STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXX**, Number 1-2, 2025
DOI: 10.24193/subbi.2025.03

BERT-BASED AUTHORSHIP ATTRIBUTION ON THE ROMANIAN DATASET CALLED ROST

SANDA-MARIA AVRAM

Abstract. Having been around for decades, the problem of authorship attribution remains a current focus. Some of the more recent instruments used are the pre-trained language models, the most prevalent being BERT. Here we used such a model to detect the authorship of texts written in the Romanian language. The dataset used is highly unbalanced, i.e., significant differences in the number of texts per author, the sources from which the texts were collected, the period in which the authors lived and wrote these texts, the medium intended to be read (i.e., paper or online), and the type of writing (i.e., stories, short stories, fairy tales, novels, literary articles, and sketches). The results are better than expected, sometimes exceeding 87% macro-accuracy.

1. Introduction

The problem of automated Authorship Attribution (AA) has been studied for decades and remains highly relevant today. It is defined as the task of determining the author of an unknown text based on its textual characteristics [1]. We note that, while traditional approaches to AA often employed artificial intelligence (AI) methods using simple classifiers (such as linear SVMs or decision trees) and classical feature sets like bag-of-words or character n-grams [2, 3], the field has evolved in recent years. The adoption of deep neural networks in natural language processing (NLP) has led to their application in authorship identification as well. More recently, pre-trained language models—in particular BERT and GPT-2—have been used for fine-tuning and accuracy improvements in AA tasks [2, 4, 5].

Received by the editors: 25 June 2025.

2010 *Mathematics Subject Classification.* 68T50.

1998 *CR Categories and Descriptors.* I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *Text Analysis*; I.2.6 [**Artificial Intelligence**]: Learning – *Induction*.

Key words and phrases. Authorship Attribution, BERT, ROST.

This paper investigates the effectiveness of a Romanian pre-trained BERT model for authorship attribution. We specifically address the challenges posed by dataset imbalance, diversity of text sources, and genre variability.

The paper is organized as follows:

Section 2: reviews related work, especially methods employing artificial intelligence for authorship attribution;
Section 3: details the dataset preparation and preprocessing steps for the Romanian BERT model;
Section 4: presents implementation details and experimental results;
Section 5: concludes with final remarks and future directions.

2. Related work

The strategies used for addressing the AA problem are multiple. Authors of [4] grouped the main approaches into 4 classes:

Ngram: -includes character n-grams, parts-of-speech and summary statistics as shown in [6, 7, 8, 9];
PPM: - uses Prediction by Partial Matching (PPM) compression model to build a character-based model for each author, with works presented in [10, 11];
BERT: - combines a BERT pretrained language model with a dense layer for classification, as in [12];
pALM: - the per-Author Language Model (pALM), also using BERT as described in [13].

Here, we will focus on the Ngram and BERT classes.

2.1. Ngram: AI methods on ROST dataset.

In [14] we introduced a dataset, named *ROmanian Stories and other Texts* (ROST), consisting of Romanian texts that are stories, short stories, fairy tales, novels, articles, and sketches. We collected 400 such texts of different lengths, ranging from 91 to 39195 words. We used multiple AI techniques for classifying the literary texts written by multiple authors by considering a limited number of speech parts (prepositions, adverbs, and conjunctions). The methods we used were Artificial Neural Networks, Support Vector Machines, Multi-Expression Programming, Decision Trees with C5.0, and k-Nearest Neighbour.

For the tests performed in [14], the 400 Romanian texts were divided into training (50%), validation (25%), and test (25%) sets as detailed in Table 1. Some of the 5 aforementioned methods required only training and test sets. In such cases, we concatenated the validation set to the training set.

A numerical representation of the dataset was built as vectors of the frequency of occurrence of the considered features. The considered features were

TABLE 1. List of authors; the number of texts and their distribution on the training, validation, and test sets

#	**Author**	No. of texts	TrainSet size	ValidationSet size	TestSet size
0	Ion **Creangă**	**28**	14	7	7
1	Barbu Şt. **Delavrancea**	**44**	22	11	11
2	Mihai **Eminescu**	**27**	15	6	6
3	Nicolae **Filimon**	**34**	18	8	8
4	Emil **Gârleanu**	**43**	23	10	10
5	Petre **Ispirescu**	**40**	20	10	10
6	Mihai **Oltean**	**32**	16	8	8
7	Emilia **Plugaru**	**40**	20	10	10
8	Liviu **Rebreanu**	**60**	30	15	15
9	Ioan **Slavici**	**52**	26	13	13
	TOTAL	**400**	**204**	**98**	**98**

TABLE 2. Names used in [14] to refer to the different dataset representations and their shuffles.

#	**Designation**	**Features to Represent the Dataset**	**Shuffle**
1	**ROST-P-1**	prepositions	#1
2	**ROST-P-2**	prepositions	#2
3	**ROST-P-3**	prepositions	#3
4	**ROST-PA-1**	prepositions and adverbs	#1
5	**ROST-PA-2**	prepositions and adverbs	#2
6	**ROST-PA-3**	prepositions and adverbs	#3
7	**ROST-PAC-1**	prepositions, adverbs and conjunctions	#1
8	**ROST-PAC-2**	prepositions, adverbs and conjunctions	#2
9	**ROST-PAC-3**	prepositions, adverbs and conjunctions	#3

inflexible parts of speech (IPoS). Three different sets of IPoS were used. First, only prepositions were considered, then adverbs were added to this list, and finally, conjunctions were added as well. Therefore, three different representations of the dataset (of the 400 texts) were obtained. For each dataset representation (i.e., corresponding to a certain set of IPoS), the numerical vectors were shuffled and split into training, validation, and test sets as detailed in Table 1. This process (i.e., shuffle and split 50%–25%–25%) was repeated three times. We, therefore, obtained different dataset representations, which were referred to as described in Table 2.

All these representations of the dataset as vectors of the frequency of occurrence of the considered feature lists can be found as text files at reference [15].

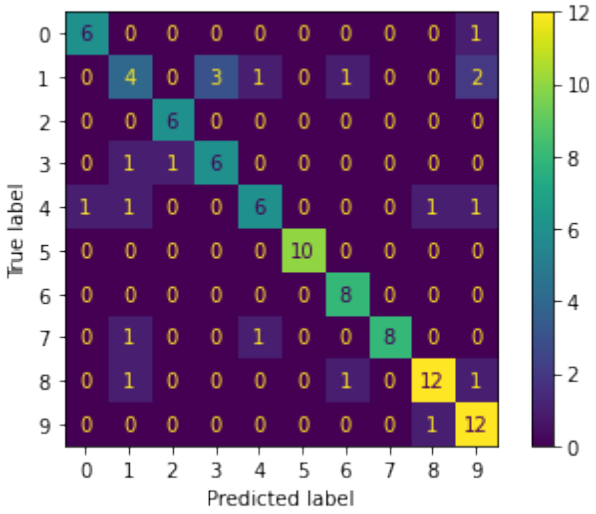

FIGURE 1. Confusion matrix on MEP's best results. The numbers from 0 to 9 are the codes given to our authors, as specified in the first column of Table 1.

These files contain feature-based numerical value representations for different texts on each line. The last column of these files contains numbers from 0 to 9 corresponding to the author, as specified in the first column of Table 1.

The best value obtained amongst all aforementioned AI methods was on ROST-PA-2 by using MEP, with a test error rate of 20.40%. This means that 20 out of 98 tests were misclassified. The obtained *Confusion Matrix* is depicted in Figure 1. The *macro-accuracy* value obtained was 80.94%. The Python code for building the *Confusion Matrix* and calculating the *macro-accuracy* value is provided at [16].

Bidirectional Encoder Representations from Transformers (BERT) is a language representation model designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers [17]. It was first introduced in October 2018 by a team from Google and is now used as a machine learning framework for natural language processing (NLP) tasks (i.e., classification, entity recognition, question answering, etc.).

BERT is based on multiple techniques that preceded it, starting from *RNN* and *LSTMs* to *Attention* and *Transformers*. Starting from 2015, *Recurrent Neural Networks* (RNNs) were used for author identification [18]. However,

RNNs have a problem with remembering long-term dependencies (over ten sequences) [19]. *Long Short Term Memory* (LSTM) architecture [20] is a special kind of RNN, introduced to overcome this weakness of the traditional RNN.

RNN encoder-decoder [21] architectures were used mostly for translations. In this context, the encoder translates the input into a "codified" vector, which is then passed to the decoder, which translates it again into the output language. The input and output vectors do not need to have the same length. The challenge with the encoder-decoder approach is that it has to transform the input sentence into a fixed-length vector that becomes a bottleneck. Therefore, such a model has difficulty translating long sentences. The *attention* technique was introduced in [22, 23], through which instead of passing the last hidden state of the encoding stage, the encoder passes all the hidden states to the decoder. This allows access to parts of the source sentence that are relevant to predicting a target word. In 2017, the concept of *Multi-Head Self Attention* was introduced in the paper "Attention Is All You Need" [24]. This technique is called *transformers* and uses self-attention for language understanding, parallelization for better translation quality, and attention to integrate the relevance of a set of values (information) based on some keys and queries. Google proposed BERT [17] in 2018, which uses a bi-directional transformer. Due to its complexity, more data is required for training such models. Therefore, there are currently many pretrained models in different languages.

BERT was used for AA in [25], while the performance of BERT was compared with other techniques [18, 13, 4] for solving the AA problem. Work in [12] concludes that BERT is the highest-performing AA method.

2.2. **Comparing methods.** The multitude of AA approaches makes it difficult to have a unified view of the state-of-the-art results. In [4], authors highlight this challenge by noting significant differences in:

: Datasets
- size: small (CCAT50, CMCC, Guardian10), medium (IMDb62, Blogs50), and large (PAN20, Gutenberg);
- content: cross-topic ($\times_t$), cross-genre ($\times_g$);
- imbalance (imb): i.e., the standard deviation of the number of documents per author;
- topic confusion (as detailed in [6]).

: Performance metrics
- type: accuracy, F1, c@1, recall, precision, macro-accuracy, AUC, R@8, and others;
- computation: even for the same performance metrics, there were different ways of computing them.

: Methods

- the feature extraction method:
  - Feature-based: n-gram, summary statistics, co-occurrence graphs;
  - Embedding-based: char or word embedding, transformers;
  - Feature and embedding-based: BERT.

The work presented in [4] tries to address and "resolve" these differences, bringing everything to a common denominator by using *macro-accuracy* as a metric.

The overall accuracy, also known as *micro-accuracy* or *micro-averaged accuracy*, weights the accuracy for each sample (text) equally. The *macro-accuracy* metric, also known as *macro-averaged accuracy*, is regarded to be more accurate in the case of imbalanced datasets, as it is computed by weighting equally the accuracy for each class (author).

A formal description of computing micro- and macro-accuracy was provided by Jacob Tyo in [26] and is described next:

$$\begin{aligned} N &= \text{total number of texts} \\ M &= \text{number of authors} \\ N_i &= \text{number of texts for author } i \\ C_i &= \text{number of texts correctly predicted for author } i \\ \mathcal{A}_{micro} &= \text{micro-averaged accuracy} \\ \mathcal{A}_{macro} &= \text{macro-averaged accuracy} \end{aligned}$$

$$\mathcal{A}_{micro} = \frac{1}{N} \sum_{i=1}^{M} C_i$$

$$\mathcal{A}_{macro} = \frac{1}{M} \sum_{i=1}^{M} \frac{C_i}{N_i}$$

The results of the state of the art as presented in [14] are shown in Table 3.

TABLE 3. State of the art *macro-accuracy* of authorship attribution models. Information collected from [4] (Tables 1 and 3). *Name* is the name of the dataset; *No. docs* represents the number of documents in that dataset; *No. auth* represents the number of authors; *Content* indicates whether the documents are cross-topic ($\times_t$) or cross-genre ($\times_g$); *W/D* stands for *words per document*, representing the average length of documents; *imb* represents the *imbalance* of the dataset measured by the standard deviation of the number of documents per author.

Dataset						**Investigation Type**			
Name	**No. Docs**	**No. Auth**	**Content**	**W/D**	**Imb**	**Ngram**	**PPM**	**BERT**	**pALM**
CCAT50	5000	50	-	506	0	76.68	69.36	65.72	63.36
CMCC	756	21	$\times_t$ $\times_g$	601	0	86.51	62.30	60.32	54.76
Guardian10	444	13	$\times_t$ $\times_g$	1052	6.7	100	86.28	84.23	66.67
ROST	400	10	$\times_t$ $\times_g$	3355	10.45	80.94	–	–	–
IMDb62	62,000	62	-	349	2.6	98.81	95.90	98.80	-
Blogs50	66,000	50	-	122	553	72.28	72.16	74.95	-
PAN20	443,000	278,000	$\times_t$	3922	2.3	43.52	-	23.83	-
Gutenburg	29,000	4500	-	66,350	10.5	57.69	-	59.11	-

For the work presented in [14], we did not investigate how BERT would perform on ROST. Therefore, we will conduct this investigation here.

3. Prerequisite

Natural language processing (NLP) models, including those discussed in Section 2.1, as well as architectures such as LSTMs and CNNs, require input data to be represented as numerical vectors. This typically involves converting linguistic features, such as vocabulary terms and parts of speech, into numerical encodings. In contrast to traditional methods that depend on fixed word frequency counts or unique indices, BERT generates contextually informed word embeddings [27], whereby the numerical representation of a word dynamically varies according to its surrounding context and semantic meaning.

BERT is a pretrained language model that requires input data to conform to a specific format [27]. During pretraining, the model constructs a vocabulary comprising individual characters, subwords, and complete words that optimally represent the training corpus. The tokenizer initially attempts to match entire words within this vocabulary; if unsuccessful, it decomposes the word into the largest possible subword units present in the vocabulary. As a final fallback, the tokenizer segments the word into individual characters [28]. This tokenization strategy often increases the length of the original input sequence.

BERT processes input sequences with a maximum length of 512 tokens, of which the special start token ([CLS]) and end token ([SEP]) are predefined and mandatory components [27].

To accommodate these constraints, it was necessary to adjust the length of our texts to ensure compatibility with the model's input requirements. Through preliminary experiments testing sequence lengths of 91, 200, and 400 words, we observed optimal performance with inputs truncated to 200 words. Notably, 91 corresponds to the length of the shortest text in our dataset, which spans from 91 to 39195 words in length. This approach balances the trade-off between preserving sufficient contextual information and adhering to BERT's architectural limitations on input size.

We have set the maximum length of the input vector to 256 to accommodate the situations where some words are not in the pretrained model vocabulary and therefore, may be divided into subwords or even individual characters. Considering that some of the texts were written a couple of centuries ago, that might happen especially for words that are no longer used.

We kept the initial shuffles as described in Table 1 with the following changes:

- for each text that was longer than 200 words, we divided it into multiple texts of 200 words maximum;

TABLE 4. List of authors; the number of texts and their distribution on the training and test sets prepared for BERT

#	**Author**	No. of texts	TrainSet size shuffle 1, 2, 3	TestSet size shuffle 1, 2, 3
0	Ion **Creangă**	**520**	446,399,393	74,121,127
1	Barbu Şt. **Delavrancea**	**922**	706,595,690	216,327,232
2	Mihai **Eminescu**	**792**	550,467,447	242,325,345
3	Nicolae **Filimon**	**475**	389,326,278	86,149,197
4	Emil **Gârleanu**	**203**	169,146,160	34,57,43
5	Petre **Ispirescu**	**674**	489,469,482	185,205,192
6	Mihai **Oltean**	**106**	81,74,86	25,32,20
7	Emilia **Plugaru**	**463**	386,329,312	77,134,151
8	Liviu **Rebreanu**	**711**	535,523,546	176,188,165
9	Ioan **Slavici**	**1966**	1660,1467,1445	306,499,521
	TOTAL	**6832**	**5411,4795,4839**	**1421,2037,1993**

- we have only training and test sets, the validation sets being added to the training sets.

Thus, we obtained 3 shuffles of the dataset with the number of texts as detailed in Table 4.

4. TESTS

For our tests, we used Google Colab Pro[29], an online framework that offers different runtime environments to run code written in Python.

4.1. **Fine-tuning a pretrained Romanian BERT model.** We used the Romanian pretrained BERT model, described in [30] and available at [31], called: `dumitrescustefan/bert-base-romanian-cased-v1`.

For training and testing, we used the following classes from HuggingFace's Transformers [32] library:

AutoTokenizer: - to download the tokenizer associated to the pretrained Romanian model we chose;

AutoModelForSequenceClassification: - to download the model itself;

Trainer: and **TrainingArguments** - to fine-tune the chosen model for our dataset, namely ROST.

The preprocessing of the texts done by the tokenizer consists of splitting the input text into words (or parts of words, punctuation symbols, etc.), usually called tokens, which are then converted to IDs/numbers. To process all the

TABLE 5. Configuration parameters

Parameter	Value
architectures	BertForSequenceClassification
hidden_act	gelu
hidden_dropout_prob	0.1
hidden_size	768
initializer_range	0.02
intermediate_size	3072
layer_norm_eps	1e-12
num_attention_heads	12
num_hidden_layers	12
torch_dtype	float32
transformers_version	4.26.0
vocab_size	50000

TABLE 6. BERT on ROST parameters

Parameter	Value
Evaluation strategy	epoch
Num Epochs	3
Instantaneous batch size per device	8
Total train batch size (parallel, distributed, and accumulation)	8
Test Batch size	8
Gradient Accumulation steps	1
Learning rate	5e-05
Seed	42
Total optimization steps	2031, 1800, 1815
Number of trainable parameters	124449034
Tokenizer maximum length	256

input texts as a batch, one needs to pad them all to the same length or truncate them to the maximum length by specifying that to the tokenizer.

The configuration for our tests is detailed in Table 5.

For our training, the parameters used are presented in Table 6.

There are three different values for *Total optimization steps* corresponding to the three shuffles.

Some training performance metrics we obtained are presented in Table 7.

4.2. **Evaluation.** For evaluating the results, we used HuggingFace's Evaluate [32]. With this library, we computed the *micro-accuracy* metric (aka. *accuracy* or *micro-averaged accuracy*). The corresponding values for this metric,

TABLE 7. BERT on ROST training performance metrics

Parameter	**for shuffle 1**	**for shuffle 2**	**for shuffle 3**
train_runtime (in seconds)	261.1692	239.6519	241.3783
train_samples_per_second	62.155	60.025	60.142
train_steps_per_second	7.777	7.511	7.519
train_loss	0.255753	0.297084	0.381490

TABLE 8. BERT on ROST evaluation performance metrics

Parameter	**for shuffle 1**	**for shuffle 2**	**for shuffle 3**
eval_accuracy	0.864883	0.864997	0.849974
eval_runtime	6.0036	8.474	8.299
eval_samples_per_second	236.69	240.382	240.149
eval_steps_per_second	29.649	30.092	30.124

obtained for the three shuffles, are shown in Table 8 alongside other evaluation performance metrics.

We also used the sklearn.metrics module from the Scikit-learn [33] library. The classes used, pertaining to this module, were:

confusion_matrix: - to compute the confusion matrix;
ConfusionMatrixDisplay: - to generate a Confusion Matrix visualization as the ones displayed in Figures 2;
classification_report: - to obtain a report with main classification metrics;
accuracy_score: - to compute the accuracy classification score, also known as micro-accuracy or overall accuracy;
balanced_accuracy_score: - to compute the balanced accuracy, also known as macro-accuracy.

To compute the confusion matrix, we provided the true and predicted classes (i.e., codes corresponding to the authors). The graphical visualizations of the confusion matrices corresponding to the results obtained for the 3 shuffles are depicted in Figure 2. The numbers from 0 to 9 are the codes given to our authors, as specified in the first column of Table 4.

Figure 2 shows a significant confusion reoccurring between Barbu Ștefănescu Delavrancea (1) and Liviu Rebreanu (8). The texts written by these authors occurred in almost the same period (i.e., 1884-1909 and 1908-1935, respectively).

For each class/author, the classification report provided various metrics:

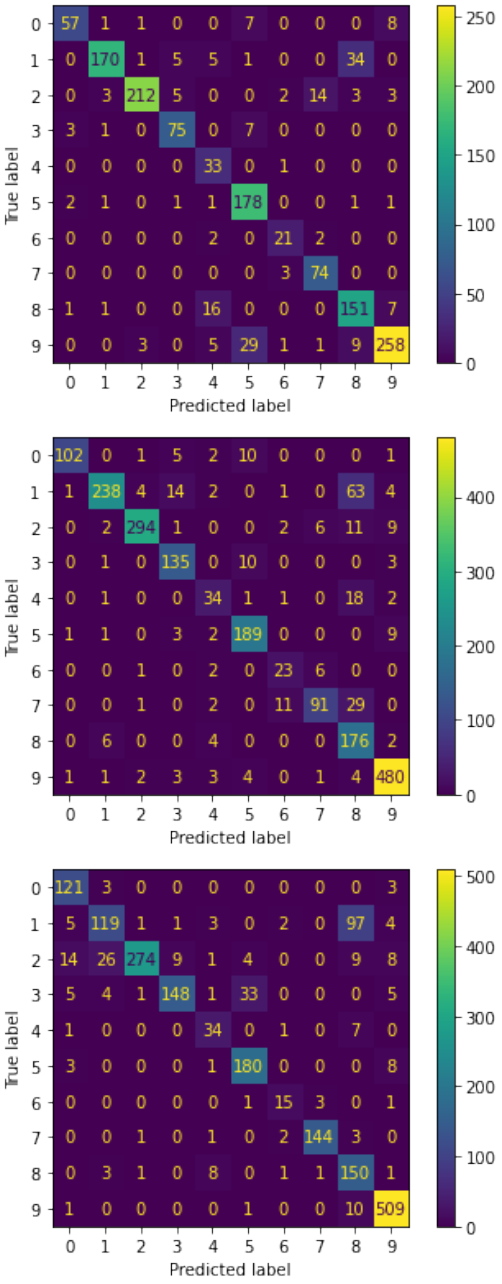

FIGURE 2. Confusion matrices for (a) shuffle 1, (b) shuffle 2, (c) shuffle 3

TABLE 9. Shuffle 1 results generated by sklearn.metrics's classification_report

Name	class	**precision**	**recall**	**f1-score**	support
Ion Creangă	0	0.905	0.770	0.832	74
Barbu Șt. Delavrancea	1	0.960	0.787	0.865	216
Mihai Eminescu	2	0.977	0.876	0.924	242
Nicolae Filimon	3	0.872	0.872	0.872	86
Emil Gârleanu	4	0.532	0.971	0.688	34
Petre Ispirescu	5	0.802	0.962	0.875	185
Mihai Oltean	6	0.750	0.840	0.792	25
Emilia Plugaru	7	0.813	0.961	0.881	77
Liviu Rebreanu	8	0.763	0.858	0.807	176
Ioan Slavici	9	0.931	0.843	0.885	306
	accuracy			0.865	1421
	macro avg	0.831	**0.874**	0.842	1421
	weighted avg	0.882	0.865	0.868	1421

: *Precision*—the number of correctly attributed authors divided by the number of instances when the algorithm identified the attribution as correct;
: *Recall (Sensitivity)*—the number of correctly attributed authors divided by the number of test texts belonging to that author;
: *F1-score*—a weighted harmonic mean of the *Precision* and *Recall*.

The classification results for the 3 shuffles are shown in Tables 9, 10, and 11.

A graphical representation of the results obtained per class and detailed in Tables 9, 10, and 11 is presented in Figure 3.

As can be seen, the worst results over the three shuffles are obtained consistently by Emil Gârleanu (4), Mihai Oltean (6), and Liviu Rebreanu (8), while the best results are obtained by Mihai Eminescu (2), Emilia Plugaru (7), and Petre Ispirescu (5).

Table 12 presents the best micro- and macro-accuracies obtained on the three shuffles.

The best overall accuracy is 0.864997, and it is obtained on the second shuffle. That is interesting, as in the investigations performed in [14] by using other AI techniques, also the second shuffle obtained the best results. This metric is also referred to as *micro-accuracy*, treating each sample (or, in this case, each text fragment) as equally important.

A more representative metric is the *macro-accuracy* as it computes the accuracy by treating each author equally. The best macro-accuracy obtained is 0.874031, and it is obtained on the first shuffle.

TABLE 10. Shuffle 2 results generated by sklearn.metrics's classification_report

Name	class	**precision**	**recall**	**f1-score**	support
Ion Creangă	0	0.971	0.843	0.903	121
Barbu Șt. Delavrancea	1	0.952	0.728	0.825	327
Mihai Eminescu	2	0.970	0.905	0.936	325
Nicolae Filimon	3	0.839	0.906	0.871	149
Emil Gârleanu	4	0.667	0.596	0.630	57
Petre Ispirescu	5	0.883	0.922	0.902	205
Mihai Oltean	6	0.605	0.719	0.657	32
Emilia Plugaru	7	0.875	0.679	0.765	134
Liviu Rebreanu	8	0.585	0.936	0.720	188
Ioan Slavici	9	0.941	0.962	0.951	499
	accuracy			0.865	2037
	macro avg	0.829	**0.820**	0.816	2037
	weighted avg	0.886	0.865	0.868	2037

TABLE 11. Shuffle 3 results generated by sklearn.metrics's classification_report

Name	class	**precision**	**recall**	**f1-score**	support
Ion Creangă	0	0.807	0.953	0.874	127
Barbu Șt. Delavrancea	1	0.768	0.513	0.615	232
Mihai Eminescu	2	0.986	0.794	0.880	345
Nicolae Filimon	3	0.937	0.751	0.834	197
Emil Gârleanu	4	0.694	0.791	0.739	43
Petre Ispirescu	5	0.822	0.938	0.876	192
Mihai Oltean	6	0.714	0.750	0.732	20
Emilia Plugaru	7	0.973	0.954	0.963	151
Liviu Rebreanu	8	0.543	0.909	0.680	165
Ioan Slavici	9	0.944	0.977	0.960	521
	accuracy			0.850	1993
	macro avg	0.819	**0.833**	0.815	1993
	weighted avg	0.871	0.850	0.850	1993

5. Conclusion and further work

In this paper, we continued our investigation started in [14], addressing the authorship attribution problem on a Romanian dataset using BERT. In order to accommodate BERT's input size limitations, we segmented texts into blocks of 200 tokens/words. This approach led to a more balanced dataset than in [14], as the texts have approximately the same length. However, other

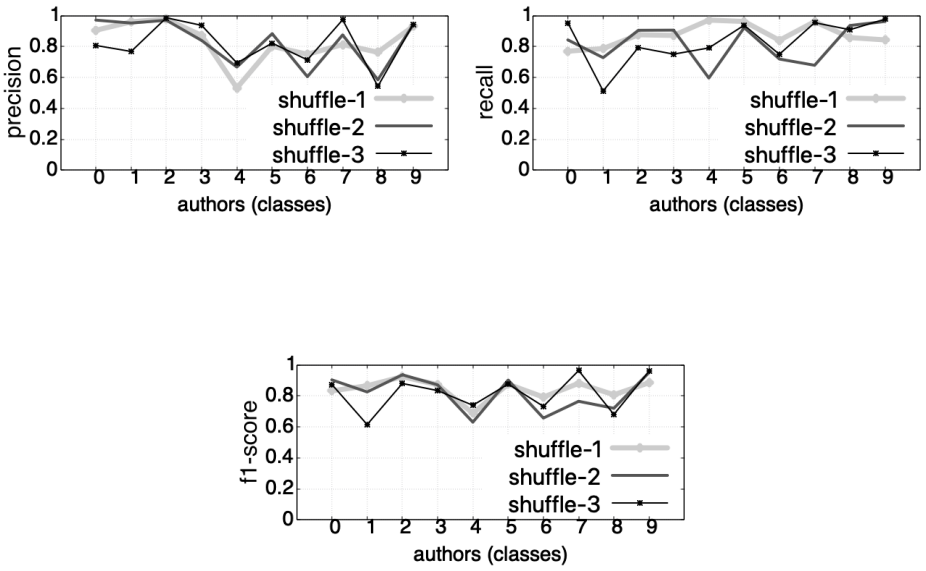

FIGURE 3. Graphical representations of results from Tables 9-11 for (a) precision, (b) recall, (c) f1-score

TABLE 12. Micro- and macro-accuracies

accuracy	**for shuffle 1**	**for shuffle 2**	**for shuffle 3**
micro-accuracy	0.864883	0.864997	0.849974
macro-accuracy	0.874031	0.819585	0.832905

sources of variability—such as the number of texts per author, the provenance of the texts, the historical periods during which the authors wrote, the intended reading medium (paper or online), and the range of genres (stories, short stories, fairy tales, novels, literary articles, and sketches)—remained.

The best result obtained in [14] was 80.94%. The best results achieved here with BERT reached 85.90%. However, due to changes in dataset segmentation made to meet BERT's processing requirements, results are not directly comparable across approaches. For a fair comparison, previous methods (including Artificial Neural Networks, Support Vector Machines, Multi-Expression Programming, Decision Trees with C5.0, and k-Nearest Neighbour) should be re-evaluated on the segmented (200-token) texts.

Our segmentation approach—dividing texts into fixed-size blocks—may disrupt sentence integrity and context. This could potentially impact BERT's performance. In future work, we plan to explore alternative segmentation strategies, such as overlapping sliding windows or segmenting at sentence boundaries, to preserve semantic coherence.

Additionally, we plan to further investigate methods such as Prediction by Partial Matching (PPM) as mentioned in Section 2. Future directions also include: (i) investigating BERT's attention maps to better understand model decision-making for Romanian authorship attribution, (ii) utilizing data augmentation techniques to address dataset imbalance, and (iii) combining BERT embeddings with other stylometric features, such as IPoS, to improve attribution accuracy.

References

[1] Wilson Alves de Oliveira Jr, Edson Justino, and Luiz Sá de Oliveira. Comparing compression models for authorship attribution. *Forensic science international*, 228(1-3):100–104, 2013.

[2] Mike Kestemont, Enrique Manjavacas, Ilia Markov, Janek Bevendorff, Matti Wiegmann, Efstathios Stamatatos, Benno Stein, and Martin Potthast. Overview of the cross-domain authorship verification task at pan 2021. In *CLEF (Working Notes)*, 2021.

[3] Mike Kestemont, Michael Tschuggnall, Efstathios Stamatatos, Walter Daelemans, Günther Specht, Benno Stein, and Martin Potthast. Overview of the author identification task at pan-2018: cross-domain authorship attribution and style change detection. In *Working Notes Papers of the CLEF 2018 Evaluation Labs. Avignon, France, September 10-14, 2018/Cappellato, Linda [edit.]; et al.*, pages 1–25, 2018.

[4] Jacob Tyo, Bhuwan Dhingra, and Zachary C Lipton. On the state of the art in authorship attribution and authorship verification. *arXiv preprint arXiv:2209.06869*, 2022.

[5] Georgios Barlas and Efstathios Stamatatos. A transfer learning approach to cross-domain authorship attribution. *Evolving Systems*, 12(3):625–643, 2021.

[6] Malik Altakrori, Jackie Chi Kit Cheung, and Benjamin C. M. Fung. The topic confusion task: A novel evaluation scenario for authorship attribution. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4242–4256, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[7] Benjamin Murauer and Günther Specht. Developing a benchmark for reducing data bias in authorship attribution. In *Proceedings of the 2nd Workshop on Evaluation and Comparison of NLP Systems*, pages 179–188, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[8] Sebastian Bischoff, Niklas Deckers, Marcel Schliebs, Ben Thies, Matthias Hagen, Efstathios Stamatatos, Benno Stein, and Martin Potthast. The importance of suppressing domain style in authorship analysis. *arXiv preprint arXiv:2005.14714*, 2020.

[9] Efstathios Stamatatos. Masking topic-related information to enhance authorship attribution. *Journal of the Association for Information Science and Technology*, 69(3):461–473, 2018.

[10] Tempestt Neal, Kalaivani Sundararajan, Aneez Fatima, Yiming Yan, Yingfei Xiang, and Damon Woodard. Surveying stylometry techniques and applications. *ACM Computing Surveys (CSuR)*, 50(6):1–36, 2017.

[11] Oren Halvani and Lukas Graner. Cross-domain authorship attribution based on compression. *Working Notes of CLEF*, 2018.

[12] Maël Fabien, Esau Villatoro-Tello, Petr Motlicek, and Shantipriya Parida. BertAA : BERT fine-tuning for authorship attribution. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 127–137, Indian Institute of Technology Patna, Patna, India, December 2020. NLP Association of India (NLPAI).

[13] Georgios Barlas and Efstathios Stamatatos. Cross-domain authorship attribution using pre-trained language models. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 255–266. Springer, 2020.

[14] Sanda-Maria Avram and Mihai Oltean. A comparison of several ai techniques for authorship attribution on romanian texts. *Mathematics*, 10(23):1–35, 2022.

[15] Sanda-Maria Avram. Rost (romanian stories and other texts), 2022.

[16] Sanda Avram. Computing macro-accuracy of mep results on rost, 2023. Software available at `https://github.com/sanda-avram/ROST-source-code/blob/main/ROST_withMEPX.ipynb`.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[18] Douglas Bagnall. Author identification using multi-headed recurrent neural networks. *arXiv preprint arXiv:1506.04891*, 2015.

[19] Jianfeng Zhang, Yan Zhu, Xiaoping Zhang, Ming Ye, and Jinzhong Yang. Developing a long short-term memory (lstm) based model for predicting water table depth in agricultural areas. *Journal of hydrology*, 561:918–929, 2018.

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

[21] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[22] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[23] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[25] Andrei Manolache, Florin Brad, Elena Burceanu, Antonio Barbalau, Radu Ionescu, and Marius Popescu. Transferring bert-like transformers' knowledge for authorship verification. *arXiv preprint arXiv:2112.05125*, 2021.

[26] Jacob Tyo. Computing the macro-accuracy, January 2023. personal communication.

[27] Chris McCormick. Bert word embeddings tutorial, 2019. Available at `http://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/`.

[28] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[29] Ekaba Bisong and Ekaba Bisong. Google colaboratory. *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, pages 59–64, 2019.

[30] Stefan Dumitrescu, Andrei-Marius Avram, and Sampo Pyysalo. The birth of Romanian BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4324–4328, Online, November 2020. Association for Computational Linguistics.

[31] Stefan Dumitrescu, Andrei-Marius Avram, and Sampo Pyysalo. bert-base-romanian-cased-v1, 2023. Available at `https://huggingface.co/dumitrescustefan/bert-base-romanian-cased-v1`.

[32] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania

Email address: `sanda.avram@ubbcluj.ro`