SzegedAI at GenAI Detection Task 1: Beyond Binary - Soft-Voting Multi-Class Classification for Binary Machine-Generated Text Detection Across Diverse Language Models

Mihály Kiss
University of Szeged
Institute of Informatics
Kiss.Mihaly@stud.u-szeged.hu

Gábor Berend
University of Szeged
Institute of Informatics
berendg@inf.u-szeged.hu

Abstract

This paper describes the participation of the SzegedAI team in Subtask A of Task 1 at the COLING 2025 Workshop on Detecting AI-Generated Content. Our solutions investigate the effectiveness of combining multi-class approaches with ensemble methods for detecting machine-generated text. This approach groups models into multiple classes based on properties such as model size or generative capabilities. Additionally, we employ a length-based method, utilizing specialized expert models designed for specific text length ranges. During inference, we condense multi-class predictions into a binary outcome, categorizing any label other than human as AI-generated. The effectiveness of both standard and snapshot ensemble techniques is evaluated. Although not all multi-class configurations outperformed the binary setup, our findings indicate that the combination of multi-class training and ensemble methods can enhance performance over single-method or binary approaches.

1 Introduction

In recent years, machine-generated text has become increasingly sophisticated, with advances in generative models. Nowadays, substantial amounts of AI-generated content are being produced, even in the academic and scientific literature (Liang et al., 2024b). Recent findings estimate that between 6.5% and 16.9% of peer review text submitted to AI conferences may be substantially modified by large language models, revealing subtle trends in the use of LLMs within academic settings (Liang et al., 2024a). This trend is further supported by the fact that various models, such as GPT (OpenAI et al., 2024), Mixtral (Jiang et al., 2024), and Gemini (Gemini et al., 2024) have become easily accessible to lay people, even those without a strong technical background. As a result, there is an increasing need for algorithms capable of predicting the difference between human- and machine-generated content. Workshops have been organized to address this need, such as those discussed in (Sarvazyan et al., 2023) and (Chamezopoulos et al., 2024).

In this paper, we present our solutions for Task 1 in the Workshop on Detecting AI-Generated Content at COLING 2025 (Wang et al., 2025). Our focus was on Subtask A, where the objective was to make a binary prediction indicating whether the text was human-written or AI-generated.

The dataset for this subtask contained only English texts. To tackle this challenge, we experimented with various classification schemes to determine the most effective setup for detecting machine-generated text. Binary, 3, 5, 6 and 41 class solutions were developed, with each configuration representing a different heuristic grouping of models based on parameters, model size, or the quality of generated text. Standard ensemble methods with soft voting were applied across all class configurations to improve robustness and performance. An additional experiment involved using a snapshot ensemble (Huang et al., 2017), capturing multiple snapshots of the model during training to create a diverse but computationally efficient ensemble. This snapshot ensemble was created only for the 41-class configuration.

In addition, a length-based solution was developed in which the texts were categorized by length. Three intervals were created based on text length, with an expert model assigned to each interval. These texts were classified according to their length and directed to the corresponding expert model, optimizing performance by matching each text to a model specialized in its specific length range.

Our final submission, the 6-class solution, achieved a macro F1 score of 0.791, though it was not our highest performing model. Our post-shared task evaluation revealed that the length-based approach achieved a macro F1 score of 0.827, while the 41-class standard ensemble scored slightly higher at 0.826.

Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect), pages 166–172
January 19, 2025.

2 Related Work

Recent advancements in detecting machine-generated text have highlighted the efficacy of ensemble learning methods, particularly those that employ soft voting strategies. In the SemEval-2024 Task 8 (Wang et al., 2024), Gu and Meng (2024) developed a class-balanced soft-voting system that fine-tuned transformer-based models, including encoder-only, decoder-only and encoder-decoder architectures (Gu and Meng, 2024). Their approach effectively addressed data imbalance and achieved state-of-the-art performance in multi-class classification tasks involving various text generators.

Building upon this, we explored the application of snapshot ensembles within transformer architectures for machine-generated text detection. This method captures multiple "snapshots" of a model at different training stages, offering computational efficiency over traditional ensembles. Although this approach may not match the robustness of standard ensembles, it presents a viable alternative when resources are limited (Huang et al., 2017).

Another approach for detecting AI-generated text leverages a Transformer Encoder that combines probabilistic features from multiple LLaMA-2 models (Sarvazyan et al., 2024). This model reached impressive accuracy in distinguishing machine-generated from human-written content. The study reveals that integrating probabilistic features from various language models can significantly boost detection precision, with LLMixtic effectively emphasizing the unique features of the final tokens in a sequence.

3 Methodology

3.1 Multi-Class Classification

While the main task was framed as a binary classification problem, it may not fully capture the diversity and variability present in machine-generated content. The category of machine-generated text is a heterogeneous group; it has outputs from a wide array of models that differ in parameter count, training data, and text generation. For example, GPT-4o will likely produce a more coherent and contextually aware text compared to flan_t5_small for the same prompt. This heterogeneity results in a "melting pot" when viewed as a binary task, it can remove meaningful features within the data. To address these challenges, we initially considered a multi-class classification approach.

DeBERTa We chose *DeBERTa-v1-base* (He et al., 2021) for this task due to its well-known strengths and its proven performance in various NLP tasks (C. Timoneda and Vallejo Vera, 2024). To confirm its superiority, we made a comparison with *BERT-base-cased* (Devlin et al., 2019), which showed that DeBERTa consistently outperformed BERT in this binary classification task, achieving higher F1 scores. BERT achieved a macro F1 score of 0.973, while DeBERTa got 0.982 on the development set.

Configuration (used in all experiments): optimizer = AdamW, base learning rate = 2e-5, weight decay = 0.01, warmup steps = 10% of total steps, batch size = 16, 4 validation per epoch and 3 epochs with early stopping.

Binary reduction It is important to note that while we trained the models on multiple classes, all our experiments ultimately reduced the output to a binary label. If the model predicts any class other than "human" it is handled as "1" (AI-generated); otherwise, it is classified as "0" (human).

3.1.1 41-class: one class per generative model

Since the shared task dataset includes information on which model generated the text, our aim was to capture this and treat each model as a different class. This approach introduced 41 classes in total, since the training texts were generated by 40 different models. However, this method may lead to a fragmented dataset, where individual classes contain limited data for robust training and analysis.

Our hypothesis is that a better classification granularity could improve model effectiveness by organizing classes based on the generative capabilities or the parameter count of the generative models. To explore this, we also implemented 3, 5, and 6 class solutions. By adopting these diverse class structures, we aimed to create categories with highly distinct properties, allowing models to effectively differentiate between them.

3.1.2 Modeling based on the parameter count

5-class Our second approach was to create groups based on the parameter count of the models, as we hypothesized that categorizing the models by this criterion might reveal performance differences more clearly. This grouping strategy allows us to examine whether models with different parameter scales show distinct behaviors or predictive capabilities in binary classification tasks. By seg-

menting models according to parameter count, we aim to highlight potential trends in model efficacy and better understand how complexity influences outcomes.

We reviewed the models and recorded the number of parameters for each, hoping that grouping them into intervals by parameter count would show clear differences in capabilities: less than 5 billion, $6-10$ billion, $11-130$ billion, and over 130 billion.

6-class In our next approach, we introduce an additional interval in the lower range, refining the parameter-based grouping to examine performance distinctions more precisely. This allows us to check how vulnerable certain parameter sizes are, especially smaller models, to capturing nuanced distinctions. We split the less than 5B category into two, at the threshold of 1B parameters, allowing a more accurate analysis of how smaller model scales may impact binary classification performance. The 6 different classes are shown in Figure 2 of Appendix A.

3.1.3 3-class: generative capability grouping

While parameter count serves as a valuable metric for model complexity, it alone may not fully capture a model's performance capabilities. We hypothesized that additional indicators, such as leaderboard rankings and perplexity scores, could provide more information on the effectiveness of a model. By including these additional evaluations, we aimed to refine our classification and account for differences that parameter count might not be able to effectively capture. This approach led us to consider leaderboard-based *ELO* score as an additional metric for a more complete evaluation.

Two leaderboards were analyzed: the LLM Explorer leaderboard[1] and Chatbot Arena[2]. Although the LLM Explorer leaderboard featured a larger selection of models, only a few were listed on the Chatbot Arena leaderboard, and only four models appeared on both leaderboards. Based on these shared models, we performed a linear regression to align the scores, thereby creating a unified scale for all models. This regression produced the following equation, where x represents the LLM explorer leaderboard score: $\text{ELO} = 1238 \cdot x + 484$. Using this formula, we calculated an ELO score for each model.

A threshold for classification was established: models with an ELO score above 800 were categorized as *Strong*, while those with scores below 800 were categorized as *Weak*. We thresholded at 800 due to a noticeable gap between 700 and 800, suggesting that this range might represent a meaningful difference in generative capabilities. For the 19 models that were not found on any leaderboards, we assigned them to the class that seemed the closest match based on our judgment. This approach resulted in a three-class classification task: *human, weak, strong*. The final groups are shown in Figure 2 of Appendix A.

3.2 Binary Ensemble

Ensemble methods, where multiple models contribute to a final prediction, are well-established for improving accuracy and robustness in machine learning tasks (Ganaie et al., 2022). In the context of machine-generated text detection, ensemble approaches combine the strengths of individual models, often leading to enhanced performance. However, these methods are computationally expensive, as they require fine-tuning multiple models independently. Despite these costs, our results demonstrate that using ensemble shows substantial improvement.

We fine-tuned three models for the original binary task and applied soft voting, comparing its performance to the average F1 score of each model operating independently. The results, showing soft voting versus the individual model performance, are presented in Table 1.

3.3 Multi-Class Ensemble

Gu and Meng (2024) demonstrated the effectiveness of ensemble methods for multi-class machine-generated text detection, showing significant improvements in both accuracy and robustness across diverse classes. Building on these findings, we further enhanced our approach by incorporating a soft voting technique within our ensemble, allowing for more reliable and consistent predictions across various text categories.

Similarly to the binary ensemble approach, we fine-tuned three models, applied soft voting, and compared the results to the average F1 score of 3 single fine-tuned models. This process was used in all of our experiments.

3.4 Using Length-Adaptive Expert Models

We examined the performance of various models in the data set to identify patterns in the generated

[1] https://llm.extractum.io/list/?small
[2] https://lmarena.ai/

texts. We observed, as expected, that larger models such as GPT-4 tended to produce significantly longer texts. This is illustrated in Figure 2 of Appendix A. This natural variance in the output length suggests that a length-based approach to model grouping could improve the detection accuracy.

With this approach, the models were divided into three length-based groups, each represented by a specialized expert model for short, medium, and long text intervals. We chose the length cutoff values at 745 and 1613, so the three groups nearly contained the same amount of texts. This structure was hypothesized to improve classification precision by leveraging each model's inherent tendencies within its length category, ultimately contributing to a more robust and accurate text classification system. The fine-tuning here was also handled as a 41-class problem.

3.5 Snapshot Ensemble

We implemented a snapshot ensemble (Huang et al., 2017), capturing several "snapshots" of a model at different stages throughout its training. Unlike traditional ensembles that train multiple independent models from scratch, this method reuses the same model's evolving states, making it significantly more resource-efficient. The diversity among snapshots is introduced through a cyclical learning rate schedule, where the learning rate varies between a high and low value over several cycles during training. We implemented the cyclical learning rate ourselves, following the original design in the paper. Each cycle allows the model to explore a wider range of parameter spaces, capturing unique representations in each snapshot. We chose to use six models in our implementation. This choice was primarily practical, ensuring enough diversity among the snapshots while maintaining computational efficiency. After that, we evaluated combinations of three snapshot checkpoints to find the set that achieves the best results together through soft voting. Later on, we used these three models.

3.6 Data Split With 3 Models

With this experiment, we also aimed to find a computationally efficient solution with a promising result. The main idea was that the availability of a sufficiently large amount of training data suggested that the models could perform well even when trained on only a subset of it, thereby reducing computational requirements without significantly reducing the performance.

In this approach, we fine-tuned three separate models for 41-class classification, each trained on a unique one third subset of the training data. To enhance the final predictions, we combined the outputs of these models using the previously introduced ensemble method with soft voting.

4 Results

Our final submission was a 6-class soft voting, which secured 7th place. We evaluated our solutions in both the development set and the published test set. Based on these evaluations, the 6-class approach was not the optimal choice. The best performing experiment was the length-based solution with soft voting, closely followed by the simple 41-class approach. The corresponding F1 scores are shown in Table 1. *Without Soft Voting* refers to the average F1 score of three individual models evaluated separately, while *Soft Voting* refers to applying soft voting across these three models.

Experiment	Without Soft Voting		Soft Voting	
	Development	Test	Development	Test
Binary	0.972	0.780	0.979	0.795
3-class	0.983	0.750	0.985	0.754
5-class	0.982	0.790	**0.986**	0.796
6-class	0.980	0.771	0.985	0.791
41-class	0.982	0.806	0.985	0.826
Snapshot	0.978	0.796	0.982	0.810
Length-based	0.968	0.820	0.973	**0.827**
Data split models	0.980	0.801	0.982	0.814

Table 1: Summary of results on development and test set by class count and ensemble setting (F1 scores)

Length-based Analysis The length-based approach achieved the best results. Our aim was to understand where it failed and how it did so to provide a deeper understanding of its limitations. To achieve this, we created a confusion matrix which is shown in Figure 1, allowing us to identify patterns and categories of errors. Our error analysis revealed that errors predominantly occurred within model families. For example, the model often struggled to accurately determine which specific *opt_* model generated a given text.

In addition, we examined the results for each text length interval. The analysis showed a clear trend: the longer the text, the better the results. The F1 scores for different character count intervals are as follows: For intervals of 0–745 characters, the F1 score is 0.621; for 746–1613 characters, it is 0.684; and for texts with 1614 or more characters, the F1 score increases to 0.840.

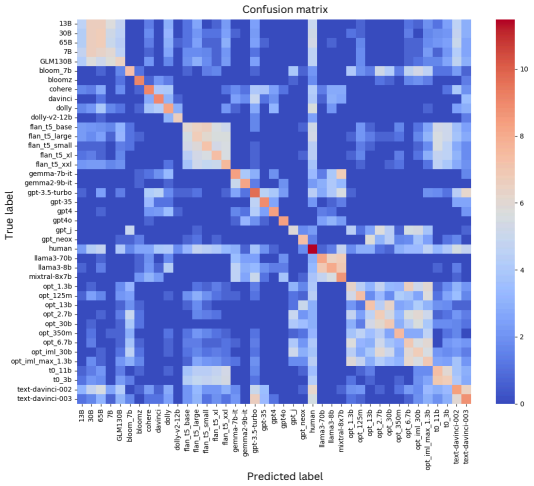

Figure 1: Log-scaled confusion matrix for the length-based solution

ZeroGPT We compared our own experiments with another solution that is easily accessible from a browser, which is ZeroGPT[3]. Detailed information about the model and techniques they use is not available, but their website promises good results. This solution can be accessed via an API call, allowing requests to be sent. For evaluation, we used 2000 random texts from the workshop test dataset and also another 2000 texts from a previous shared task dataset named AuTexTification (Sarvazyan et al., 2023), which is completely independent of our solutions. We could not evaluate more data as it would have required payment. We did the same evaluation on the exact same texts with our own solutions. Their system operates by returning a whole number between 0 and 100. This value indicates the likelihood that the text was written by a human. Since their results are provided in this format, a threshold had to be established. This threshold was set at 50, treated as a closed interval, which means that a score of 50 was also included in this category. Modifying this threshold value does not significantly affect the predictions results.

The results presented in Table 2 indicate that ZeroGPT exhibited poorer performance compared to all of our solutions in both datasets. The workshop test data set comprised text generated by advanced models, while the independent data set included text from both advanced and less advanced models. This highlights the robustness and adaptability of our solutions to machine-generated text classification tasks.

[3] https://www.zerogpt.com/

Experiment	Shared task	AuTexTification
zeroGPT	0.615	0.715
Binary	0.794	0.797
3-class	0.830	0.767
5-class	**0.841**	0.814
6-class	0.838	0.805
41-class	0.831	0.836
Snapshot	0.812	0.826
Length-based	0.807	**0.845**

Table 2: F1 scores for ZeroGPT and our solution, evaluated on 2000 texts each from the AuTexTification (Sarvazyan et al., 2023) and this shared task test set

5 Conclusion

Our findings highlight that the binary approach may not be the most effective way to detect machine-generated texts. Instead, we recommend considering multi-class solutions, as they might improve classification by capturing distinctions among AI-generated texts. In our experiments, we found that different approaches performed well on different datasets: one solution showed strong results on the development set, while another excelled on the test set. For example, the length-based approach was not promising in the development set but performed well on the test set.

To evaluate our approach comprehensively, we considered not only the results, but also the computational efficiency. We implemented a snapshot ensemble technique and also a data split approach with 3 models. Both strategies demonstrated improved performance compared to a single model fine-tuned on the full training dataset while maintaining the same computational costs. However, the more expensive solutions still outperformed these.

Notably, the 3-class approach showed a drop in performance on the test set, likely due to the absence of weaker models in the test data. This suggests that certain multi-class configurations may be less effective when faced with a set of high-performing models only. However, even with these variations, the 41- and 5-class solutions outperformed the binary approach and provided a consistent gain over binary modeling.

Ultimately, our approach demonstrates that moving beyond binary classification to a multi-class strategy, especially when using ensemble soft voting, can yield better results in detecting machine-generated text.

Acknowledgments

The research received support from the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory. Additionally, we are grateful for the possibility to use ELKH Cloud (see (Héder et al., 2022); https://science-cloud.hu/) which helped us in achieving the results published in this paper.

References

Joan C. Timoneda and Sebastián Vallejo Vera. 2024. BERT, RoBERTa or DeBERTa? comparing performance across transformer models in political science text. *The Journal of Politics*.

Savvas Chamezopoulos, Drahomira Herrmannova, Anita De Waard, Drahomira Herrmannova, Domenic Rosati, and Yury Kashnitsky. 2024. Overview of the DagPap24 shared task on detecting automatically generated scientific paper. In *Proceedings of the Fourth Workshop on Scholarly Document Processing (SDP 2024)*, pages 7–11, Bangkok, Thailand. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

M.A. Ganaie, Minghui Hu, A.K. Malik, M. Tanveer, and P.N. Suganthan. 2022. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151.

Gemini, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Emily Pitler, Timothy Lillicrap, and et al. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Renhua Gu and Xiangfeng Meng. 2024. AISPACE at SemEval-2024 task 8: A class-balanced soft-voting system for detecting multi-generator machine-generated text. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1476–1481, Mexico City, Mexico. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations*.

Mihály Héder, Ernő Rigó, Dorottya Medgyesi, Róbert Lovas, Szabolcs Tenczer, Ferenc Török, Attila Farkas, Márk Emődi, József Kadlecsik, György Mező, Ádám Pintér, and Péter Kacsuk. 2022. The past, present and future of the ELKH cloud. *Információs Társadalom*, 22(2):128.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Weixin Liang, Zachary Izzo, Yaohui Zhang, Haley Lepp, Hancheng Cao, Xuandong Zhao, Lingjiao Chen, Haotian Ye, Sheng Liu, Zhi Huang, Daniel Mcfarland, and James Y. Zou. 2024a. Monitoring AI-modified content at scale: A case study on the impact of ChatGPT on AI conference peer reviews. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 29575–29620. PMLR.

Weixin Liang, Yaohui Zhang, Zhengxuan Wu, Haley Lepp, Wenlong Ji, Xuandong Zhao, Hancheng Cao, Sheng Liu, Siyu He, Zhi Huang, Diyi Yang, Christopher Potts, Christopher D Manning, and James Y. Zou. 2024b. Mapping the increasing use of LLMs in scientific papers. In *First Conference on Language Modeling*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Areg Mikael Sarvazyan, José Ángel González, and Marc Franco-salvador. 2024. Genaios at SemEval-2024 task 8: Detecting machine-generated text by mixing language model probabilistic features. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 101–107, Mexico City, Mexico. Association for Computational Linguistics.

Areg Mikael Sarvazyan, José Ángel González, Marc Franco-Salvador, Francisco Rangel, Berta Chulvi, and Paolo Rosso. 2023. Overview of AuTexTification at IberLEF 2023: Detection and attribution of machine-generated text in multiple domains. *Proces. del Leng. Natural*, 71:275–288.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, and Thomas Arnold. 2024. SemEval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 2057–2079, Mexico City, Mexico. Association for Computational Linguistics.

Yuxia Wang, Artem Shelmanov, Jonibek Mansurov, Akim Tsvigun, Vladislav Mikhailov, Rui Xing, Zhuohan Xie, Jiahui Geng, Giovanni Puccetti, Ekaterina Artemova, Jinyan Su, Minh Ngoc Ta, Mervat Abassy, Kareem Elozeiri, Saad El Dine Ahmed, Maiya Goloburda, Tarek Mahmoud, Raj Vardhan Tomar, Alexander Aziz, Nurkhan Laiyk, Osama Mohammed Afzal, Ryuto Koike, Masahiro Kaneko, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2025. GenAI content detection task 1: English and multilingual machine-generated text detection: AI vs. human. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*, Abu Dhabi, UAE. International Conference on Computational Linguistics.

A Average Text Length and Strong Models

In Figure 2, we present the 6-class groups, and the striped bars indicate models categorized as *Strong*. All other models not listed under the *Strong* models are categorized as *Weak*.

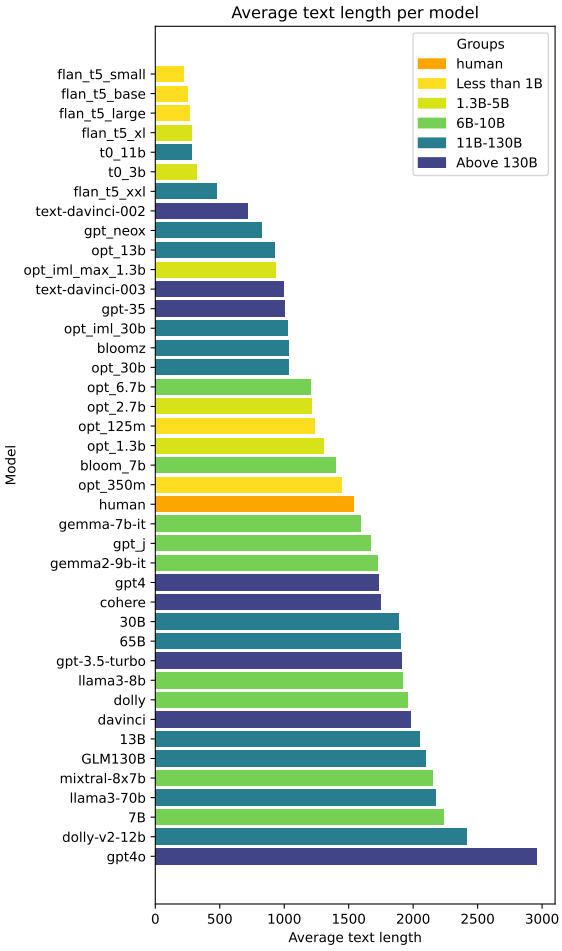

Figure 2: Average character per model