other Plots and information

Some of the information that couldn't be covered due to page limiations along with detials for system replication have been added here.

A.1 POS tag usage : human vs machines

It can be seen from Figure 8 , Figure 9 and Figure 10 that machine generated texts had higher share of certain POS tags in the machine genrated parts compared to the human written parts. This was observed in all 3 sets, the train and dev had similar distributions as a result of using same generators i.e ChatGPT and the test had a bit of a variation due to multiple different generators i.e LLaMA2 and GPT4. Although the percentile comparison did vary from train,dev and test sets , it was minimal.

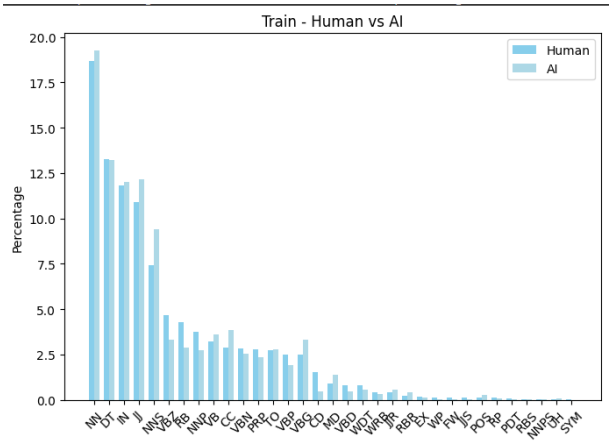

Figure 8: Percentile distribution of each POS tag in train set : human VS machine

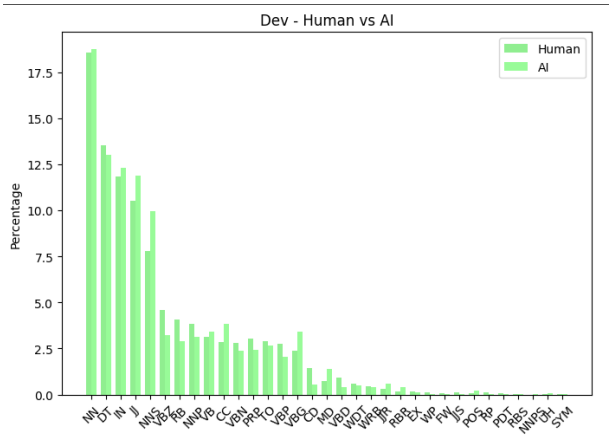

Figure 9: Percentile distribution of each POS tag in dev set : human VS machine

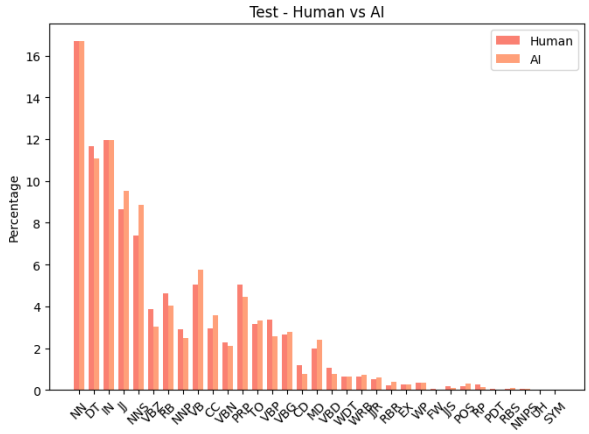

Figure 10: Percentile distribution of each POS tag in test set : human VS machine

A.2 MAE characteristics : DeBERTa vs Longformer

As discussed in the paper , there were some instances where one model performed significantly better than the other as seen in Figure 11 and Figure 12 hinting that an ensemble of both's predictions might yield better results.

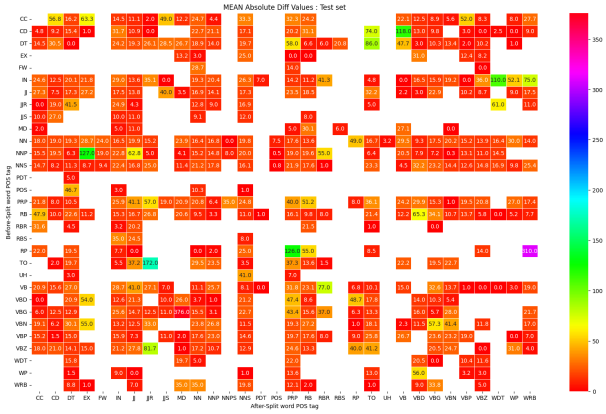

Figure 11: Median MAE based on pre and post text boundary POS tags : DeBERTa-CRF

B potential for misuse and possible solutions

as seen in Figure 6 and Figure 7, the biggest error cases in pre and post text boundary POS tags were the ones which were not present at all or in very minute amount in the training data, nearly 92 percentage of cases had less than 10 samples to train on and 64 percentage of cases had no samples at all in the training set. A potential solution would be including ample amount of data for all possibilites to cover wider range of texts.

Figure 12: Median MAE based on pre and post text boundary POS tags : Longformer.pos-CRF

C System Description

DeBERTa-CRF was the official submission , longformer.pos-CRF had almost the same performance on the test set. i.e 18.538 and 18.542.

Official submission model configuration	
Base model	microsoft/deberta-v3-base
Finetuning :	
Learning rate	2×10^{-5}
Weight decay	1×10^{-2}
CRF Dropout rate	75×10^{-4}
Max length	1024 tokens
Epochs	30
Optimizer	Adam
Preprocessing	No
Trained on	only train set
Sentence separation	nltk: '!' , '.' , '?'
Hardware	1x V100 GPU 16GB RAM

Table 6: Official submission system description : DeBERTa-CRF

Other models that have been tested but were found to have a big margin of performance with above listed models
Due to time and computaional resources limitation, only a part of hyperparameter space was explored.
Time taken for training

D Effect of Text boundary location on performance

The location of text boundaries with respect to lenght of the text samples are varying over the training and testing set as seen in Figure 13 and

Secondary model configuration	
Base model	allenai/longformer-base-4096-extra.pos.embd.only
Finetuning :	
Learning rate	2×10^{-5}
Weight decay	1×10^{-2}
CRF Dropout rate	1×10^{-2}
Max length	4096 tokens
Epochs	30
Optimizer	Adam
Preprocessing	No
Trained on	only train set
Sentence separation	nltk: '!' , '.' , '?'
Hardware	1x V100 GPU 16GB RAM

Table 7: Unofficial submission system description : Longformer.pos-CRF

Other models tested
microsoft/deberta-v3-large
microsoft/deberta-v3-small
microsoft/deberta-v3-xsmall
SpanBERT/spanbert-base-cased
SpanBERT/spanbert-large-cased
allenai/longformer-base-4096
allenai/longformer-large-4096
allenai/longformer-large-4096-extra.pos.embd

Table 8: Other models tested as part of the task

Hyperparameter space explored	
Learning rate	1×10^{-5} 2×10^{-5} 3×10^{-5}
Weight decay	1×10^{-2} 2×10^{-2} 25×10^{-3} 5×10^{-2}
CRF Dropout rates	2×10^{-2} 15×10^{-3} 1×10^{-2} 90×10^{-4} 80×10^{-4} 75×10^{-4} 70×10^{-4} 60×10^{-4}
Max length	1024 tokens 1024-4096 *for longformer
Epochs	10 to 30
Optimizers	Adafactor Adam
Training data	full train set full train+dev set 80% train set stratified on no.of.sent..

Table 9: Hyperparameters explored on the models

Figure 14. Despite training on samples where the text boundaries are in the first half in most of the cases, the models did perform well on the testing set where there is a good amount of samples with text boundaries in later half. This is an area where the proprietary systems struggled.

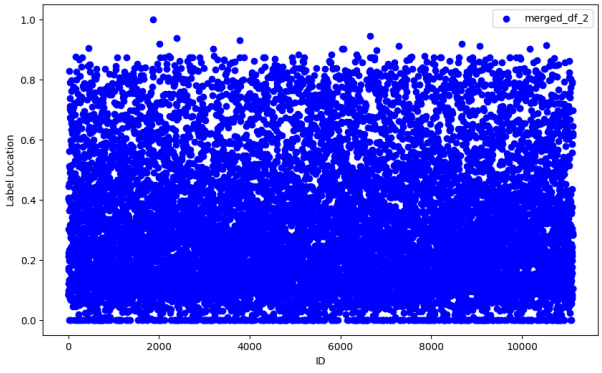

Figure 13: Location of text boundary : training set

Training time	
On Train set	11h 38m (30epochs)
On Train+dev set	14h 4m (30epochs)

Table 10: Training time for the models

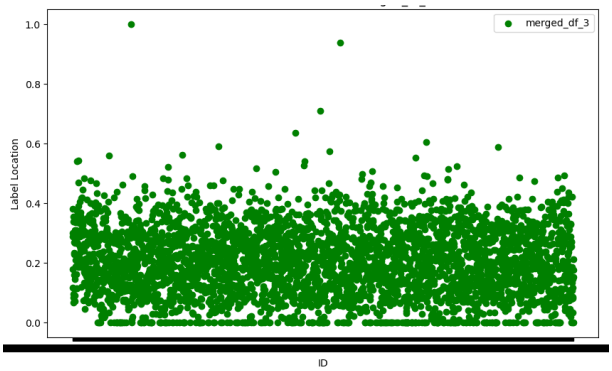

Figure 14: Location of text boundary : training set