A. MODEL SUMMARY

A1. Background on you/your team

- Competition Name: LLM - Detect AI Generated Text
- Team Name: nlp team
- Private Leaderboard Score: 0.974994
- Private Leaderboard Place: 3rd place

- Name: Yevhenii Maslov
- Location: Kyiv, Ukraine
- Email: sqrt.evmaslov@gmail.com

- Name: Zhirui Zhou
- Location: Chengdu, Sichuan, China
- Email: evilpsycho42@gmail.com

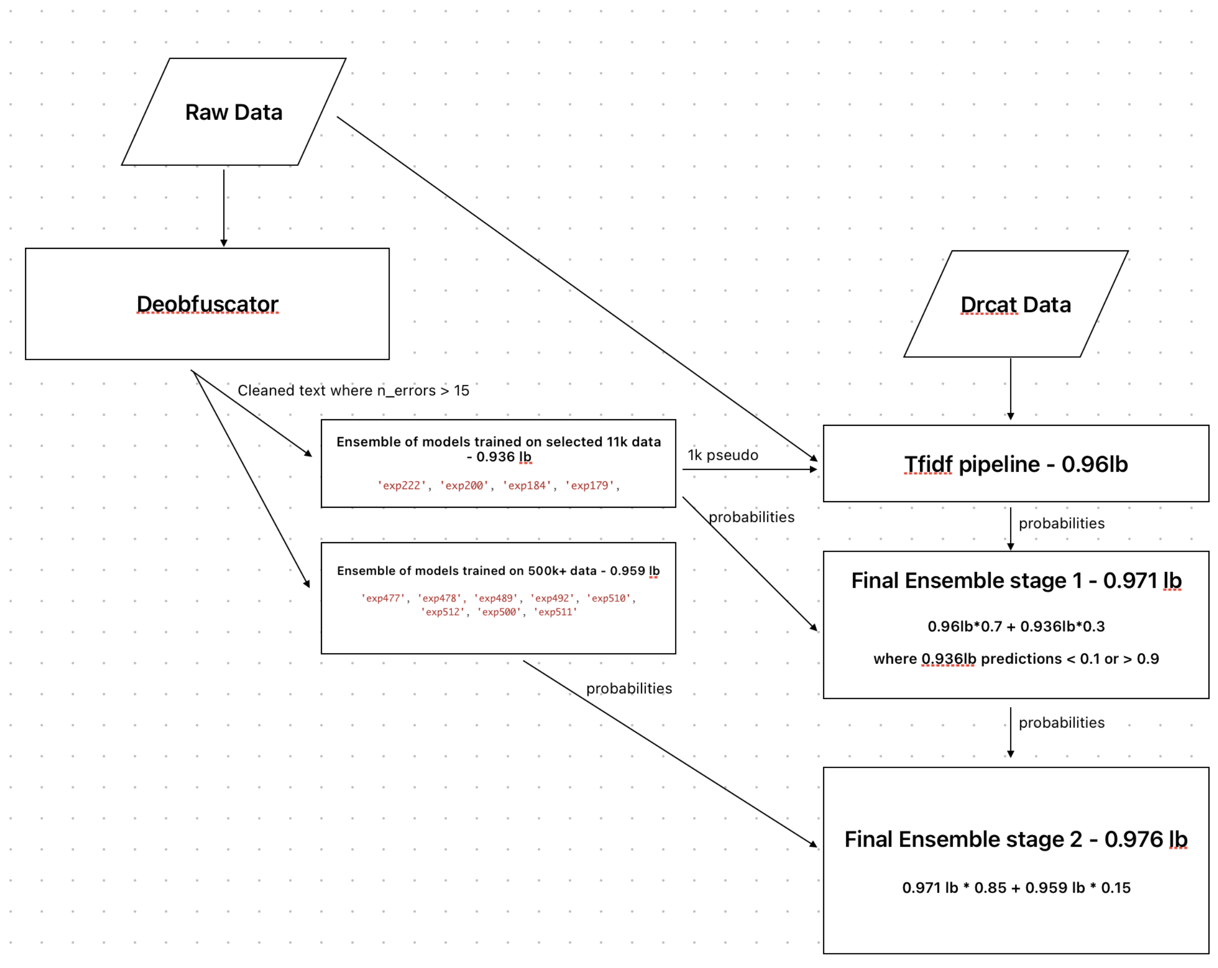
- Name: Ivan Isaev
- Location: Cologne, North Rhine-Westphalia, Germany
- Email: gm.ivan.isaev@gmail.com

- Name: Muhammad A.
- Location: Karachi, Sindh, Pakistan
- Email: m.ahmed.memonn@gmail.com

- Name: Dmitrii Khizbullin
- Location: Thuwal, Makkah Province, Saudi Arabia
- Email: dmitrii.khizbullin@gmail.com

A2. Summary

Our solution is a weighted average of the 12 deberta-v3-large models and tf-idf pipeline.



The first part of our solution consists of training transformer models to classify if LLM generated the text.

- We used a dataset of 11k human-written and LLM-rewritten persuade essays to train the first 4 models.
- The other 3 models were trained on the publicly available Pile subset (MIT license), SlimPajama dataset (Apache 2.0 license), and LLM-generated continuations for truncated samples from the first two datasets. We used 500k, 1m, and 1.13m samples to train the models.
- The remaining 5 models are finetuned versions of the previous 3 models on the 11k dataset.

As the second part of our solution, we trained a custom tokenizer on test data, made tf-idf features from tokenized sequences, and then fitted an ensemble of Naive Bayes, SGDClassifier, LGBMClassifier, and CatBoostClassifier on a combination of publicly available train data and pseudo labels obtained from transformers ensemble.

A3. Data Generation

We used the persuade dataset as a source for human-written essays. Then we asked different LLMs to rewrite each essay in a few different ways:

- Sentence-level. We used nltk.sent_tokenize() to split the essay into sentences and gave separate sentences to the LLM without any additional context.
- Essay-level. We gave a full essay to LLM and tuned the prompt to obtain an output sequence of approximately the same length as the original text.
- Partially rewritten essay. Same as sentence-level, but we randomly selected 30-80% of sentences to rewrite.

We generated around 200k samples this way using GPT-3.5, Mistral-7b, Llama-7b, and neural-chat-7b-v1. Also, we used almost all community-shared datasets.
We found that a lot of samples are too easy for the model to classify, and training with all data results in worse public and private scores, so we used the following algorithm to select training samples:

- Train the initial transformer using @alejopaullier data
- At each iteration, add samples that the previous model failed to predict correctly - 500 human-written and 500 generated, with the highest distance from the true label.
- Train a new model and repeat

After some number of iterations, we got the best public score of 0.927 (compared to 0.78 with all data) with an 11k subset. The ensemble of models on deobfuscated data has a 0.936 public score.

Inspired by @jsday96 post we generated continuations for a subset of Pile and SlimPajama datasets. We filtered out text that was too short/too long, contained code or math, non-English text, and had a high non-letters/letters ratio. We used vllm at this stage. We split sampling parameters into 3 scenarios depending on the temperature value and used random values for top_p/min_p and presence_penalty / frequency_penalty within bounds specified for each scenario. The models used and the amount of generated samples are presented in the table below.

Source	**Number of samples**
Pile and Slimpajama	560563
TheBloke/Llama-2-13B-chat-AWQ	55907
mncai/agiin-13.6B-v0.1	39007
upstage/SOLAR-10.7B-Instruct-v1.0	37520
HuggingFaceH4/zephyr-7b-beta	33195
yevheniimaslov/Mistral-7b-persuade	27733
mistralai/Mistral-7B-Instruct-v0.1	25550
OpenHermes-2.5-Mistral-7B	25283
mindy-labs/mindy-7b-v2	24689
TheBloke/Yi-34B-AWQ	21226
Weyaxi/OpenHermes-2.5-neural-chat-v3-3-Slerp	20658
microsoft/Orca-2-13b	18621
TheBloke/WizardLM-13B-V1.2-AWQ	17931
TheBloke/openchat_3.5-AWQ	17651
TheBloke/LMCocktail-10.7B-v1-AWQ	16900
TheBloke/WizardCoder-Python-34B-V1.0-AWQ	14329
TheBloke/Airoboros-L2-13B-2.1-AWQ	12292
HyperbeeAI/Tulpar-7b-v2	11439
Q-bert/Terminis-7B	11324
rishiraj/smol-7b	11202
TheBloke/CodeLlama-34B-AWQ	10795

cookinai/Valkyrie-V1	10525
mistralai/Mistral-7B-v0.1	10440
GPT3.5	9837
perlthoughts/Falkor-7b	9810
mistralai/Mistral-7B-Instruct-v0.2	9585
TheBloke/Nous-Hermes-2-Yi-34B-AWQ	9416
DopeorNope/SOLARC-M-10.7B	9020
Intel/neural-chat-7b-v3-3	8369
Cohere-command	5933
Sao10K/Frostwind-10.7B-v1	5717
TheBloke/Llama-2-70B-Chat-AWQ	4755
persuade	4724
persuade-generated	4602
meta-llama/Llama-2-7b-chat-hf	3929
teknium/OpenHermes-2.5-Mistral-7B	3776
moth	3530
TheBloke/StableBeluga2-70B-AWQ	3442
01-ai/Yi-6B-200K	3352
cognitivecomputations/dolphin-2.2.1-mistral-7b	1984
open-sourced-books	1071
TheBloke/Mythalion-13B-AWQ	443

The best single deberta-large is trained with 1m samples generated this way (~500k human-written and ~500k generated) and has 0.956 public and 0.967 private scores. The ensemble of the models has 0.959 public and 0.967 private scores.

A4. Training Method

We trained almost all our models with the same hyperparameters. We used 256 maximum sequence length (1512 for inference), 48-96 batch size, and 3 epochs for training. The exact hyperparameters that were used could be found in each model config file.

For pile/slimpajama dataset we used a random time shift as data augmentation, to train on different sequences each epoch and make a model pay more attention to local text features. It improved our ROC AUC by ~0.002.

Overall, it takes around 10 minutes for model training on the 11k dataset and 18-20 hours on the 1.2m dataset.

All models were trained using Nvidia A6000-Ada and A100. You can check the model's training details in the table below.

Experiment name	**Dataset Size**	**Training Time**	**Max length**	**Finetuning**	**Final Ensemble**	**Private/Public Score**
exp179	11k	10 minutes	256	-	+	0.836584/ 0.915014
exp184	11k	30 minutes	512	-	+	0.846938/ 0.912894
exp200	11k	10 minutes	256	-	+	0.842764/ 0.918333
exp222	11k	10 minutes	256	-	+	0.836628/ 0.917656
exp475	500k	510 minutes	256	-	-	0.953418/ 0.945229
exp477	11k	15 minutes	1024	Exp475, 1 layer+head	+	0.962718/ 0.951210
exp478	11k	15 minutes	1024	Exp475, 4 layers+head	+	0.958963/ 0.949333
exp489	1.04m	1050 minutes	256	-	+	0.967491/ 0.956442
exp492	11k	15 minutes	1024	Exp489, 1 layer+head	+	0.962499/ 0.956810
exp510	11k	15 minutes	1024	Exp489, 1 layer+head, seed42	+	0.965401/ 0.959107

exp512	11k	45 minutes	1024	Exp489, 1 layer+head, 3 epochs	+	0.960692/ 0.956288
exp500	1.138m	1260 minutes	256	-	+	0.961253/ 0.950405
exp507	1.04m	600 minutes	256	-	-	0.944087/ 0.945921
exp511	11k	45 minutes	1024	Exp507, 1 layer+head, 3 epochs	+	0.941812/ 0.949552

A5. Interesting findings

- The most important idea that sets us apart from others is the large and diverse dataset for transformer training
- During the deberta-large training with our dataset, the gradient exploding problem occurs too often. One solution is to clip the gradient norm, which results in a slightly worse score. We relaunched training several times without gradient clipping to get our best model.
- We used deobfuscator to correct errors in essays, but only if the number of typos was greater than 15. It improved the individual model score by 0.005-0.01.
- During inference, we ran our ensemble of transformers first, obtained the test set pseudo labels, and added 1k set to tfidf pipeline training. We used only samples in which the transformers were most confident (probabilities lower than 0.01 or higher than 0.99). Together with hyperparameters tuning this improved our private score to 0.927.
- We found that an ensemble of transformers does not improve the private score much, compared to the single best model.
- We used distance-based postprocessing: For each prompt_id, if the number of samples is greater than 1000, we fitted umap on tfidfs (the same as in tfidf-catboost pipeline, but per-prompt), calculated distance to 7 closest human-written and 7 generated samples, and scaled predictions by the ratio human_distance / generated_distance with clipping to (0.75, 1.25). It slightly improved public and private LB.

A6. Simple Features and Methods

We can reduce inference time to 20 minutes, by using a single deberta-v3-large with optuna-optimized head. This model achieves 0.976756 private and 0.956990 public scores.
The model score is better than our final ensemble, but we disregarded this model since our public lb didn't improve.

Head was optimized as following:

- We took 10k random samples from pile-pajama dataset
- Embeddings (after pooling layer) of the selected samples were extracted using exp489 model weights
- Head was initialized as a vector of shape 1024, using optuna.suggest_float() with bound (-1, 1)
- Predictions were calculated as the dot product between embeddings and head
- Score was calculated as ROC AUC between true labels and predictions
- We optimized weights of the head using 100 trials, saved best weights to the file
- Then we started a new set of trials, using bounds +-0.1 from the weights in the file
- We made 50 such iterations and took the head weights with the best objective metric

A7. Submission Execution Time

We measured the execution time of each component in our final ensemble. You can see the results in the table below.

Component	**Training Time**	**Inference Time**	**Private/Public Score**
Deobfuscator	34 minutes	40 minutes	-
Transformers Ensemble 1, lb 0.936	60 minutes	80 minutes	0.866882/0.936051
Transformers	3570 minutes	160 minutes	0.967873/0.959737

Ensemble 2, lb 0.959			
Tfidf Pipeline	180-240 minutes	180-240 minutes (training during inference)	0.927937/0.960957

Full submission takes around 7-8 hours to make predictions.