Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 397–406

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2021)

AutoAt: A deep autoencoder-based classification model for supervised authorship attribution

Anamaria Briciu*, Gabriela Czibula, Mihaiela Lupea

Department of Computer Science, Babeş-Bolyai University
1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania

Abstract

Authorship attribution is the task of determining the likely author of a given text, with applications in domains such as literature and literary history, social network analysis, software engineering and cybersecurity. *AutoAt*, a deep autoencoder-based classification model which exploits the ability of autoencoders to encode meaningful data patterns is proposed to solve this task. Experiments are conducted on a data set of 1571 poems authored by 8 Romanian poets using a distributed document representation. The proposed approach obtains comparable or better results with respect to other machine learning classifiers. Additionally, the formulation of the *AutoAt* model allows for the computation of the probability that a test instance belongs to a given author class, which may be a useful property in a variety of authorship attribution applications. This aspect and the fact that *AutoAt* performs well in the difficult task of authorship attribution on poetic data without the step of feature engineering being informed by domain knowledge show that the proposed classifier is a general one, with potential to be used successfully in other fields.

Keywords: ; deep autoencoders; authorship attribution; natural language processing; doc2vec
2000 MSC: 62H30; 68T07

1. Introduction

Authorship analysis is a domain of Natural Language Processing (NLP) which includes text data mining and classification with the aim of extracting information from a text about the writing style, author's sociolinguistic characteristics and, finally, identifying the text's author. There are several related tasks in this domain: (1) *Authorship Attribution* (AA) - identifying the author(s) of a set of texts, (2) *Author Profiling* (AP) - finding author's demographics, such as age, gender, occupation and educational level, (3) *Authorship Verification* (AV) - checking whether a text was written or not written by an author, *Plagiarism Detection* (PD) - searching for paragraphs reproduced from the texts of other

* Corresponding author. Tel.: +4-264-405-327; fax: +4-264-591-906.
E-mail address: anamaria.briciu@ubbcluj.ro

1877-0509
10.1016/j.procs.2021.08.041

authors. For instance, *software plagiarism* is a major problem in educational and corporate environments which may be tackled through authorship attribution [6] by identifying textual similarities between pieces of source code.

Uncovering the hidden characteristics of authors from a corpus of textual data has many societal applications with specific aims in various domains [9], [30]: *literature and history* (to determine the paternity of disputed or anonymous literary and historical documents; to detect pastiche; to compare the styles of different authors); *education* (to understand the personality of students; to detect plagiarism in academic work [17]); *social network analysis* (to extract users' profiles such as identity, sociolinguistic characteristics and their opinions [24]); *cybercrime investigation* (to identify malicious activities such as spamming, ransom messages, harassment, money laundering, illegal material distribution in e-mails, social blogs, SMS-text messages [29]; to provide evidence in courts of law); *software engineering and cybersecurity* (to identify the author of a given piece of source code [19]; to detect software plagiarism [6] and malicious code; to prevent cyber attacks).

An overview of the recent approaches to authorship analysis is provided in [30], summarizing the *authorship detection fields*, the *features* of authorship analysis, the proposed *techniques* and the *corpora* used.

With roots in *stylometry*, early research in the Authorship Attribution task focused on extracting the *explicit features* of texts. Based on the statistical analysis of an author's work, their writing style is identified. For more than fifty years, AA systems have used *stylistic features* for texts' representation, namely varying combinations of lexical features (character and word-based features), syntactic features, structural features and content-specific features. With the advent of deep learning methods, fixed-length, continuous and dense vectors called *document embedding* representations (that contain *hidden/implicit features* of texts) learned from a corpus of documents are used successfully to represent texts in different classification tasks, including authorship analysis tasks.

As far as the *classification* step is concerned, approaches have been diverse, employing both one-class and multi-class perspectives, using unsupervised methods, supervised methods and, more recently, deep learning classification models.

Within the *unsupervised machine learning* field, *autoencoders* (AEs) are frequently used for non-linear dimensionality reduction and thus for representing, through their lower-dimensional latent space representation, relevant data features. In this paper we are exploiting the ability of AEs to encode meaningful data patterns relevant for distinguishing the author of a text. A supervised classifier *AutoAt*, based on an ensemble of deep AEs, is introduced for authorship attribution. The proposed classifier is general and it can be applied for AA in various domains. As an application domain we selected the authorship attribution task in Romanian poetry and used document embedding representations [21] of poems.

The contribution of the proposed approach is threefold, as we contribute in all three processes that define an authorship attribution system: (1) the distributed representation of poetic texts for encoding the writing style, the syntactic and semantic aspects; (2) the architecture consisting of an ensemble of deep AEs; and (3) the evaluation of the system on a data set containing 1571 poems authored by 8 Romanian poets. To our knowledge there are no research works in authorship attribution for poetry with document embedding representation of poems and a multi-class classification model based on autoencoders.

To sum up, the paper aims at answering the below mentioned research questions:

RQ1 How to introduce a multi-class classification model based on an ensemble of deep autoencoders to supervisedly identify the author of a given text, based on the encoded structural and conceptual relationships between the documents written by the same author?

RQ2 What is the performance of the approach introduced for answering RQ1 for identifying the authors of Romanian poetry and how does it compare to the performance of similar classification models?

RQ3 What is the relevance of the document embedding representation of the poetic texts in discriminating among different authors?

The remainder of the paper is organized as follows. Section 2 presents work in the authorship attribution field related to our proposal and introduces the fundamentals of autoencoders. In the next section, the *AutoAt* classification model for authorship attribution is proposed, and all its components are described. The experimental part (data set, results and discussions) is presented in Section 4. The last section contains conclusions and directions for future work.

2. Background

This section starts by reviewing existing related work in authorship attribution for poetry (in Section 2.1). Then, Section 2.2 presents the fundamentals of autoencoders, the main deep learning model used in the proposed approach.

2.1. Related work

In this section approaches related to the proposed AA system in at least one of the following aspects: (1) the representation of literary texts using document embeddings, (2) deep autoencoders used as classification models (3) the text type - poetry and (4) the language - Romanian, are presented.

The authorship attribution task for **poetry** has not been addressed so much compared to prose. Guzman-Cabrera [15] proposes a categorization method based on the use of n-grams of characters and the extraction of unlabeled examples from the web to enrich the training set. For evaluation, a corpus containing poems of five contemporary Mexican poets was used. Machine learning algorithms such as Naïve Bayes and SVM were employed in authorship attribution tasks for English poetry [12] and Arabic poetry [1]. The experiment performed by Gallagher and Li [12] based on 406 same theme English poems of five poets from World War I proved that for this task, the bag-of-words representation of texts provides better results than the stylometric representation. The features used by Ahmed et al. [1] were characters, sentence length, word length, meter, rhyme and first word in sentence. The corpus for the experiment was composed of poems authored by seventy-three poets from different eras.

In many NLP tasks (text classification, sentiment analysis, automated translation) the **distributed document representation** learned by the `doc2vec` model [21] proved to outperform other techniques for texts' representation. This type of representation has rarely been applied to the authorship attribution task, but not for poetry. The authors of [24] used `doc2vec` representation and a logistic regression classifier to solve the author profiling (AP) task, working on texts in social media, such as Twitter, blogs, and reviews. Document embeddings learned on various types of n-grams (characters, words, POS-tags) have been successfully applied for cross-topic authorship attribution [16].

The Author Verification task in the domain of cybercrime, with the aim of verifying the authors behind Internet Relay Chat (IRC) messages was approached in [29]. A one-class classifier based on a deep **autoencoder** was successfully employed to identify threat levels of messages. An autoencoder working as a one-class classification model was proposed in an automated document retrieval task to filter the documents "of interest", the experiments being performed on the standard Reuters database [23].

For **Romanian literature**, the authorship identification task was approached by Dinu et al. [11]. A hierarchical clustering algorithm, based on a stylistic similarity between texts, was applied in order to distinguish between the styles of two novelists, Mateiu Caragiale and Radu Albala. Frequency rankings of function words, considered style markers, were proposed as features for texts, and Rank distance was used to compare texts. This idea was extended in [10] to solve pastiche detection, with the aim of exposing different authors who attempted to impersonate or to mimic Mateiu Caragiale's style.

In the context presented above, the novelty of our approach consists in solving the authorship attribution task in Romanian poetry based on a deep classification model and using the document embedding representation of poems.

2.2. Autoencoders

AEs are well-known deep learning models used in multiple areas such as medical data analysis [25], image analysis [20], bioinformatics [7], speech processing [8], and so on. An AE is composed of two neural networks which are trained to approximate the identity function (i.e. to reconstruct the input), thus being known as *self-supervised learning* techniques. The two neural network components are stacked one on top of the other: the first network is an *encoder* which transforms the input data into a *hidden* representation (also known as *intermediate* or *latent* representation). while the second network is a *decoder* aimed to reconstruct the input value from the latent representation. Thus, AEs learn to encode the structure of the data they are trained on very well, being able to reconstruct only the data sampled from the same distribution as the training data, but without succeeding to recreate data dissimilar to the training samples.

3. Methodology

With the goal of answering the research question RQ1, the *AutoAt* classification model for authorship attribution is proposed. *AutoAt* consists of an ensemble of AEs (one AE for each author) used with the goal of encoding the underlying characteristics (both structural and conceptual) of the documents belonging to the same author. The experiments will highlight that the AE's lower-dimensional latent space is able to encode relevant characteristics of the documents belonging to a certain author and thus the ensemble of AEs will be able to distinguish between documents belonging to different authors. After each of the AEs is trained on the documents of a certain author, a new document d will be assigned to the author corresponding to the autoencoder A if d is highly similar to the information encoded by A and dissimilar to the information encoded by the other autoencoders.

The problem of authorship attribution may be formalized as a multi-class classification problem. Assuming that a set of authors $\mathcal{A} = \{A_1, A_2, \dots A_n\}$ and a set of documents (texts) $\mathcal{T} = \{T_1, T_2, \dots T_r\}$ are given, the goal is to predict the class (author) $A_i, 1 \leq i \leq n$ to which a certain text $t \in \mathcal{T}$ is likely to belong.

Thus, from a machine learning perspective, we aim at approximating a target function $f : \mathcal{T} \rightarrow \mathcal{A}$ that maps documents from $\mathcal{T}$ to a certain class/author $a \in \mathcal{A}$.

The *AutoAt* classifier consists of n autoencoders $AE_1, AE_2, \dots, AE_n$, the autoencoder AE_i corresponding to the author A_i ($\forall 1 \leq i \leq n$). AE_i will be self-supervisedly trained on the documents (texts) from $\mathcal{T}$ authored by the author A_i. Thus, through its latent state representation, AE_i will unsupervisedly learn features relevant for author A_i which will be useful in discriminating between the different authors. The present approach consists of four stages illustrated in Figure 1: *data preprocessing and representation*, *training* and *evaluation*. The stages of *AutoAt* will be further detailed.

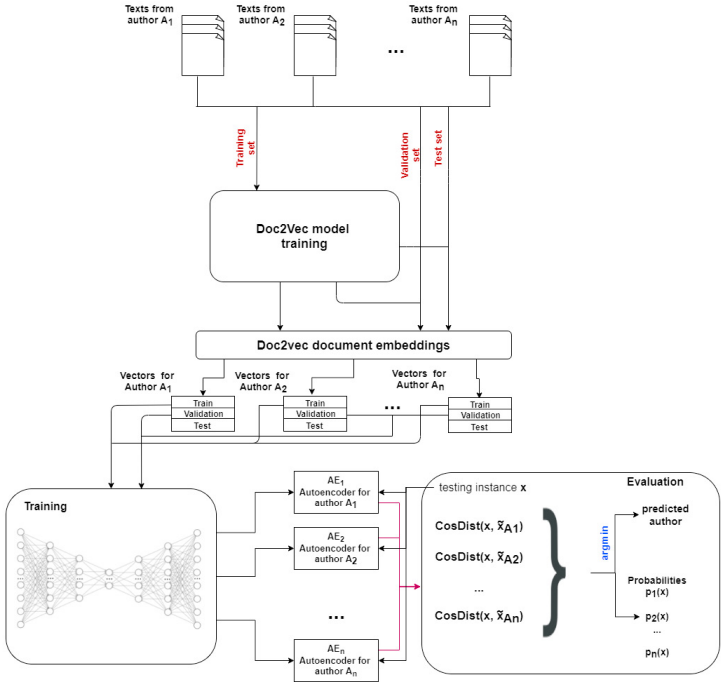

Fig. 1: Overview of *AutoAt*.

3.1. Data preprocessing and representation

This stage refers to the preparation of the texts to be used in training the document embedding models, from which numerical vectors will be generated as distributed representations (denoted by d) of the documents $t \in \mathcal{T}$.

Text preprocessing is the first step in the process of building a model suited to a natural language processing task, and often includes tokenization, stop words removal, lemmatization and stemming. However, many authorship analysis studies have highlighted the importance of punctuation and function words in author identification [18]. Therefore, the only preprocessing employed on the data set beyond tokenization is lemmatization of word tokens, which was found to generate slightly better results than the use of completely unprocessed word tokens. Both of these transformation steps were done using the NLP-Cube library [3].

The `doc2vec` (*Paragraph vector*) [21] model learns how to project a document from a corpus of documents into a latent m-dimensional space. This model is a neural network with three layers: an input layer (corresponds to the context: a window of surrounding words and the documents as special words), one hidden layer (m neurons, corresponding to the number of features) and an output layer (corresponds to the predicted target word). After the training process, m-dimensional vector representations of the words (*word embeddings*) and of the documents (*document embeddings*) are obtained. The distributed vector representation of a document encodes different aspects of the text: content, syntax, semantics and other hidden features. Thus, the documents can be compared in the m-dimensional space to find similarities based on the learned features.

In the proposed approach no explicit features, such as stylistic features, are used to represent the poems. The `doc2vec` model is applied in order to learn implicit (hidden) features from the poets' works and generate the document embeddings of poetic texts. In the experiments, we want to verify whether the extracted features characterize the author's writing style, the literary movement he belongs to and whether they succeed in distinguishing the authors.

Therefore, for each author A_i ($1 \leq i \leq n$), a data set D_i will be built using all available documents/texts authored by A_i and represented as m-dimensional vectors, as shown above. The library used for the `doc2vec` model is gensim [28].

3.2. Training

As illustrated in Figure 1, an ensemble of n autoencoders will be built by training a distinct autoencoder AE_i for each author A_i. For training each A_i ($\forall 1 \leq i \leq n$) we will use 70% of D_i (i.e. 70% documents authored by A_i), 20% of D_i for validating the model and the remaining of 10% from each D_i will be afterwards used for testing.

In the current study sparse denoising AEs are used to learn lower-dimensional representations for the input documents (represented as m-dimensional numerical vectors, as described in Section 3.1). The latent representations learned by the autoencoders will encode information which will be relevant for characterizing the documents of specific authors. The main goal is to learn meaningful hidden representations that are specific to the authors. The loss function used for measuring how close the output of the AE is to its input, is being computed as $L(\tilde{x}, x) = \frac{1}{m} \sum_{j=1}^{m} (\tilde{x}_j - x_j)^2$, where x represents the m-dimensional input and $\tilde{x}$ represents the model's m-dimensional output.

Various autoencoder architectures were tested in the experimental step. The architecture for which the best results were obtained is based on the idea of an initial sudden reduction of the input dimensions, and then a progressive reduction to a 2-dimensional encoding followed by symmetric reconstruction. For input vectors of size 100, the architecture consists of an input layer with m neurons (i.e. 100) followed by seven hidden layers (three hidden layers for encoding, with 16, 8, and 4 neurons, 2 neurons on the encoding layer and three symmetric hidden layers for decoding). For input sizes 150 and 300, the best performing architecture consists of an input layer with a number of neurons equal to the dimensionality of the input data, followed by eleven hidden layers (five hidden layers with 128, 32, 16, 8, and 4 neurons, 2 neurons on the encoding layer and five symmetric hidden layers for decoding).

For both architectures, all the hidden layers use the ReLU activation function [13], except the encoding layer which uses the linear activation. The output size is equal to the input size (i.e. m neurons), with the linear activation function. The network is trained using stochastic gradient descent enhanced with the *Adam* optimizer [13], the minibatch perspective and an early stopping criterion (by monitoring the loss convergence on the validation set).

3.3. Evaluation

After the *AutoAt* classifier was trained (see Section 3.2), it is tested in order to evaluate its performance. For testing 10% from each data set D_i ($\forall 1 \leq i \leq n$) was used, i.e. 10% documents (taken for each author A_i) which were unseen during training.

3.3.1. Classification

During the classification stage, when a new document d has to be classified, *AutoAt* searches for the autoencoder that minimizes the "distance" between d and $\tilde{d}$ (the instance reconstructed by the autoencoder). The "distance" between two texts (documents) represented as m-dimensional numerical vectors d_1 and d_2 is expressed as the cosine distance between them, $CosDist(d_1, d_2) = 1 - cos(d_1, d_2)$, where $cos(d_1, d_2)$ represents the cosine similarity between d_1 and d_2, scaled to [0,1]. The intuition behind this classification decision is the following: if the input document d is the most similar to the reconstruction of d provided by AE_i (i.e. d is closer to its reconstruction), it is very likely that d has a high structural and conceptual similarity to the information encoded by AE_i and thus it is highly probable to be authored by A_i.

For each testing instance d, *AutoAt* determines the probabilities $p_1(d), p_2(d), \ldots, p_n(d)$, where $p_i(d)$ represents the probability that the input instance d belongs to class A_i, where $1 \leq i \leq n$. Let us denote by $CosDist(d, \tilde{d}_{A_i})$ the cosine distance between the instance d and its reconstruction, $\tilde{d}_{A_i}$, through the autoencoder A_i. The probabilities $p_i(d), \forall i \in \{1, 2, \ldots, n\}$ are defined as $p_i(d) = \frac{1 - CosDist(d, \tilde{d}_{A_i})}{n - \sum_{j=1}^{n} CosDist(d, \tilde{d}_{A_j})}$.

It has to be noted that $p_i(d)$ are probabilities, i.e. $0 \leq p_i(d) \leq 1$ and $\sum_{i=1}^{n} p_i(d) = 1, \forall d$. Moreover, the probability $p_i(d)$ increases as $CosDist(d, \tilde{d}_{A_i})$ decreases (i.e. $p_i(d) \geq p_j(d)$ if $CosDist(d, \tilde{d}_{A_i}) \leq CosDist(d, \tilde{d}_{A_j})$). Thus, $\text{argmin}_{j=1,n} CosDist(d, \tilde{d}_{A_j}) = \text{argmax}_{j=1,n} p_j(d)$, meaning that an instance d will be classified by *AutoAt* as belonging to the author (class) A_k such that $k = \text{argmin}_{i=1,n} CosDist(d, \tilde{d}_{A_i})$.

3.3.2. Testing

Due to the randomness in selecting the training/validation/testing subsets, a cross-validation testing methodology was applied for obtaining a more precise evaluation of our model's performance. Thus, the training/validation/testing split is repeated 10 times.

For measuring the performance of *AutoAt* on a certain testing set, the *F-score* (denoted by $F\text{-}score_i$) is determined for the author (class) A_i as the harmonic mean between the *precision* ($Prec_i$) and *recall* ($Recall_i$) values for the i-th class, $F\text{-}score_i = \frac{2 \cdot Prec_i \cdot Recall_i}{Prec_i + Recall_i}$ [14]. Since the data sets D_i used in our experiments are slightly imbalanced, aggregated *Precision*, *Recall* and *F-score* are computed as the weighted average of the $Prec_i$, $Recall_i$ and $F\text{-}score_i$ values obtained for the classes: $Precision = \frac{\sum_{i=1}^{n}(w_i \cdot Prec_i)}{\sum_{i=1}^{n} w_i}$, $Recall = \frac{\sum_{i=1}^{n}(w_i \cdot Recall_i)}{\sum_{i=1}^{n} w_i}$, $F\text{-}score = \frac{\sum_{i=1}^{n}(w_i \cdot F - score_i)}{\sum_{i=1}^{n} w_i}$, where w_i represents the cardinality of D_i.

The aggregated *Precision*, *Recall* and *F-score* values are then averages over the 10 runs during the cross-validation and the 95% *confidence interval* [5] of the mean values is reported.

4. Results and discussion

This section details the experimental part of the paper, with the goal of assessing the performance of *AutoAt* and answering the research questions RQ2 and RQ3. The experiments will be conducted following the methodology introduced in Section 3 for the authorship attribution task in Romanian poetry by using the document embedding representations [21] of poems.

4.1. Data set

For the experimental part, a corpus containing the poetic works of eight Romanian poets was used. They are representative authors of Romanian poetry from the second half of the nineteenth century and the first part of the twentieth century. Their works belong to different literary movements: *classicism*, *realism*, *romanticism*, *modernism*, *symbolism* and *sămănătorism* (a specific Romanian movement characterized by the return to the past and to history, the true cult for the patriarchal village).

The data set contains a total of 1571 instances for eight classes, as shown in Table 1. There are 554 043 tokens in total in the data set. The number of tokens includes words, punctuation characters and other symbols. The *entropy* computed for the data set is 0.967, value which indicates a data set which is fairly balanced.

	Authors							
	Alexandru Macedonski	George Coșbuc	George Topîrceanu	Ion Minulescu	Mihai Eminescu	Octavian Goga	Vasile Alecsandri	Ștefan O. Iosif
ID	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
No. poems	190	212	113	159	366	181	186	164
No. tokens	39 403	124 809	31 525	35 380	182 270	37 761	72 025	30 870

Table 1: Description of data set

Figure 2 depicts a two-dimensional t-SNE [22] projection of the numerical vectors (learned by the `doc2vec` model) representing the documents in an 150-dimensional space. The figure shows a relatively low degree of separation between classes/authors. Some author classes are more easily identified, with the majority of instances grouped together (e.g. I. Minulescu (A_4), V. Alecsandri (A_7), G. Coșbuc (A_2) and O. Goga (A_6)). For the rest of the authors, instances are spread out: M. Eminescu (A_5) has both clearly separated instances and overlaps with the authors Al. Macedonski (A_1), Șt. O. Iosif (A_8) and V. Alecsandri (A_7) in particular. G. Topîrceanu (A_3), Șt. O. Iosif (A_8) and Al. Macedonski (A_1), in turn, seem to be the author classes most difficult to differentiate.

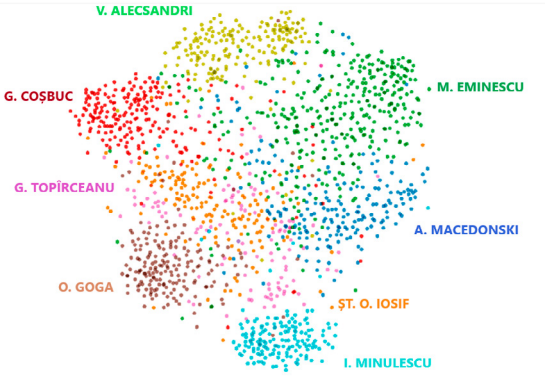

Fig. 2: t-SNE visualization of the data instances.

The difficulty in finding bounds for the A_5 class may be due to the large number of instances, distributed across phases of creation [4] and various topics. Moreover, neighboring authors are linked by literary movements, either as contemporaries (A_2, A_8, A_6) or as successors in a transformative evolution of content and style (A_7 and A_5 or A_5 and A_1).

4.2. Parameters setting

For the AEs' implementations we used the scikit-learn library [26]. Optimization of the autoencoder is achieved via stochastic gradient descent enhanced with the Adam optimizer [13]. The initial learning rate used is 10^{-4}. Additionally, the learning rate is reduced on a plateau of the validation loss, by a factor of 0.1 if there are 20 training epochs without change. The algorithm is run with minibatch updates, using a batch size of 4 and a maximum of 1000 training epochs. Early stopping is employed by monitoring the convergence of the validation loss. Specifically, if the validation loss does not decrease by more than 0.005 in 25 epochs, training is stopped and the best performing model is kept.

4.3. Results

Table 2 presents the experimental results obtained by applying the *AutoAt* classification model on the data set described in Section 4.1, by following the methodology introduced in Section 3. The first column depicts the number

of features used for representing the instances (i.e. dimensionality of the doc2vec representation). The next columns present the values for the performance metrics obtained for each author (for the author A_i these values are denoted by $Prec_i$, $Recall_i$ and $F\text{-}score_i$, defined in Section 3.3.2), averaged over 10 runs of the algorithm. A 95% *confidence interval* (CI) [5] is used for the mean value. The best obtained results are highlighted.

Number of features (m)	Performance measure	Authors								Overall
		A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	
100	*Prec*	0.85	0.83	0.58	0.89	0.92	0.73	0.84	0.67	**0.81** ± 0.017
	Recall	0.72	0.84	0.64	0.95	0.73	0.89	0.90	0.67	**0.79** ± 0.017
	F-score	0.77	0.83	0.67	0.92	0.82	0.81	0.86	0.66	**0.79** ± 0.018
150	*Prec*	0.88	0.85	0.63	0.89	0.93	0.73	0.82	0.68	**0.82** ± 0.019
	Recall	0.74	0.82	0.73	0.95	0.74	0.81	0.86	0.66	**0.8** ± 0.019
	F-score	0.80	0.83	0.68	0.92	0.82	0.81	0.86	0.66	**0.81** ± 0.02
300	*Prec*	0.82	0.83	0.62	0.91	0.98	0.68	0.79	0.73	**0.82** ± 0.015
	Recall	0.75	0.85	0.70	0.95	0.67	0.90	0.94	0.66	**0.8**± 0.014
	F-score	0.78	0.84	0.66	0.93	0.79	0.77	0.85	0.68	**0.79** ± 0.014

Table 2: Experimental results. A 95% CI is used for the overall performance.

The results from Table 2 indicate that the best overall performance of *AutoAt*, in terms of *F-score*, was obtained for 150 features, namely a dimensionality of 150 for the doc2vec representation of the documents. The best *F-score* of 0.81 highlights a good performance of *AutoAt* and shows that the ensemble of autoencoders successfully learned (from the provided doc2vec representation) meaningful, discriminative features for most of the authors. In addition, the small values obtained for the confidence intervals denote the stability of the model. We also observe that the performances for the various number of features are close, which indicates that doc2vec is able to extract meaningful features, independent of the feature set dimensionality.

We note that the best distinguishable author is A_4 (an overall *F-score* of 0.92), followed by A_7 (*F-score* of 0.87) and A_2 (*F-score* of 0.83). However, there are authors (ex. A_3 and A_8) for which poorer performances were obtained. Overall, the results are perfectly correlated with the t-SNE visualization from Figure 2. A possible cause for the poorer performance for certain authors may be the fewer number of training instances (poems), from which doc2vec could not extract representative features and, consequently, the autoencoders could not obtain good enough encodings.

The difficulty measure [2] is used in the literature for measuring the percentage of instances for which the nearest neighbour (excluding the class label when computing the distance) has a different label. The higher the value of the difficulty for a class is, the more difficult it is for a classifier to identify that class. The difficulties depicted in Figure 3, which were computed considering 150 features for representing the instances of an author and the cosine distance between them, support our findings, showing that there is a correlation between the *F-score* and the difficulties computed for each author.

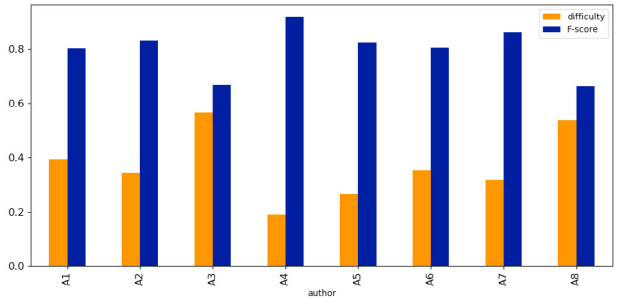

Fig. 3: Correlation between the *F-score* values and the *difficulties* computed for each class/author.

Studying the output classification results in more detail by looking at probabilities, for all 10 runs, we note that for about 50% of the misclassified instances (poems) the second most probable author is the real author. The corresponding probabilities for the predicted and the real authors are very close, indicating similarities between their poems, explained either by membership to the same literary movement or a similar content with respect to topic. A possible

improvement at the decision level is to choose from the two most probable authors (using a threshold between the probabilities) by comparing the reconstructed vectors with the authors' vectors. The poet (class) for which the greatest similarity is obtained between his vector and the vector reconstructed by his AE, will be predicted. Such author (class) vectors can be learned by the `doc2vec` model by having all documents belonging to the same author share a tag/label in training.

Since there are no approaches in the literature for AA in Romanian poetry, we decided to apply six well-known machine learning classifiers on the data set described in Section 3.1, following the testing methodology used for evaluating the performance of *AutoAt* (the performance measures were computed as shown in Section 3.3.2 and the testing was repeated by running the algorithm 10 times, on 10 training-validation-testing splits (once for every split): *Support Vector Classifier* (SVC), *Multilayer Perceptron* (MLP), *Logistic Regression* (LR), *Gaussian Naive Bayes* (GNB), *k Nearest Neighbors* (kNN) and *Decison Trees* (DT). The comparison is depicted in Table 3, where for *AutoAt* the best performance is reported, namely that obtained using 150 features for the `doc2vec` representation. For SVC and all the other algorithms, a grid search was applied for determining the optimal parameters. 95% CIs are used for the values averaged over the ten runs of the classifiers.

Number of features (m)	**Classifier**						
	AutoAt	**SVC**	**MLP**	**LR**	**GNB**	**kNN**	**DT**
150	**0.81** ± 0.02	**0.83** ± 0.012	0.81 ± 0.017	0.78 ± 0.019	0.52 ± 0.027	0.41 ± 0.022	0.3 ± 0.024

Table 3: Comparison between *AutoAt* and classifiers from the literature in terms of *F-score*. 95% confidence intervals are used for the results.

The results from Table 3 highlight that *AutoAt* compares favorably with existing classifiers, it is slightly outperformed (only by 2%) by SVC, has the same performance with MLP and outperforms the other four classifiers. Thus, in about 83% of the cases (5 out of 6), *AutoAt* has a performance higher or equal to the existing classifiers.

The research questions stated at the beginning of our paper can now be answered. We introduced *AutoAt*, a multi-class classification model based on an ensemble of deep autoencoders to supervisedly identify the author of a given text, based on the encoded structural and conceptual relationships between the documents written by the same author. The experiments conducted on a corpus of poetic texts authored by eight Romanian poets showed that the document embeddings containing features extracted by `doc2vec` model are appropriate representations that capture the text's content, syntactic and semantic aspects and other specific characteristics of the author. These distributed representations used as input data by *AutoAt* were transformed into more condensed latent representations which discriminated between the authors very well, allowing the classifier to correctly recognize the author of a given poem.

5. Conclusions and future work

The generality of the *AutoAt* multi-class classifier introduced for authorship attribution has to be noted. Even if we have selected the task of identifying the author of literary texts, the approach is general and applicable in various domains (e.g. plagiarism detection, source code attribution) for deciding the author of a certain document based on the conceptual similarity with other documents written by that author as well as the conceptual dissimilarity with other authors' work. *Deep autoencoders* were used with the goal of uncovering both structural and conceptual characteristics of the texts written by certain authors.

The application of the *AutoAt* deep classifier using document embedding representations of poetic texts proved to successfully solve the authorship attribution task in Romanian poetry. Based on the performed experiments we conclude that the hidden features learned by `doc2vec` model encode the specificity of a poet's work, including the literary movement he belongs to and other implicit characteristics that differentiate him from other authors.

As future work we propose a combination of the implicit (hidden) features extracted by `doc2vec` and stylistic features specific to poetry (features that capture phonic phenomena: euphony, assonance, alliteration, rhyme and words properties: frequency distribution, vocabulary richness [27], [4]) for poems' representation. In addition, we will further investigate the application of *AutoAt* for authorship attribution in other domains, such as source code authorship attribution.

Acknowledgments

This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI – UEFISCDI, project number PN-III-P4-ID-PCE-2020-0800, within PNCDI III.

References

[1] Ahmed, A., Mohamed, R., Mostafa, B., 2017. Machine learning for Authorship Attribution in Arabic poetry. International Journal of Future Computer and Communication 6, 42–46.
[2] Boetticher, G.D., 2007. Advances in Machine Learning Applications in Software Engineering. IGI Global. chapter Improving the Credibility of Machine Learner Models in Software Engineering.
[3] Boroș, T., Dumitrescu, S.D., Burtica, R., 2018. NLP-cube: End-to-end raw text processing with neural networks, in: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Association for Computational Linguistics, Brussels, Belgium. pp. 171–179.
[4] Briciu, A., 2019. Quantitative analysis of style in Mihai Eminescu's poetry. Studia Universitatis Babeș-Bolyai Informatica 64, 80–95.
[5] Brown, L., Cat, T., DasGupta, A., 2001. Interval estimation for a proportion. Statistical Science 16, 101–133.
[6] Burrows, S., Tahaghoghi, S.M.M., 2007. Source code Authorship Attribution using n-grams, in: RMIT UNIVERSITY, pp. 32–39.
[7] Czibula, G., Codre, C., Teletin, M., 2021. Anomalp: An approach for detecting anomalous protein conformations using deep autoencoders. Expert Systems with Applications 166, 114070.
[8] Deng, J., Zhang, Z., Marchi, E., Schuller, B., 2013. Sparse autoencoder-based feature transfer learning for speech emotion recognition, in: ACII, IEEE. pp. 511–516.
[9] Ding, S.H.H., Fung, B.C.M., Iqbal, F., Cheung, W.K., 2019. Learning Stylometric Representations for Authorship Analysis. IEEE Transactions on Cybernetics 49, 107 – 121.
[10] Dinu, L., Niculae, V., Șulea, O., 2012. Pastiche detection based on stopword rankings. Exposing impersonators of a Romanian writer, in: Proceedings of EACL 2012, Workshop on Computational Approaches to Deception Detection, pp. 72–77.
[11] Dinu, L., Popescu, M., Dinu, A., 2008. Authorship Identification of Romanian texts with controversial paternity, in: Proceedings of LREC 2008, pp. 3392–3397.
[12] Gallagher, C., Li, Y., 2019. Text categorization for Authorship Attribution in English Poetry. Intelligent Computing 858, 249–261.
[13] Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.
[14] Gu, Q., Zhu, L., Cai, Z., 2009. Evaluation measures of the classification performance of imbalanced data sets, in: Computational Intelligence and Intelligent Systems, Springer Berlin Heidelberg. pp. 461–471.
[15] Guzman-Cabrera, R., 2020. Author Attribution of Spanish poems using n-grams and the web as corpus. Journal of Intelligent & Fuzzy Systems 39, 2391–2396.
[16] Gómez-Adorno, H., Posadas-Durán, J., Sidorov, G., Pinto, D., 2018. Document embeddings learned on various types of n-grams for cross-topic Authorship Attribution. Computing 100, 741—-756.
[17] Hansen, N., Lioma, C., Larsen, B., Alstrup, S., 2014. Temporal Context for Authorship Attribution. A Study of Danish Secondary Schools. Multidisciplinary Information Retrieval. IRFC 2014. Lecture Notes in Computer Science 8849, 22 – 40.
[18] Juola, P., 2006. Authorship attribution. Information Retrieval 1, 233–334.
[19] Kalgutkar, V., Kaur, R., Gonzalez, H., Stakhanova, N., Matyukhina, A., 2019. Code Authorship Attribution: Methods and challenges. ACM Computing Surveys (CSUR) 52, 1 – 36.
[20] Le, Q., 2013. Building high-level features using large scale unsupervised learning, in: ICASSP, IEEE. pp. 8595–8598.
[21] Le, Q., Mikolov, T., 2014. Distributed representations of sentences and documents, in: Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014, pp. 1188–1196.
[22] van der Maaten, L., Hinton, G., 2008. Visualizing Data using t-SNE. Journal of Machine Learning Research 9, 2579–2605.
[23] Manevitz, L., Yousef, M., 2007. One-class document classification via neural networks. Neurocomputing 70, 1466–1481.
[24] Markov, I., H., G.A., JP., P.D., G., S., A., G., 2017. Author Profiling with Doc2vec neural network-based document embeddings. Advances in Soft Computing. MICAI 2016, Lecture Notes in Computer Science 10062, 117–131.
[25] Niţică, , Czibula, G., Tomescu, V., 2020. A comparative study on using unsupervised learning based data analysis techniques for breast cancer detection, in: 2020 IEEE 14th International Symposium on Applied Computational Intelligence and Informatics (SACI), pp. 000099–000104.
[26] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.
[27] Popescu, I., Lupea, M., Tătar, D., Altmann, G., 2015. Quantitative analysis of poetic texts. De Gruyter Mouton.
[28] Rehurek, R., Sojka, P., 2011. Gensim–python framework for vector space modelling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic 3.
[29] Shao, S., Tunc, C., Al-Shawi, A., Hariri, S., 2019. One-class Classification with Deep Autoencoder Neural Networks for Author Verification in Internet Relay Chat, in: Proceedings of 16th IEEE/ACS International Conference on Computer Systems and Applications, pp. 1–8.
[30] Swain, S., Mishra, G., Sindhu, C., 2017. Recent approaches on Authorship Attribution techniques — An overview, in: 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), pp. 557–566.