TueCICL at SemEval-2024 Task 8: Resource-efficient approaches for machine-generated text detection

Daniel Stuhlinger
University of Tübingen
daniel.stuhlinger@student.uni-tuebingen.de

Aron Winkler
University of Tübingen
aron.winkler@student.uni-tuebingen.de

Abstract

Recent developments in the field of NLP have brought large language models (LLMs) to the forefront of both public and research attention. As the use of language generation technologies becomes more widespread, the problem arises of determining whether a given text is machine generated or not. Task 8 at SemEval 2024 consists of a shared task with this exact objective. Our approach aims at developing models and strategies that strike a good balance between performance and model size. We show that it is possible to compete with large transformer-based solutions with smaller systems. Our code can be found on GitHub. [1]

1 Introduction

Recent developments in the field of NLP have brought large language models (LLMs) to the forefront of both public and research attention. With the introduction of GPT-3 (Brown et al., 2020), made accessible to the public through ChatGPT, the gates were open to the generation of high-quality text through AI. This leads to a sort of arms race both in integrating AI into customer-facing products, as well as in developing the models themselves, leading, among others, to Facebook's Llama (Touvron et al., 2023) and the open source model Mistral (Jiang et al., 2023).

Many fields have seen a dramatic increase in the use of LLMs, including arts, education, software development, and many more. Initial public reaction to the popularisation of LLM-powered chat interfaces already highlighted its potential problems for plagiarism detection in scientific, educational and other contexts (Dehouche, 2021). In a landmark development, the 5-month-long strike of the Writer's Guild of America[2] resulted in an agreement which included safeguards for writers against the use of artificial intelligence.[3]

The widespread use of language generation technologies is only expected to grow. However, this development is accompanied by a surging need to automate the process of flagging machine-generated text.

Crothers et al., 2023 offers a comprehensive survey of machine-generated text detection strategies. Statistical methods, with global text vector representations, were among the early strategies adopted to tackle the issue. These feature vectors include, for example, TF-IDF, frequency features investigating word or n-gram distributions, readability-related features such as the Gunning-Fog index, or linguistic features such as POS-tag distributions and coreference resolution relationships within a text. The subsequently introduced neural approaches show better performance, even when paired with the aforementioned text representation strategies. Most prominently, transformer fine-tuning has established itself as more or less of a standard, with RoBERTa (Liu et al., 2019) in particular being the most strongly represented model. Zero-shot approaches, optionally coupled with fine-tuning, have also seen experimentation, but have been observed to generalise poorly across domains.

Task 8 at SemEval-2024 (Wang et al., 2024) is a shared task built around the idea of detecting machine-generated texts across a variety of domains and setups. It is structured across three subtasks (subtask A, subtask B, subtask C), our team decided to only tackle subtasks A and C. Subtask A is a binary classification task, with the objective of determining whether a text is human- or machine-generated. The subtask has a monolingual (English) and a multilingual track - our team only

[1]https://github.com/cicl-iscl/TueCICL_SemEval2024
[2]https://www.vox.com/culture/2023/9/24/23888673/wga-strike-end-sag-aftra-contract
[3]https://www.nbclosangeles.com/news/local/hollywood-writers-safeguards-against-ai-wga-agreement/3233064/

submitted for the monolingual track. Subtask C is a boundary detection task: here, texts have a human segment followed by a machine-generated segment. The objective of the subtask is to correctly predict the boundary index.

Our approach for both tasks was to try to obtain competitive solutions with low resource cost. For this reason we deployed two different model classes: LSTM-based models (Hochreiter and Schmidhuber, 1997) and Multilayer Perceptrons (MLP) (Popescu et al., 2009) and relied on various strategies of representing the input texts, either at the token or at the text level. We experimented with character-level approaches, systems relying on pretrained Word2vec (Mikolov et al., 2013) embeddings, and linguistically motivated features at the text level through spaCy (Honnibal et al., 2020) and the TextDescriptives (Hansen et al., 2023) package.

2 Methods and experimental setup

2.1 Subtask A

For subtask A, our intuition was that surface-level and stylistic features would be more effective than semantics in discriminating between human and machine-generated text. To build on this idea, we developed three approaches.

The first approach involved training a character-level LSTM. We expect the stylistic features of the texts to be good indicators of the generator, and working at the character level is known to capture this information well. For example, character n-gram models have been used successfully in the field of authorship attribution, which relies heavily on style (Stamatatos et al., 2013).

First, input texts are tokenized at the character-level, all tokens are mapped to their lowercase variants, and lastly numerals and punctuation are mapped to a `<NUM>` and a `<PUNCT>` special token respectively. White-space elements (space, tab, newline) are also mapped to a special token `<WS>`. At this point, the tokenized and transformed inputs are fed through an LSTM, and the representation of the last token is used for prediction.

The second approach is constructed along the lines of the first in terms of technical setup, but deals with words rather than characters. Large transformer-based solutions benefit from vast amounts of pre-training, but at a heavy computational cost – using pretrained embeddings as model inputs appeared to be a good compromise between heavy models and training from scratch, as was the case with the character-level LSTM. We used the pretrained Word2vec embeddings from the Wikipedia2Vec (Yamada et al., 2020) project to map texts to vectors, but maintained the other steps, such as mapping numerals, punctuation and white-space to special tokens.

The third approach is not recurrent in its nature – instead, we used the TextDescriptives pipeline (Hansen et al., 2023) through Spacy (Honnibal et al., 2020) to obtain 66 lingustically motivated features to globally represent the text. Such linguistic features have a long tradition in NLP, for example in the field of readability analysis (for example Vajjala and Meurers, 2012), and have been observed to be valid and cheap-to-compute representations in a variety of settings. Since they are most well known for capturing the style of a text, rather than semantics, they appear to be very well suited for the present task. Additionally, we computed the mean perplexity of the document using GPT-2 (Radford et al., 2019) and added it to the feature vectors. This follows the idea that the perplexity of a document assigned by a LLM should be higher for human written texts than for machine generated texts (Chaka, 2023). Our third approach computes this global feature vector for the input text, then generates a prediction through a simple MLP. The model consists of 3 linear layers with Tanh activation functions in between and was trained for 2000 epochs with a learning rate of 0.0003.

Lastly, we formulated a joint model which takes as input the final representations (the last hidden states) of each of the three previous approaches, thus generating a single prediction.

Model	**Type**	**L***	**H****
Character-level	LSTM	2	512
Word2vec	LSTM	2	512
Language features	MLP	3	256
Joint model	FFN	-	512

Table 1: Summary of models for subtask A. * Number of layers. ** Hidden size.

2.2 Subtask C

For subtask C, which required predicting the boundary position in texts between a human and a machine-generated segment, we adopted the same guiding principles in developing our solutions as we had done in subtask A. The overarching objective was to arrive at competitive models while remaining within a certain size constraint.

An additional challenge for this subtask, aside from an increase in difficulty in the objective itself, was the relative scarcity of training data. While in subtask A the training set had over one hundred thousand records, the training material in subtask C consisted of just below four thousand texts. As such, any strategy that did not involve pretraining would be severely disadvantaged in this setting.

Our first approach for this subtask was also a character-level solution, with the same general setup as in subtask A. For generating a prediction, however, the representations at every token are evaluated, and a classification is performed. In this sense, the model is predicting whether any given character is in the human or the machine-generated segment of the text. The first token predicted to be machine-generated is taken to be the boundary position. Importantly, the LSTM we implemented for this subtask is bidirectional, meaning at every token the model has awareness of both the left and right contexts.

To offset the relative lack of training items, together with the absence of any model pre-training in this case, we trained this model on both subtask A and subtask C data for 5 epochs, then trained further on only task C data for 3 further epochs. This improved performance significantly on the development set.

We also implemented a Word2vec solution along the lines of what was described for subtask A. We opted for a bidirectional LSTM, and enhanced the training data as described earlier, though the effects of this were less prominent owing to the use of pretrained embeddings.

We also built a joint model over the aforementioned character- and Word2vec models. This consisted in a FFN whose inputs were the concatenated representations at the word level. For the character-level model, this meant averaging the representation at every character for any given word.

Model	**Type**	**L***	**H****
Character-level	LSTM	2	512
Word2vec	LSTM	2	512
Joint model	FFN	-	256

Table 2: Summary of models for subtask C. * Number of layers. ** Hidden size.

3 Results

3.1 Subtask A

Model	**Dev**	**Test**	**Ranking**
Baseline	0.72	0.88	20
Character-level	0.85	0.55	127
Word2vec*	0.82	0.72	85
Language features*	0.63	0.88	21
Joint model*	0.83	0.69	96

Table 3: Results for SemEval-2024 Task 8, subtask A. Dev and Test columns report the accuracy on the respective data partitions. The ranking column refers to the model ranking in the shared task competition. The scores and ranking of the unofficial submissions were not provided by the organisers and computed by us. There was a total of 137 submissions.
* unofficial submissions

Table 3 shows the results for each model on subtask A. On the development set, almost all models outperform the transformer baseline provided by the organisers. The best performing model was the character-level model, with an accuracy of 0.85 – this was our final submission for the shared task.

While the two recurrent models and the joint model do not differ very much from one another, the FFN built on linguistically motivated global feature vectors sets itself apart in that it is the worst performing model on the development set.

The character-level model only achieves an accuracy of 0.55 on the test set, while the Word2vec and joint models achieve 0.72 and 0.69 respectively – all falling short of the baseline. Surprisingly, the language feature model is head and shoulders above the rest when it comes to the test set, with an accuracy of 0.88 that matches the baseline. Our assumption is that this is due to a conceptual difference between development and test set. A possible reason could be that the domain for human-written texts in the test set were student essays only. The development set on the other hand consisted of human-written texts from multiple domains. The linguistic features prevalent in the student essays seem to be more distinctive for classifying the documents compared to the texts from multiple domains.

3.2 Subtask C

Table 4 outlines our results for subtask C. In this subtask, we were unable to match the transformer baseline.

Model	Dev	Test	Ranking
Baseline	3.53	21.54	14
Character-level*	8.35	45.83	28
Word2vec*	7.02	38.35	27
Joint model	6.36	34.88	25

Table 4: Results for SemEval-2024 Task 8, subtask C. Dev and Test columns report mean absolute error (MAE) on the respective data partitions. The ranking column refers to the model ranking in the shared task competition. The scores and ranking of the unofficial submissions were not provided by the organisers and computed by us. There was a total of 33 submissions.
* unofficial submissions

Our official submission, the joint model, achieved a mean standard error of 6.36 on the development set, falling short of around 3 points from the baseline provided by the organisers. The difference is even more dramatic when it comes to the test set, where the gap widens to around 13 points. This is also far off from the best performing solutions in the shared task, which achieved a MAE of 15.7.

The character and Word2vec models failed to outperformed both the baseline and the joint model for the development set, and this remains the case in the test set. This reinforces the idea that extracting as much information as possible from the texts is key to performance in this subtask.

Overall, the models developed for subtask C have proven to be somewhat unrefined. The test set seems to be particularly punishing towards solutions that do not generalize well, but the results, while highlighting the shortcomings of our models, also point toward the potential that these approaches can have, with more attention dedicated to them.

4 Conclusion and Discussion

Our objective for Task 8 at Semeval-2024 was to compete with large-scale solutions with models that could run on commonplace systems like mid-range laptops. For this purpose, we discarded LLM-based solutions that are prevalent in the related work in the field, opting instead for LSTMs and MLPs whose size can more easily be controlled.

While our team did manage to keep model size under control (none of the proposed solutions require more than 1 GB of memory), the systems we proposed performed less than ideally in the task itself. On both subtasks we participated in, our best models failed to match the transformer baseline in the test set, despite positive results in the development set.

Despite the final results, we believe our approach to be valid. Our development processes likely ended up producing models that were overly tuned to the development set. With more time, it would be possible to produce more refined and generalizable solutions. Trying different training strategies, like contrastive learning, or different architectures, such as mixture-of-experts systems, might be a good direction to follow in future work.

To most problems that arise in the field of NLP, researchers and companies increasingly respond with huge models that require dedicated servers to run. But for many users, keeping their data safely on their own machines is a priority, thus discarding many of the contemporary LLM-based services. System designers should aim to strike a compromise between size and performance, and prioritise users being able to own their workflows when possible. Like our attempt did for this shared task, we believe researches should consider these objectives when proposing new solutions.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Chaka Chaka. 2023. Detecting AI content in responses generated by ChatGPT, YouChat, and Chatsonic: The case of five AI content detection tools. *Journal of Applied Learning and Teaching*, 6(2).

Evan N. Crothers, Nathalie Japkowicz, and Herna L. Viktor. 2023. Machine-generated text: A comprehensive survey of threat models and detection methods. *IEEE Access*, 11:70977–71002.

Nassim Dehouche. 2021. Plagiarism in the age of massive generative pre-trained transformers (GPT-3). *Ethics in Science and Environmental Politics*, pages 17–23.

Lasse Hansen, Ludvig Renbo Olsen, and Kenneth Enevoldsen. 2023. TextDescriptives: A Python package for calculating a large variety of metrics from text. *Journal of Open Source Software*, 8(84):5153.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Marius-Constantin Popescu, Valentina E. Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 7(1):579–588.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Ph D Stamatatos et al. 2013. On the robustness of authorship attribution based on character n-gram features. *Journal of Law and Policy*, 21(2):7.

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.

Sowmya Vajjala and Detmar Meurers. 2012. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 163–173, Montréal, Canada. Association for Computational Linguistics.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024. SemEval-2024 task 8: Multigenerator, multidomain, and multilingual black-box machine-generated text detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation*, SemEval 2024, Mexico City, Mexico.

Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. 2020. Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 23–30. Association for Computational Linguistics.