Binary Human-written vs Machine-generated Text Classification
NLP Course Project

Paolo De Angelis, Edoardo Merli, Giacomo Piergentili and **Lorenzo Scaioli**
Master's Degree in Artificial Intelligence, University of Bologna
{ paolo.deangelis7, edoardo.merli, giacomo.piergentili2, lorenzo.scaioli}@studio.unibo.it

Abstract

We tackled monolingual Subtask A of SemEval-2024 Task 8: Binary Human-Written vs Machine-Generated Text Classification. Given a full text, the task's goal is to determine whether it is human-written or machine-generated. We compared two DistilBERT baselines, two DeBERTaV3-Large models and a set of classical ML models that leveraged TF-IDF features on top of a custom tokenizer. We observed that more complex models (DeBERTaV3) performed better on the validation set, but worse on the test set, compared to the other models. We attribute this to the test data being out-of-distribution with respect to the training and validation data. In the final part of the report, we point out how regularization techniques could help to avoid this.

1 Introduction

The impressive capabilities of LLMs and their broad accessibility have brought, together with obvious benefits, noteworthy risks to society. Many of these threats have been analyzed in detail (Abdali et al., 2024), some of the most significant ones being: spreading fake news and misinformation, both voluntarily generated or caused by factual inconsistency in LLMs' answers; negatively impacting the education system by providing youngsters with an alternative to the struggle necessary for the true learning process to take place; violating copyright in academic papers by generating answers coming from copyright protected information they might have been trained on.

In this setting, being able to correctly distinguish between human-written and machine-generated text becomes an essential problem to confront. Since humans perform only slightly better than chance when classifying machine-generated vs. human-written text, there is a need to develop automatic systems to identify machine-generated text with the goal of mitigating its potential misuse.

We can distinguish the possible approaches to solve the task in two main groups: black-box and white-box approaches. In particular, in the black-box scenario, the access is limited to the output text produced by LLM given an arbitrary input. Conversely, in the white-box context, we gain access to the model's output probabilities for individual tokens. Some of the most used black-box approaches are Supervised Detection, Zero-shot detection, Retrieval-based detection, Watermarking and Feature-based detection (Wang et al., 2024b). Instead, a famous white-box approach is GLTR - a Giant Language model Test Room - that can help to detect machine-generated text by analyzing the probability of LLM to use certain words (Gehrmann et al., 2019).

In our study, we considered Subtask A (monolingual) of SemEval-2024 Task 8 (Wang et al., 2024a) as our primary reference and source of data. We developed black-box approaches to distinguish human-written and machine-generate text, in particular:

- baseline: DistilBERT with frozen backbone
- baseline: DistilBERT fully fine-tuned
- DeBERTaV3-Large fully fine-tuned
- DeBERTaV3-Large using LoRA
- custom tokenizer + TF-IDF + (Multinomial Naive-Bayes / SVM / SGD / Ensemble of the three)

Given that it wasn't feasible to train the biggest model (DeBERTaV3 fully fine-tuned) on the whole dataset, we trained two versions of each model (except for the one just mentioned): one using the full dataset to assess the impact of more data and one using a balanced subset of the data for a fair benchmark.

We observed that more complex models performed better on the validation set, but worse on

the test set. For instance: DeBERTaV3-Large-LoRA reached a 0.8787 validation accuracy but only 0.6039 test accuracy. On the flip side, a much simpler model like Multinomial Naive-Bayes on Tf-Idf features scored 0.6554 and 0.8622 on validation and test accuracy respectively. We attribute this phenomenon to the test data potentially being out-of-distribution with respect to the training and validation data.

2 Background

2.1 Low-Rank Adaptation (LoRA)

Fine-tuning allows to adapt a (Large) Language Model to a target domain or task. Still, it can be computationally very costly or infeasible to do so: the larger the model, the more layers to update, the slower the convergence time and GPU RAM required to fine-tune.

Low-Rank Adaptation (LoRA) (Hu et al., 2021) is a technique to achieve the same results of fine-tuning by updating a number of parameters that is just a small percentage with respect to the number of parameters of the model.

Regular fine-tuning can be described mathematically as follows:

$$\Delta W = -\alpha \nabla L_W$$
$$W_{ft} = W + \Delta W$$

with W being the pre-trained weights, ΔW the weights update computed during fine-tuning and $\Delta_f t$ the fine-tuned weights.

We can also keep the weights update matrix separate and compute the outputs using:

$$a = Wx + \Delta W x$$

Now, the main idea behind LoRA is that, while the weight matrices of a pre-trained model have full rank, the weights update matrices have a low "intrinsic dimension", i.e. can be represented by a low-rank matrix. Practically speaking, this means that we can decompose ΔW as follows:

$$\Delta W = W_B \cdot W_A$$
$$a = Wx + W_B(W_A x)$$

with $W_A : \mathbb{R}^d \to \mathbb{R}^r$ and $W_B : \mathbb{R}^r \to \mathbb{R}^d$. Most importantly, $r << d$, i.e. we can use a LoRA rank r that is much smaller than the input dimension of the layer. r in general is a tunable hyperparameter, balancing the trade-off between faster training/smaller computational requirements (lower r) and low-rank matrix capacity (higher r).

To be even more precise, $\Delta W x$ is then scaled by $\frac{\alpha}{r}$, where α is another hyperparameter (usually fixed to the first r we try and not tuned, to help reducing the need of retuning hyperparameters when we vary r). This scaling balances the relative importance of the LoRA contribution.

We used LoRA in order to be able to train DeBERTaV3-Large on the whole training dataset.

3 System description

In this section, we describe the models that we used to tackle the task.

3.1 Baseline: DistilBERT frozen and fully fine-tuned

As our baselines, we used the HuggingFace model "distilbert-base-uncased" (Sanh et al., 2020), in his version for sequence classification, i.e. with an added classification head on top (a simple fully connected layer).

We tried to either use the language model backbone frozen and only train the head, or to also fine-tune the backbone.

A more detailed view of the architecture can be found in figure 5 of Appendix A.

3.2 DeBERTaV3-Large fully fine-tuned and LoRA

We used the HuggingFace model "deberta-v3-large" (He et al., 2023), in his version for sequence classification, i.e. with an added classification head on top, similarly to DistilBERT. This is our heavier, highest-capacity model, intending to see how performance improved as we scaled to larger Transformer models.

In this case, we tried to either fully fine-tune the whole backbone, which constrained us to only use a subset of the training set, given our limited computing resources, or to use Low-rank Adaptation (LoRA, described in Section 2) to still adapt the backbone to the task while being able to leverage all the training data. In both cases, the head is trained from scratch.

A more detailed view of the architecture can be found in figure 5 of Appendix A.

3.3 BPE tokenizer + TF-IDF + ML algos

This third model wants to be a more “classical" approach to sentence classification. This pipeline works as follows:

- First, we fit a Byte-Pair Encoding tokenizer (Sennrich et al., 2016) to the target dataset (validation or test set) vocabulary.
- We use this custom tokenizer to tokenize both the training and target datasets.
- Then, we vectorize the tokenized texts using Tf-Idf. Similarly to what we did for the tokenizer, the vocabulary is that of the target set.
- Lastly, we used the computed Tf-Idf features as inputs for one of the following three ML algorithms: Multinomial Naive-Bayes, SVM or SGD. We also measured the performance of an ensemble of the best-hyperparameters model of each one of them.

4 Data

The data for the task is an extension of the M4 dataset (Wang et al., 2024b). All three splits (train, validation, test) are approximately balanced (47.10%, 50%, 52% respectively). Moreover, the machine-generated texts are created by the following LLMs: DaVinci, ChatGPT, Dolly, Cohere for the training set, Bloomz for the validation set. (as shown in Figure 1). Interestingly, the training and validation sets have sentences generated by different LLMs; we believe the idea behind this choice was to assess the ability of the model to learn to detect text generated by unseen LLMs, i.e. to generalize over the entire possibilities of generated text.

Texts from both the training and validation sets come from different sources, namely WikiHow, Wikipedia, Reddit, ArXiv and Pearread. Notably, the distribution of these sources is not identical between the positive and negative labels in the training set (see Figure 2). For the versions of models trained on the subset of data, we sampled the data such that it would be balanced with respect to the “source" field.

Unfortunately, we don’t have information regarding the test set LLMs, except if they are machine-generated or human-written.

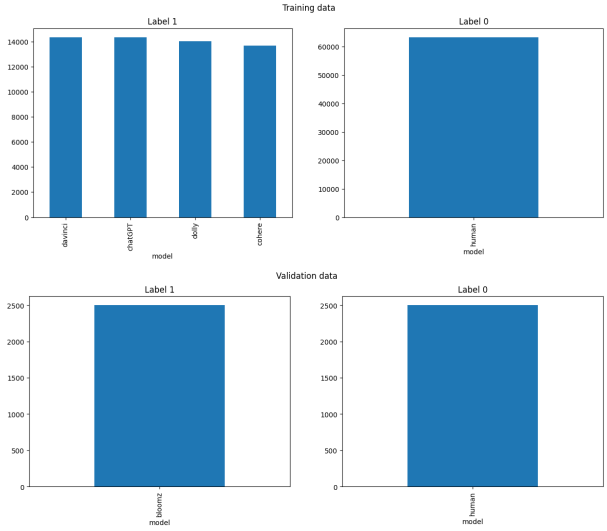


Figure 1: Distribution of LLMs used to generate training and validation data, together with respective amount of human-written texts

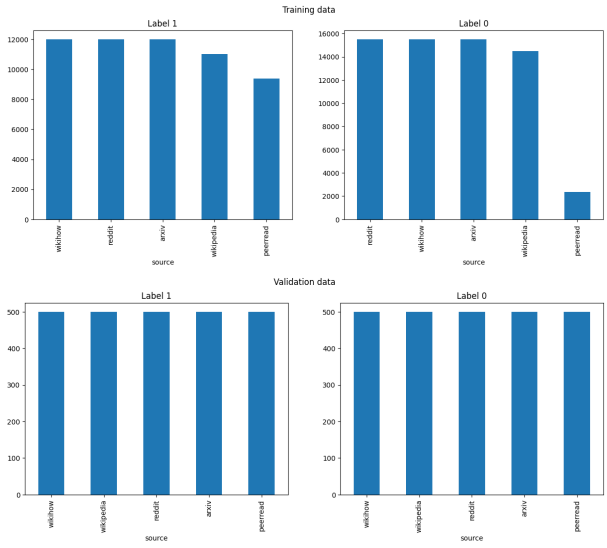


Figure 2: Distribution of sources for positive and negative labels in the training and validation sets. Training data is not balanced with respect to source.

More precise statistics about the data (for example, sentence length distribution across datasets) can be found on the challenge GitHub page (see section 8) and on the notebook *“data_exploration.ipynb"* of our GitHub repository (linked in section 8).

With regards to pre-processing steps operated on the data before feeding it to the model, we limited ourselves to tokenizing the texts for DistilBERT and DeBERTa, with a maximum length of the tokenized sequence fixed to 512 as a good tradeoff between document content and training feasibil-

ity. For the Tf-Idf pipeline, we didn't process the texts before feeding them to the pipeline (which tokenizes them "internally", as described in Section 3).

5 Experimental setup and results

5.1 Hyper-parameters

Each model was trained in two versions: one using the full dataset and one using a balanced subset of the data. This was done since it wasn't feasible to train the biggest model (DeBERTaV3 fully fine-tuned) on the whole dataset, but we still wanted to also have a fair benchmark to compare models.

The models, after several rounds of testing different configurations and hyper-parameters, were trained with the following hyperparameters respectively:

DistilBERT frozen: epochs: 3, batch size: 32, learning rate backbone: 0 (frozen), learning rate classification head: 2e-2, learning rate schedule: cosine decay.

DistilBERT fully fine-tuned: same as above, with the exception of learning rate backbone: 1e-5.

DeBERTaV3-Large fully fine-tuned: epochs: 3, batch size: 1, gradient accumulation steps: 4, learning rate backbone: 5e-6, learning rate classification head: 1e-2, learning rate schedule: cosine decay.

DeBERTaV3-Large LoRA: epochs: 1, batch size: 4, learning rate backbone: 8e-5, learning rate classification head: 8e-5, learning rate schedule: cosine decay, LoRA rank: 64, LoRA alpha: 16.

Tf-Idf pipeline: we performed a hyperparameter grid search for each one of the ML models. We point out to the notebook *"custom-tokenizer_tf-idf.ipynb"* of the GitHub repository for the extensive list of the respective hyper-parameters search spaces.

All four neural network models were trained using the AdamW optimizer on 2 x 16GB T4 GPUs in parallel (using data parallelism) on the Kaggle platform.

5.2 Metrics

The metrics used in the challenge to evaluate the models are accuracy and f1-score.

We run each model-dataset pair on the five following seeds: 42, 91, 184, 333, 647. We then computed the mean and standard deviation for each one of the seeds and configurations. The results can be seen in Table 1.

Below, we report a plot for a visual interpretation of the results. The plots for the other metric-dataset configurations can be found in Appendix B, figure 6.

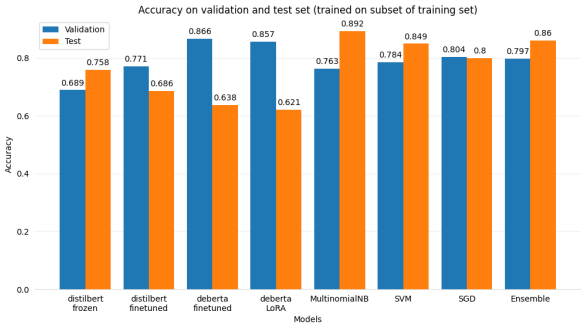


Figure 3: Accuracy on validation (blue) and test (orange) set for each model, trained using a balanced subset of the training set (in order to compare DeBERTaV3-large fully fine-tuned as well).

6 Discussion

Figure 3 highlights a trend where the Transformer models (first four columns) progressively improve in accuracy as we scale their size and complexity, but proportionally perform worse in the test set.

We attribute this phenomenon to the test data potentially being out-of-distribution with respect to the training and validation data. If this were the case, the high capacity of these models would enable them to reach a really low bias, at the expense of an increase of variance that would affect performance once the data would start shifting from the one they were trained on. This could also be the case because, given the nature of the task (detecting machine-generated data no matter which technique used to generate it), it makes sense that organizers wanted to evaluate performance on a really different dataset. Standard regularization techniques to reduce variance could be: collecting more diverse data, parameter norm penalties, label smoothing and dropout.

If we focus only on the validation data, the two DeBERTaV3 models outperform all others in both accuracy and F1-score across the two possible training dataset configurations.

With respect to the test data instead, the Multinomial Naive Bayes classifier on the Tf-Idf features performs best across all metrics and configurations. In general, the ML + Tf-Idf models have more robust performance across target datasets.

Future work As our explanation for the inconsistent performance of the Transformer models is just a hypothesis, it would be appropriate to validate it numerically, by measuring the distances between the distributions of texts of the test split with respect to the training and validation splits. Some possible alternatives could be: using the Jaccard similarity on the raw texts or a metric in embedding space like the cosine similarity, with the embeddings computed using Tf-Idf, Word2Vec or Transformers. Measuring distance pairwise over all documents would then be enough to compute distances across groups of texts.

To improve the performance of Transformer models instead, future efforts could try to tackle their sequence length bottleneck: we truncated the number of tokens fed to the models to a maximum length of 512, but as we can see from figure 4 below, the word counts (which is a lower bound on the token count) distribution presents non-negligible frequencies of texts ranging even up to double that sequence length.

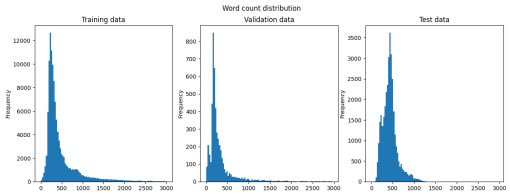


Figure 4: Word count distributions for the three dataset splits

A way to approach this without affecting training performance or having to scale the hardware resources would be to run the model at inference time on all contiguous 512 token portions of the input and then average the prediction.

7 Conclusion

In this report, we summarized the results we obtained from tackling the monolingual Subtask A of SemEval-2024 Task 8: Binary Human-Written vs Machine-Generated Text Classification. We trained four black-box models (two DistilBERT, two DeBERTaV3-Large) and four classical ML models (Multinomial Naive-Bayes, SVM, SGD, ensemble of the three) on top of Tf-Idf features. In the validation set, DeBERTaV3-Large reached 0.8787 accuracy, while on the test set, Multinomial Naive-Bayes on Tf-Idf features achieved an accuracy of 0.8622. While both values are really promising, no model scored high on both validation and test accuracy: the two models respectively got 0.6039 test accuracy and 0.6554 validation accuracy, but this trend holds also for the other models.

We hypothesized that the test data could be out-of-distribution with respect to the training and validation data, with the more complex models reaching low bias and high variance, and the simpler ones the opposite.

8 Links to external resources

- Github repo: link
- Challenge and dataset: SemEval2024 - task 8

References

Sara Abdali, Richard Anarfi, CJ Barberan, and Jia He. 2024. Decoding the ai pen: Techniques and challenges in detecting ai-generated text.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. Gltr: Statistical detection and visualization of generated text.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, and su. 2024a. 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 2041–2063, Mexico City, Mexico. Association for Computational Linguistics.

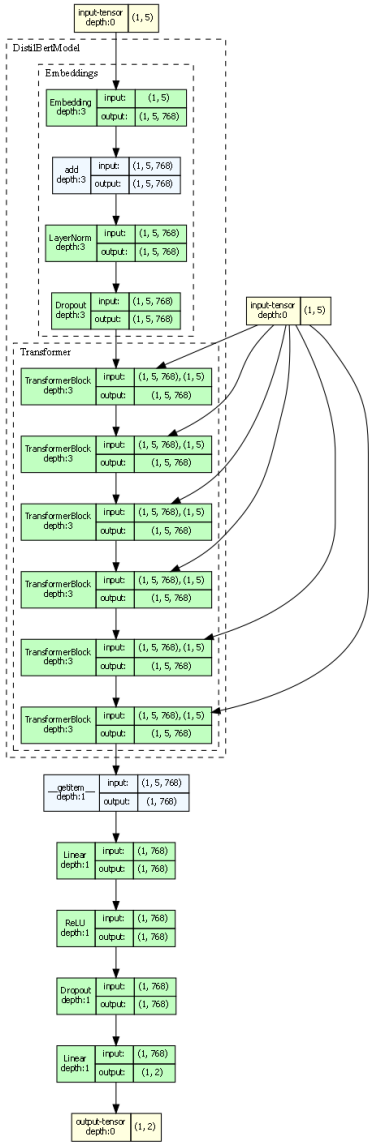
Yuxia Wang, Jonibek Mansurov, Petar Ivanov, and Jinyan Su. 2024b. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection.

Validation Set	**Accuracy**		**F1-score**		**Accuracy (subset)**		**F1-score (subset)**	
Model	Mean	Std	Mean	Std	Mean	Std	Mean	Std
distilbert-frozen (baseline)	0.7146	0.0015	0.7051	0.0059	0.6886	0.0055	0.6785	0.0111
distilbert-finetuned (baseline)	0.7959	0.0057	0.7834	0.0130	0.7706	0.0096	0.7504	0.0136
debertav3-large-finetuned	-	-	-	-	**0.8657**	0.0096	**0.8518**	0.0131
debertav3-large-LoRA	**0.8787**	0.0082	**0.8748**	0.0101	0.8566	0.0037	0.8486	0.0062
tfidf-multinomialNB	0.6554	-	0.7207	-	0.763	-	0.7427	-
tfidf-SVM	0.8634	-	0.8747	-	0.7844	-	0.7425	-
tfidf-SGD	0.8388	0.0033	0.8575	0.0024	0.8037	0.0325	0.7789	0.0470
tfidf-ensemble	0.8417	0.0029	0.8592	0.0021	0.7974	0.0123	0.7671	0.0187
random-classifier	0.4968	-	0.4962	-	0.512	-	0.51	-
majority-classifier	0.5000	-	0.0	-	0.5000	-	0.0	-

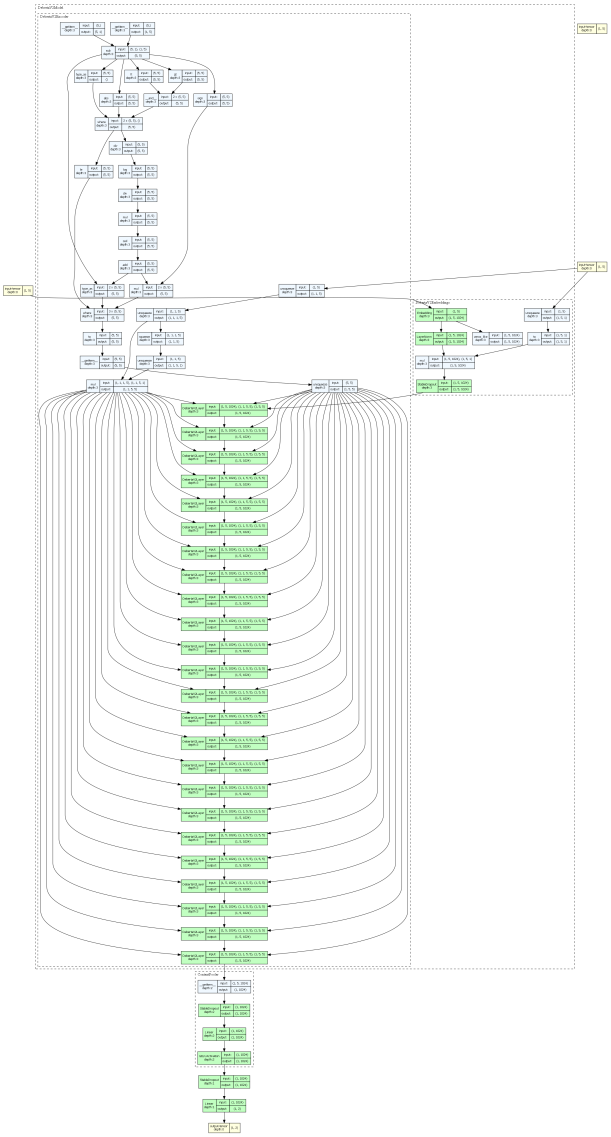
Test Set	**Accuracy**		**F1-score**		**Accuracy (subset)**		**F1-score (subset)**	
Model	Mean	Std	Mean	Std	Mean	Std	Mean	Std
distilbert-frozen (baseline)	0.7620	0.0205	0.8058	0.0125	0.7581	0.0092	0.8069	0.0066
distilbert-finetuned (baseline)	0.7276	0.0134	0.7881	0.0073	0.6861	0.0106	0.7638	0.0063
debertav3-large-finetuned	-	-	-	-	0.6376	0.0258	0.7432	0.0140
debertav3-large-LoRA	0.6039	0.0176	0.7259	0.0088	0.6213	0.0379	0.7355	0.0197
tfidf-multinomialNB	**0.8622**	-	**0.8727**	-	**0.8924**	-	**0.8878**	-
tfidf-SVM	0.721	-	0.7782	-	0.8489	-	0.8495	-
tfidf-SGD	0.6552	0.0049	0.7464	0.0024	0.7996	0.0135	0.8136	0.0060
tfidf-ensemble	0.7189	0.0003	0.7793	0.0001	0.8599	0.0030	0.8607	0.0042
random-classifier	0.5018	-	0.5149	-	0.4996	-	0.5142	-
majority-classifier	0.4748	-	0.0	-	0.4748	-	0.0	-

Table 1: Results for the implemented models on validation and test set. (subset) is used to identify the version trained on the balanced subset of the training set.

A Model architectures



DistilBERT



DeBERTa-V3-Large

Figure 5: Model architecture of DistilBERT and DeBERTa-V3-Large

B Plots

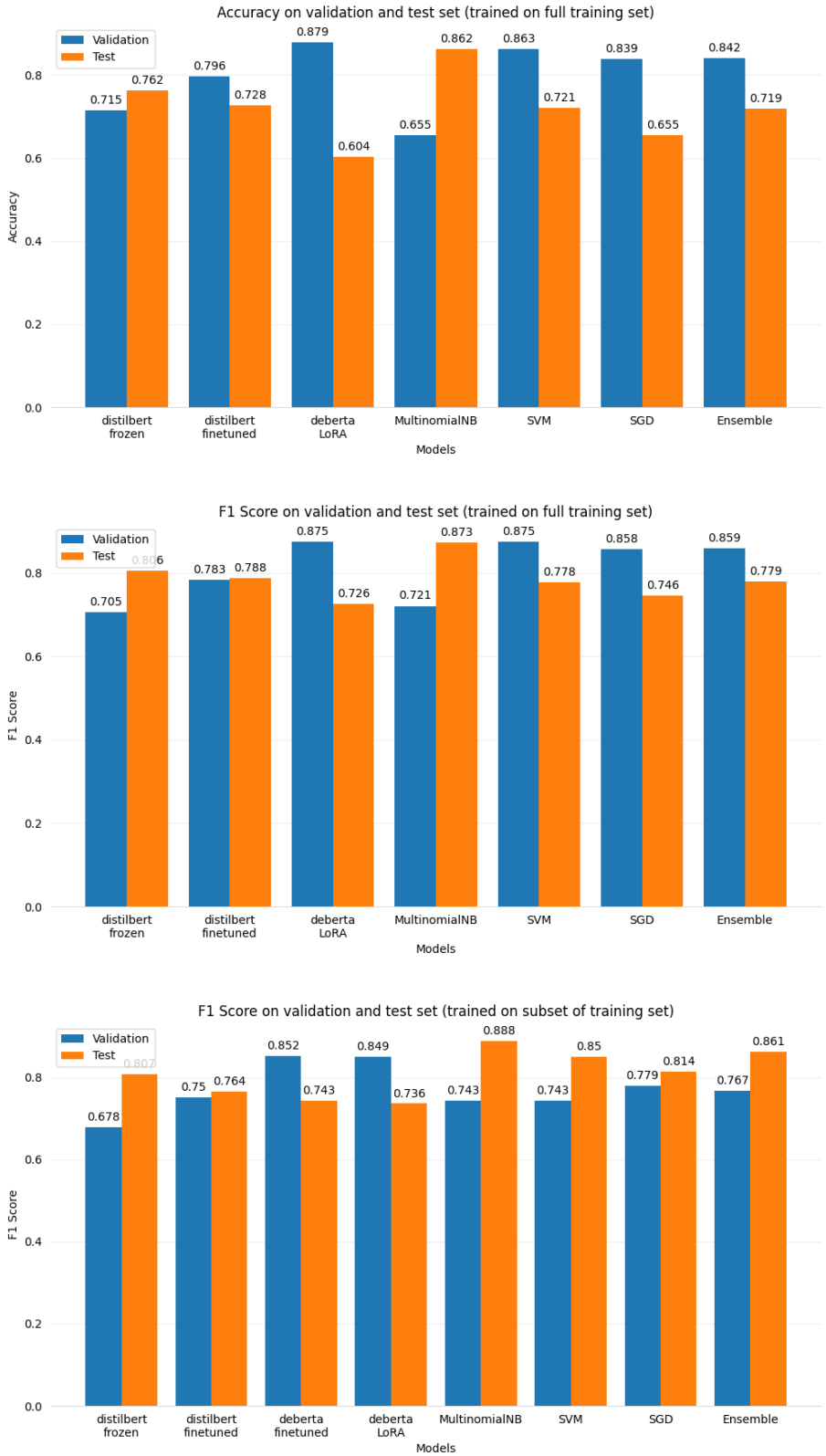


Figure 6: Remaining bar plots of the different models for accuracy and F1-score, varying the training dataset size (full or subset).