Nicla at DeSegMa-IT: DistilBERT for MGT Detection and LightGBM Regressor for HMT Segmentation

Nicla Auletta[1,*]

[1]Department of Computer Science, University of Pisa

Abstract

Recent advancements in Generative AI have led to the development of systems able to generate text that is often indistinguishable from human-written content. Although this capability has many beneficial applications, it also enables malicious actors to generate synthetic content for deceptive purposes (e.g., proliferation of fake news). As a direct response to all these needs DeSeGma-IT is introduced in the EVALITA 2026 campaign. This challenges the independent and identically distributed assumption (IID), since the generators used to create the training set are different from those used to generate the test set.

DeSegMa is structured into two sub-tasks. Machine Generated Text (MGT) Detection which involves the detection of machine-generated texts at the document level and Human-Machine Text (HMT) Segmentation which involves the segmentation of a document into the human-written and the machine-generated part.

For the first sub-task, DistilBERT is chosen because it has fewer parameters and it is faster than BERT, but it keeps nearly all the language understanding capabilities. For the second sub-task, Light Gradient Boosting Machine (Light GBM) is chosen because it is faster than Gradient Boosting Decision Tree (GBDT) while achieving almost the same accuracy.

The results demonstrated that both models maintain stable performance when moving from validation data to unseen test data.

Keywords

DistilBERT, MGT Detection, LightGBM Regressor, HMT Segmentation.

1. Introduction

Recent advancements in Generative AI and Large Language Models have led to the development of systems able to generate text that is often indistinguishable from human-written content. Although this capability has many beneficial applications, it also enables malicious actors to generate synthetic content for deceptive purposes (e.g., proliferation of fake news).

Beyond public opinion concerns, the European Commission will enforce compliance with the AI Act. In particular, article 50 requires that text generated by General Purpose AI have to be marked and automatically recognizable. Additionally, while state-of-the-art MGT detectors have reported high accuracy, such results often stem from unrealistic experimental settings.

As a direct response to all these needs, DeSeGma-IT - Detection and Segmentation of Machine Generated Text in Italian - [1] is introduced in the EVALITA 2026 campaign [2]. In fact, it challenges the independent and identically distributed assumption (IID), since the generators used to create the training set are different from those used to generate the test set.

DeSegMa is structured into two sub-tasks. SubTask A - Machine Generated Text (MGT) Detection in the Wild - involves the detection of machine-generated texts at the document level, it is structured as a binary-classification problem and it is defined as follows:
Given a piece of text, assign it the label 0, if the text is written by a human, and 1 otherwise.

EVALITA 2026: 9th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, Feb 26-27, Bari, IT
*First author.
niclaauletta.na@gmail.com (N. Auletta)
https://github.com/NiclaAuletta (N. Auletta)
https://www.linkedin.com/in/https://www.linkedin.com/in/nicla-auletta/ (N. Auletta)

SubTask B - Human-Machine Text (HMT) Segmentation - involves the segmentation of a document into the human-written and the machine-generated part, it focuses on localization and it is defined as follows:
Given a piece of text, return the index of the first character that is generated by an LLM [1].

2. Dataset

2.1. Data Understanding

For SubTask A the features of the dataset are text - the text to classify - and label - the ground truth, where 0 means that the text is human written and 1 that it is machine generated.

For SubTask B the features of the dataset are human - the first part of the text, written by a human -, llm - the machine-generated continuation of the text - and human_len - the index of the first character outputted by the language model [1].

2.2. Data Preparation

First, duplicate rows are dropped from both the datasets because with them, the model might incorrectly learn that these instances are much more common or significant than they are.

For SubTask A, only technical noise that do not help detect whether a text is a MGT are removed.
In fact, HTML tags and markdown artifacts, like # and * are used to represent formatting, while URLs are references.
Moreover, emojis add context, emotion, and nuance, but they are the very things computers find hard to quantify and here the goal does not concern sentiment.
Finally, duplicate whitespaces should be normalized to a single space to ensure consistent formatting.

Instead, characters that may be decisive for detection are not removed. For instance, stopwords provide crucial grammatical structure, while punctuation marks may or may not offer valuable information based on the context. Stylometric features, including punctuation marks effectively enhance AI-generated text detection [3].

For SubTask B, nothing is removed from the text before executing this code because it is needed to return the index of the first character from which the llm part starts. Any preprocessing would shift offsets, breaking the mapping between the original text and the computed embeddings. So, the only step done is to concatenate the human and the llm texts.

3. Description of the system

3.1. Subtask A

3.1.1. Model Selection and Architecture

For the classification task, DistilBERT [4] is chosen. It is a compact transformer-based architecture derived from BERT via knowledge distillation. This model is selected for its favourable trade-off between computational efficiency and performance, as it keeps 97% of the language understanding capabilities, while being 60% faster and 40% smaller in parameter count [4]. Specifically, the uncased version is chosen to ensure robustness across varying capitalization styles[1].

This model is primarily aimed at being fine-tuned on tasks that require the whole textual content, as is the case for DeSegMa subtask A. Given its ability to effectively capture the subtle semantic and syntactic

[1] https:huggingface.co/distilbert/distilbert-base-uncased

patterns critical for differentiating between machine-generated text (MGT) from human-written text (HWT) [5], DistilBERT serves as an ideal back bone for this sequence-level classification task.

3.1.2. Data Processing and Experimental Setup

Textual inputs were tokenized and truncated to the model's maximum sequence length. Dynamic padding is implemented during batch collation, ensuring that sequences were padded only to the length of the longest element within each batch. In fact, this is more efficient with respect to padding the whole dataset to the maximum length [6].

For model validation and hyperparameter tuning, the original training corpus was partitioned to reserve 10% of the data as an independent validation set. This split ensures a reliable assessment of generalization and mitigates the risk of overfitting during the fine-tuning process.

3.1.3. Hyperparameter tuning

To improve the accuracy and the generalization of the model hyperparameter search is run through Bayesian optimization, due to its balance between exploitation (choosing values that performed well previously) and exploration (trying new values to discover better configurations) and its greater efficiency with respect to grid and random search.

As the dataset size grows and models get more complex, this process becomes more expensive since the evaluation of a single set of hyperparameters often requires retraining a new model. Moreover, although the validation error decreases as the training dataset size increases, the ranking of hyperparameter configurations is still relatively consistent across dataset subsets. So, to make this process faster, the training and the validation set previously obtained are subsampled by taking from them respectively 200 and 100 samples. This search is done for 20 trials, based on accuracy, the score used for the evaluation [1], on the following hyperparameters:

- **Learning rate** is a measure of how fast the machine learning model performs data training.
  With a high learning rate, then your system will rapidly adapt to new data, but it will also tend to quickly forget the old data. With a low learning rate, the system will learn more slowly, but it will also be less sensitive to noise in the new data.
- **Number of train epochs** is the number of training iterations over the dataset [7].
  With few epochs, the system can save computation, but it also means we have little training information. With many epochs, the networks can converge during evaluation [8].
- **Per device train batch size** refers to the number of training examples used in one iteration on training process [9].
  A small batch size ensures that each training iteration is very fast. Although a large batch size will give a more precise estimate of the gradients, in practice this does not matter much since the direction of the true gradients do not point precisely towards the optimum [7].
- **Weight decay** works by adding a penalty for large weights to the loss function, encouraging the model to learn more generalizable features.
  Setting it too high can make it difficult for the model to capture important patterns in the data. Setting it too low may not provide sufficient regularization. [10].
- **Warmup ratio** is the ratio of total training steps where learning rate will warm up.
  Learning rate warm-up is a technique used to stabilize the initial training phase. This allows the model to avoid the unstable gradients that can occur at the beginning of training, particularly with larger models [11].

In Table 1, it is possible to see the hyperparameters search space and the best hyperparameters set. Notice that for the number of train epochs are included only the integers in the interval, while for the learning rate and for the last two hyperparameters also the floats are taken in account. Additionally, for the learning rate the value is sampled from the range in a logarithmic scale, while for the other hyperparameters the linear domain is used.

Table 1
SubTask A: Hyperparameters Search Space and Best Hyperparameters Set

Hyperparameters	Search Space	Best
learning rate	[1e-6, 5e-5]	2.336911157693361e-05
number of train epochs	[2,5]	5
per device train batch size	{4, 8}	4
weight decay	[0.0, 0.3]	0.20388579400869386
warmup ratio	[0.0,0.2]	0.18924653336093455
ACCURACY		**0.92**

3.2. Subtask B

3.2.1. Model Selection and Architecture

For the segmentation task, Light Gradient Boosting Machine (LightGBM) Regressor is chosen[2]. LightGBM [12] is a newer version of Gradient Boosting Decision Tree Algorithm (GBDT). This model is selected because it speeds up the training process of GBDT by up to 20 times while achieving almost the same accuracy [12].

In particular, the implementation of Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) alongside efficient split finding techniques marks LightGBM as a highly efficient and effective algorithm for regression problems [13]. Since Subtask B requires to map to the target numerical values (the index of the character where the machine generated part of text starts), LightGBM Regressor would be a good back bone for this segmentation task.

3.2.2. Data Processing and Experimental Setup

Textual inputs are encoded by using a Sentence-BERT model [14] as a frozen feature extractor. More precisely, all-MiniLM-L6-v2[3] is used because it is a more compact version of the BERT model from the MiniLM (Mini Language Model) and it retains much of the semantic and linguistic understanding of larger transformer models. MiniLM, being trained by using the contrastive learning approach, encourages the mapping of semantically similar sentences closer together in vector space [15].

For model validation and hyperparameter tuning, the original training corpus was partitioned to reserve 20% of the data as an independent validation set. This split provides a large chunk of data for the model to learn from while still reserving a reasonable amount for evaluation.

3.2.3. Hyperparameters tuning

As for Subtask A, hyperparameters search is run through Bayesian optimization. To make this process faster, the training set is subsampled selecting a subset with 20% of the samples, that is 16% of the provided training set.

Moreover, k-fold cross-validation with 3 folds is implemented on this subset to evaluate a model's performance and make sure that it is not dependent on a particular training-test split of the data.

Finally, early stopping is activated to stop training if the model does not improve for a certain number of rounds, saving time and avoiding overfitting [16]. For hyperparameter search, the number of stopping rounds is set to 50, while for training it is set to 100. This search is run for 50 trials, based on MAE (Mean Absolute Error), the score used for the evaluation [1], on the following hyperparameters:

[2]https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html
[3]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

- **Number of leaves** specifies the maximum number of leaves in each tree.
  The higher the value, the better the model's performance will be, with a trade-off of having a longer computational time. However, there is a threshold where the impact of adding more leaves will not have much additional impact on the model's performance or even have a negative impact due to overfitting [17].
- **Max depth** specifies the maximum depth of each decision tree.
  A lower value can help us prevent overfitting. However, a value that's too low can lead to an underfitting problem.
- **Learning rate** used for SubTask A.
- **Number of estimators** controls the number of trees in the ensemble.
  Generally, more trees are better. However, a point of diminishing returns is often reached, and overfitting occurs when there are too many trees [18].
- **Min child samples** specifies the minimum number of samples required in the leaf nodes.
  A higher value can help us prevent overfitting. However, a value that is too high can lead to an underfitting problem.
- **Subsample** (bagging fraction) controls what fraction of the data is used for each training iteration.
  Column sample by tree (feature fraction) controls the fraction of features (columns) to consider when building each tree.
  Lower values of these last two hyperparameters can help us prevent overfitting and lower the computational time. However, a value that is too low can lead to an underfitting problem [17].
- **Regularization alpha** and **regularization lambda** help contain overfitting by penalizing the excessive weights assigned to certain tree splits.
  These elements act as internal brakes, ensuring the model does not excessively memorize the training data [19].

In Table 2, it is possible to see the hyperparameters search space and the best hyperparameters set. Notice that for the all the hyperparameters the linear domain is used. Additionally, for the number of leaves, max depth, number of estimators, min child samples are included only the integers in the interval, while for the others also the floats are taken in account.

Table 2
SubTask B: Hyperparameters Search Space and Best Hyperparameters Set

Hyperparameters	Search Space	Best
number of leaves	[20, 150]	104
max depth	[3, 20]	7
learning rate	[0.01, 0.3]	0.025883831670545893
number of estimators	[100, 2000]	662
min child samples	[5, 100]	40
subsample	[0.5, 1.0]	0.8759855365611233
column sample by tree	[0.5, 1.0]	0.73998086648529
regularization alpha	[0, 1]	0.03013121841344469
regularization lambda	[0, 1]	0.6016304218749495
MAE		**100.2806**

4. Results

Finally, once the models are trained on their respective training sets, using the validation set for evaluation, they are run on the corresponding test sets and the resulting predictions are submitted.

For SubTask A, the model achieved an accuracy of 0.99, indicating that the model learned the training distribution extremely well. On the unseen test set, the model obtained an accuracy equal to 0.92. This

drop is expected and indicates that the model encounters data that differs slightly from the training distribution. This score therefore represents the true generalization performance of the system.

The high-test accuracy indicates that the model successfully captures linguistic differences between human and machine generated text within the dataset [20]. This system gained the 4th position in the leaderboard shown in Table 3 [1].

Table 3
SubTask A: Leaderboard

Position	Team Name	Accuracy
1	Gradient Descenders	0.945755
2	Kenji Endo	0.942649
3	UniTor	0.928774
4	**Nicla**	**0.924348**
5	Stochastic Gradient Descenders	0.921638

For SubTask B, the model achieved a MAE of 98.15 reflecting the model's ability to learn the structural and linguistic cues present in the training corpus. On the unseen test set, the model obtained a MAE of 102.04. This drop is expected and suggests that the unseen test data introduces patterns or variability not fully captured during training. This score suggests that the model generalizes reasonably well beyond the training distribution.

The results demonstrated that both models maintain stable performance when moving from validation data to unseen test data. This system gained the 5th position in the leaderboard shown in Table 4 [1].

Table 4
SubTask B: Leaderboard

Position	Team Name	Accuracy
1	Stochastic Gradient Descenders	52.54
2	MINDS	56.53
3	Gradient Descenders	62.66
4	UniTor	81.60
5	**Nicla**	**102.04**

5. Discussion

The systems we propose ranked 4th and 5th in Subtask A and Subtask B respectively, showing that they can perform the tasks of detecting and segmenting MGT but they can be improved further.

For SubTask A, accuracy alone does not fully characterize performance. Additional metrics would provide deeper insight into class specific behaviour. It is important to note that MGT Detection is sensitive to dataset composition. Detectors struggle in it if the texts come from a domain, a generator, or a language that the model has not seen during training [21].

Therefore, although the model performs well on the provided test set, further evaluation on out of distribution data is necessary to assess robustness. As large language models evolve rapidly, maintaining high detection performance in real world scenarios remains a challenging task.

For SubTask B, the results indicate that the model is able to approximate the transition point between human and machine generated text with reasonable precision. The task itself is challenging, as the boundary between human and machine text does not always correspond to clear lexical or syntactic shifts, making precise prediction difficult [22].

The model's performance may also be influenced by the composition of the training data. If the dataset contains limited stylistic diversity, the model may learn patterns that do not generalize perfectly to new domains or generators [23]. Future work could focus on expanding the dataset, incorporating more diverse writing styles, or exploring architectures specifically designed for boundary detection.

Declaration on Generative AI

During the preparation of this work, the author used:

- **Reverso** to Text Translation
- **Copilot** to Drafting content, Content enhancement, Paraphrase and reword, Citation management.
- **Chat GPT** to Drafting content, Content enhancement.

After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] G. Puccetti, A. Pedrotti, A. Esuli, Desegma-it at evalita 2026: Overview of the detection and segmentation of machine generated text in italian task, 2026.

[2] F. Cutugno, A. Miaschi, A. P. Aprosio, G. Rambelli, L. Siciliani, M. A. Stranisci, Evalita 2026: Overview of the 9th evaluation campaign of natural language processing and speech tools for italian, in: Proceedings of the Ninth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2026), CEUR.org, Bari, Italy, 2026.

[3] S. Fariello, G. Fenza, F. Forte, M. Gallo, M. Marotta, Distinguishing Human from Machine: A Review of Advances and Challenges in AI-Generated Text Detection, International Journal of Interactive Multimedia and Artificial Intelligence (2025). URL: https://www.researchgate.net/publication/387246195_Distinguishing_Human_From_Machine_A_Review_of_Advances_and_Challenges_in_AI-Generated_Text_Detection.

[4] V. Sanh, L. Debut, J. Chaumond, T. Wolf, DistilBERT: A Distilled Version of BERT: smaller, faster, cheaper and lighter, in: EMC^2: 5th Edition Co-located with NeurIPS 2019, 2020. URL: https://www.emc2-ai.org/assets/docs/neurips-19/emc2-neurips19-paper-33.pdf.

[5] T. O. Abiola, T. A. Bizuneh, O. J. Abiola, T. O. Oladepo, O. E. Ojo, G. Sidorov, O. Kolesnikova, CIC-NLP at GenAI detection task 1: Leveraging DistilBERT for detecting machine-generated text in English, in: Proceedings of the 1stWorkshop on GenAI Content Detection (GenAIDetect), 2025. URL: https://aclanthology.org/2025.genaidetect-1.29.pdf.

[6] HuggingFace, Text classification, 2022. URL: https://huggingface.co/docs/transformers/tasks/sequence_classification.

[7] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media, 2022. URL: https://0-lucas.github.io/digital-garden/99.-Books/Hands-on-Machine-Learning.pdf.

[8] R. Egele, F. Mohr, T. Viering, P. Balaprakash, The unreasonable effectiveness of early discarding after one epoch in neural network hyperparameter optimization, Neurocomputing (2024). URL: https://www.sciencedirect.com/science/article/pii/S0925231224007355.

[9] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. URL: http://www.deeplearningbook.org.

[10] P. Iusztin, N. Sen, J. Chaumond, H. Tahir, A. Gulli, J. Chaumond, H. Tahir, A. G. Gulli, LLM Engineer's Handbook, Sciendo, 2024. URL: https://doi.org/10.0000/9781836200062.

[11] A. Sarkar, Deep Learning Dynamics: The Science Behind AI Training: Exploring the Strategies, Challenges, and Innovations Shaping Modern AI Development, Sarkar, A., 2025.

[12] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, in: Advances in Neural Information Processing Systems, 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[13] C. Kahraman, S. C. Onar, S. Cebi, B. Oztaysi, A. C. Tolga, I. U. Sari, Intelligent and Fuzzy Systems: Intelligent Industrial Informatics and Efficient Networks Proceedings of the INFUS 2024 Conference, Volume 2, Springer Nature Switzerland, 2024.

[14] N. Reimers, I. Gurevych, Sentence-BERT: Sentence embeddings using Siamese BERT-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019. URL: https://aclanthology.org/D19-1410/.

[15] H. Arabnia, L. Deligiannidis, F. Shenavarmasouleh, S. Amirian, F. Mohammadi, Computational Science and Computational Intelligence: 11th International Conference, CSCI 2024, Las Vegas, NV, USA, December 11–13, 2024, Proceedings, Part V, Springer Cham, 2025.

[16] G. Kunapuli, Ensemble Methods for Machine Learning, Manning, 2023.

[17] L. Owen, Hyperparameter Tuning With Python: Boost your machine learning model's performance via hyperparameter tuning, Packt Publishing, 2022.

[18] A. van Wyk, Machine Learning with LightGBM and Python A Practitioner's Guide to Developing Production-ready Machine Learning Systems, Packt Publishing, 2022.

[19] D. Rodrigues, LEARN LIGHTGBM: Build Accurate Models with Scalable Machine Learning, StudioD21, 2025.

[20] A. Masih, B. Afzal, S. Firdoos, J. Mahmood, A. Ali, M. S. Abdulnabi, D. M. Balungu, Classifying human vs. AI text with machine learning and explainable transformer models, Scientific Reports (2025). URL: https://www.nature.com/articles/s41598-025-27377-z.

[21] Y. Wang, J. Mansurov, P. Ivanov, J. Su, A. Shelmanov, A. Tsvigun, C. Whitehouse, O. Mohammed Afzal, T. Mahmoud, T. Sasaki, T. Arnold, A. F. Aji, N. Habash, I. Gurevych, P. Nakov, M4: Multi-generator, Multi-domain, and Multi-lingual Black-Box Machine-generated Text Detection, in: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), 2024. URL: https://aclanthology.org/2024.eacl-long.83/.

[22] L. Kushnareva, T. Gaintseva, G. Magai, S. Barannikov, D. Abulkhanov, K. Kuznetsov, E. Tulchinskii, I. Piontkovskaya, S. Nikolenko, AI-generated Text Boundary Detection with Roft, in: 1st Conference on Language Modeling (COLM) 2024, 2024. URL: https://arxiv.org/pdf/2311.08349.

[23] K. Jiao, Q. Wang, L. Zhang, Z. Guo, Z. Mao, M-Rangedetector: Enhancing Generalization in Machine-Generated Text Detection through Multi-Range Attention Masks, in: Findings of the Association for Computational Linguistics: ACL 2025, 2025. URL: https://aclanthology.org/2025.findings-acl.469.pdf.