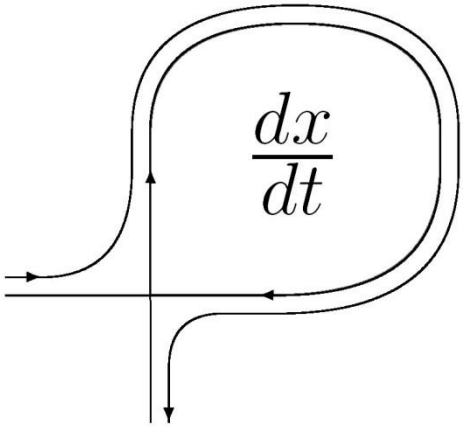

DIFFERENTIAL EQUATIONS
AND
MANAGEMENT PROCESSES
N. 2, 2024
Electronic journal,
reg. El No. FS77-39410 dated
04/15/2010
ISSN 1817-2172

http://diffjournal.spbu.ru/
e-mail:jodiff@mail.ru

AI and machine learning
in control processes

Using Bellman optimality principle for the generative autoencoder architecture for the problems of the attribute data typesetting and semantic description in data management

Kuznetsov S.V. 1,2,*

[1]Unidata LLC
[2]St. Petersburg State University

[*]sergey.kouznetsov@gmail.com

Annotation. The publication presents an original way of solving the problems of identifying data type (typesetting) and semantic description of the attributes when managing structured data and master data. A formal definition of the generalized attribute typesetting problem is given, which allows generation of the additional data types – a key aspect of the applied data management tasks. This problem allows using the discrete Bellman optimality principle under special criteria of the target function. A unified architecture of the deep generative neural network addressing simultaneously the generalized attribute typesetting and semantic description generation problems is proposed. The architecture is based on the adversarial autoencoder architecture (AAE) using the mechanisms of soft-attention, and long-term memory (SCRN). The effectiveness of such implementation, in particular, is achieved through the application of the principles of dynamic programming within each epoch of the network training.

Keywords: generative neural networks, metadata, data typesetting, data types generation, discrete Bellman optimality principle, neural network architecture, deep learning, GAN, data profiling, master data, AAE, SCRN, LSTM, memory cell, soft attention.

1. Introduction

Increased volumes of data and computing power over the past decades have served as a powerful impetus for the development of machine learning, as well as new areas of applications of the classical discrete optimization problems algorithms that appeared back in the middle of the 20th century [1]. Preparation and correct interpretation of data is an important applied task and a key aspect for all machine learning algorithms, since they are both used for initial training and "pre-trained models" [2]. Over the past two decades, a whole class of tools and methodologies managing different types of data has emerged [3], while the use of machine learning and deep neural networks, in particular, has become the subject of research relatively recently [4,5]. Good examples of such investigations are ensembled architecture of the long short-term memory (LSTM) and Naïve Bayes Classifier for the metadata classification [6], and the use of convolutional networks to detect missing data [7].

This paper examines tasks related to the field of data profiling & story telling - determining the data type (typesetting) and semantic description of data. The problem of the correct and complete data typesetting is critical for the subsequent effective work with data, and has several levels of complexity. In its primitive form, this is a comparison of the basic data types, which is considered in [6], but the most important ones are contextual data types created based on the data itself [8], e.g. lookups or categories of the possible abbreviations or so-called codifications, which is especially true for the codified products and engineering data [3]. The problem of typesetting without the ability to generate new data types was considered for the special case of the classification problem for medical and biotechnological data [9], where comparisons of various solving techniques are provided, including support vector machines (SVM), clustering and an adversarial generating network GAN (Generative Adversarial Network). GAN architecture using LSTM mechanisms to search for anomalies in time series data for large data sets in various industries is proposed in [10].

Below a generalized typesetting problem is considered, including the generation of new data types (lookups and registries) based on their features in the form of a discrete optimization problem with various target functions proposed. For the latter, certain criteria are formulated to ensure the fulfillment of Bellman's discrete optimality principle, which allows the use of both dynamic programming and TD learning methods to solve the original discrete optimization problem. Thus, an effective scheme is used to find the extremum of the target function within each training epoch. The architecture of the deep neural network of generating autoencoder using memory and attention mechanism allowed to create a data description based on historical information and analysis of the attribute values, which is especially important when working with changing over time and streaming data.

The problems under consideration are of practical importance, because they are integral parts of the data management projects at any level, since all common approaches and data management practices always begin with data profiling and analysis phases [3,11,12]. Usually, so-called subject matter experts and data specialists are involved with such problems, which lead to high costs and time consumption, and in its turn extend the whole projects drastically. The algorithms and architecture of the deep neural network proposed below are automated solutions to these problems, which are capable of learning and rebuilding with the receipt of new data or constantly updating data on regular basis. The experiments with various data sets were carried out using open-source machine learning software libraries including TensorFlow and Keras having a goal of achieving applicable results matching manual analysis of the data types and semantic description without performance analysis, which is subject for the further investigations.

The article is structured as follows. Section 2 formalizes the concept of the data type, its characteristics for an abstract source of information (which can be streaming data, unknown uploads, so-called "black box data", reverse engineering results, etc.), and it formulates the generalized problem data typesetting, for which the fulfillment of Bellman's discrete optimality principle is proven. Next, in Section 3, the architecture of a computational neural network for the defined problem is given. Section 4 is devoted to the problem of generating the semantic description of the data attributes, which solution architecture turns out to be similar to the Section 3 architecture. Section 5 of the publication provides a combined GAN architecture with the soft attention and long-term memory mechanisms to simultaneously solve both problems.

2. Formal statement of the data typesetting problem

Let's consider structured data in the form of a marked file as input. The task of classical data profiling is to determine "what kind of data it is," i.e. for all of its attributes find out its data types, structure, most frequently used values, data completeness, inter-relation and dependencies of these attributes and other parameters [3,11]. The task of the descriptive part (also called as "data story telling") is to find out the semantic meaning of this data, that is, what exactly the values of a particular attribute mean, and how they are changing over time. In this section a number of definitions and functions for the formal definition of the problem are introduced.

First, let's make some assumptions. As a rule, the general meaning of the input data is clear, and the collection of so-called technical metadata is usually done first [11], i.e. this could be analysis of a file or DBMS system tables. However, this method is not applicable everywhere (the input can be a direct upload to a file in the form of textual and numeric information). Therefore, let's assume that:

1) the input data is structured by attributes (which data type is to be determined) and their values are atomic (it means that the value is not split into multiple attributes), their number is known - n, as well as the total number of records – l;
2) the affiliation of a particular value to any data type is expressed by a certain set of characteristics (*features*), and for elementary standard data types they are already known [8,13]. Obviously, a value can be interpreted in more than one way in terms of data types, for example, a string value with a date.

Neither the first nor the second assumptions are the simplifications of the problem, because there are many algorithms and tools to parse data into attributes, including machine learning methods [14], which provide this initial assumed information.

Thus, let's consider the input data X to be a matrix of elements over the $\mathcal{X}$ space of the dimension $l \times n$, and X_i is the corresponding vector with the values of the i^{th} attribute.

To identify features, let's will treat each feature as a function f_k: $\mathcal{X} \to [0,1]$, where $f_k(x_{ij}) = 0$ means that the specified feature is not fulfilled for a given element, and 1 means that it is fulfilled. Some features may have non-binary interpretations and produce a so-called "scoring" value from 0 to 1. The set of features is finite, and in the future, at each iteration, the neural network will learn, i.e. supplement and update the feature list based on the results obtained. Thus, at the training iteration m, the list of features will look like $F^m = \{f_k^m\}_0^{Km}$, where the vector $f_k^m(X_i)$ contains the result of matching all values of the i^{th} attribute to the k^{th} feature at the iteration m.

At the beginning of the training, there is a certain set of elementary data types - T^0 and, accordingly, an initial list of features F^0. Similar to features, at each training iteration, the set of types T^m will be updated. Each data type is described by a certain set of characteristics, namely:

$$\forall t \in T^m \; \exists! \, F_t^m \subseteq F^m \text{and if } t_1 \neq t_2, \text{ then } F_{t_1}^m \neq F_{t_2}^m. \tag{1}$$

Let's say that a certain value x has a pure type t if the following is satisfied $\forall f \in F_t^m \; f(x) = 1$. As noted, some features may not have discrete feature function value, at the same time some values of X might be "noisy", in which case the value of the feature function will only be close to 1, but never equal 1.

As mentioned above, it is quite likely that several types are suitable for some attribute values but the problem considered here is to determine the type for the entire attribute as accurately as possible. When measuring features for the attribute values, so-called attribute *feature measurement matrix* is obtained: $MF_i^m = \{f_k^m(x_{ij}) | f_k^m \in F^m, x_{ij} \in X_i, k \in 1{:}K_m, j \in 1{:}l\}$. The set of such matrices will be denoted by MF. For each data type, let's introduce a matrix operator to work with these matrices:

$$TF{:}\, MF \times T^m \to \mathbb{R}^l \times \mathbb{R}^{K_m} \mid TF(MF_i^m, t) = \{f_k^m(x_{ij})\}, if \; f_k^m \in F_t^m \tag{2}$$

$and\ \mathbb{0}\ if\ f_k^m \notin F_t^m\}$

Thus, for each data type, the operator TF "filter out" the feature measurement matrix and resets the values for those features that are not included in the argument type t.

To formally state the typesetting problem in the form of an optimization problem, let's introduce the target function that measures "the affiliation" of an attribute X_i to the data type t - $c^m: \mathcal{X} \times T^m \to \mathbb{R}$. Obviously, this is a family of target functions, and, as will be shown below, it does not make sense to change the target functions during iterations. Thus, the formal definition of a simple typesetting problem at iteration m becomes a problem of maximizing the target function c^m.

Definition 1. A *simple typesetting problem* for a given attribute X_i at iteration m, a set of types T^m and a list of features F^m is to find such a type $t_i^m \in T^m | \ t_i^m = \underset{t \in T^m}{argmax}\, c^m(X_i, t)$.

The typesetting problem defined this way is the simplest classification problem, formulated as an extremal problem, for which various solution methods can be proposed [15]. However, applied industries require the ability to create new data types based on the input that did not exist initially [4,7]. Usually these are contextual data types, e.g. lookups and registries. According to (1) at each next iteration the set T^{m+1} can be supplemented in such a way as to maximize the value of the target function in subsequent iterations. Accordingly, let's supplement it with all such sets of features whose value of the target function for any attribute i will be above certain threshold η_i^m (which will also be a parameter of iteration m):

$$T`^{m+1} := T^m \cup \left\{ \bar{t} \leftrightarrow F_{\bar{t}}^{m+1} \subseteq F^m \middle| \exists i\ that\ c^m\left(X_i, \bar{t}\right) \geq \eta_i^m \right\} \quad (3)$$

At each iteration m, the features list F^m can be expanded (so called hidden or latent features could be added to the features list), new features that are used for more accurate definition of the data type of an attribute can be added [14]. Thus, the additional task of new data types generation is to solve a suboptimal discrete optimization problem with a target function c^m on all subsets of the features list F^m. This can be done efficiently by re-using the results of the previous iterations, which will require the target function c^m to satisfy a certain property below. Let us denote θ^m - the set of optimization parameters, and η^m - the set of iteration m threshold values, then the target function must satisfy the following condition:

$c^{m+1}(X_i, t) = c^m(X_i, t) + v(X_i, F^{m+1}, \theta^{m+1}, \eta^{m+1})$, where the function v depends on the parameters of the current iteration only and is non-negative. (4)

Definition 2. *Generalized Typesetting Problem.* Let the primary sets of types T^0 and features F^0 are given. Assuming that the results of the previous iteration m of the generalized problem are known - T^m, F^m. The iteration $m+1$ task is to determine the data type for X_i, namely $t_i^{m+1} = \underset{t \in T^m}{argmax}\, c_i{}^{m+1}(X_i, t)$, where c_i is the target function used for the i^{th} attribute, for which the property (4) is satisfied, and further obtain a new feature set $F^{m+1} \supseteq F^m$, and generate new data types $T^{m+1} \supseteq T^m$ for which (3) is satisfied based on these new features.

In the general definition 2 we consider that it is possible to use different target functions for different attributes. However, in practice, when testing the architecture described below, a single target function based on the Frobenius metric (5) was used.

Note, that it is intentionally calculated the data type t_i^{m+1} first, and update the sets of data types T^{m+1} and features F^{m+1} afterwards. The property (4) introduces the concept of transition between iterations in such a way that it allows us to formulate the Bellman recurrence relation for the target function c^{m+1}, and

the fact that it does not deteriorate (this is a reason why non-negative of v is required in (4)) and its changes are influenced solely by the parameters of the current iteration leads us to the following important statement.

Statement 1. Bellman's discrete optimality principle is satisfied for the generalized typesetting problem defined above.

This Statement 1 allows us to use various methods to solve the generalized typesetting problem. For small dimensions, the most effective method is dynamic programming [16]. However, for a large number of parameters, which is typical for applied tasks, finding all possible combinations of the dynamic programming method becomes computationally complex, so TD- and Q- learning turns out to be ones of the most effective strategies [17].

The choice of the target function is important from the convenience and simplicity of its calculation points of view, and the condition (4) seriously simplifies the solution of the discrete optimization problem. The universal target function is the Frobenius metric of the matrix $TF(MF_i^m, t)$:

$$c_F(X_i, t) = \sqrt[2]{\sum_{k=1}^{K_m} \sum_{j=1}^{l} TF(MF_i^m, t)} \tag{5}$$

Statement 2. Using (5) as the target function of the generalized function typesetting problem $c^m(X_i, t) = c_F(X_i, t)$ is possible since it satisfies condition (4).

This statement obviously follows from the fact that all the values of the matrix $TF(MF_i^m, t)$ are non-negative, and the fact that $F^{m+1} \supseteq F^m$ means that the matrix $TF(MF_i^{m+1}, t)$ will be obtained from the corresponding matrix of the previous iteration by adding several columns with non-negative values, which provides the desired values for v.

Different target functions can be used for different data attributes. For example, using of so-called "L^0 metric" could be promising for some attributes, that is, counting non-zero elements of a matrix [18], for which (4) is also satisfied. For further exposition and corresponding implementation of the architectures given below c_F will be used as a target function.

3. Neural network architecture for the generalized typesetting problem

The architecture of the computing neural network for a generalized typesetting problem is considered in this section. The use of c_F is proposed as the optimization of the target function (5), and the algorithm iteratively updates the sets T^m, F^m and the sets of parameters: vectors for the Generator and Discriminator - $\theta^m = (\theta_d, \theta_g)$, and the thresholds η^m.

Talking about discrete data in general, two approaches are possible depending on whether the density of the distribution of the attribute values X_i by the already defined data types is known. Following the classification of the generative models from [18], the first approach defines models with the explicit density, and given the discreteness of the values, variational approximations (VAE) are usually used. However, in the case of a generalized typesetting problem, it is impossible to predict the distribution by the newly generated data types since they are being produced iteratively and the whole distribution is changing at each iteration, so the architecture of generative networks (GANs) is proposed below. Also, it is important to note that the computing (neural) network should be able to generate examples that correspond to the initial model [4,5], which is fully satisfied here based on the assumptions below and the fact that data is meaningful in each particular attribute.

In the classical GAN architecture, the Encoder encodes the values of the vector X_i and the values of the measured features MF_i^m into one of the known types T^m, but does not generate T^{m+1}since there is nowhere to get suitable combinations of the hidden features, and if generated randomly, then reasonable data types cannot be achieved. Therefore, modified architecture of the generative autoencoder (AAE -

adversarial autoencoder) is introduced below, which adds a Discriminator that tries to distinguish the generated distribution of the encoded features from the encoded real value along with its feature values. It will also be convenient to entrust the Discriminator with the responsibility of making a decision on updating hidden features F^{m+1} and generating data types for the next iteration T^{m+1} based on them.

To generate data, let's take the noise vector z of a multidimensional normal distribution $\mathcal{N}(0,1)$ over the feature space: $G(z,\theta_g)$: $F^m \rightarrow \overline{\mathcal{X}}$, where θ_g is a set of optimization parameters from vectors of the appropriate dimension to perform a linear transformation operation with vectors of dimension K_m: $z \oplus \theta_g = z * \theta_g^1 + \theta_g^2$, and $\overline{\mathcal{X}}$ is the space of all possible values of the input data of the problem, including all the input values $\mathcal{X} \subseteq \overline{\mathcal{X}}$. Two transformation layers for the noise vector z are required, since it is impossible to obtain a normal distribution with a help of linear transformation, so for the activation function to take is $softplus(x) = \log(e^x + 1)$:

$$G(z,\theta_g) = log(e^{z \oplus \theta_g} + 1) \oplus \theta_g \tag{6}$$

Comment. It is important to note the fact that values are generated this way, specifically over the space of all features F^m, but not the data types of the current iteration T^m, ensures the subsequent update of the features set F^{m+1} and, accordingly, set of data types T^{m+1}. Otherwise, the model won't be able to find additional features and thus generate the new data types.

The task of the Discriminator is to distinguish the input vector - either generated or encoded real value X_i and the values of its features MF_i^m, namely, $D(x,\theta_d)$: $\overline{X} \rightarrow [0,1]$, where θ_d is a similar set of optimization parameters. And the goal of the Discriminator is to maximize the value D on X_i, and to minimize it on the generated values. Let us use the sigmoidal activation function $\sigma(x) = 1/(1+e^{-x})$ for the parameterized values of the analyzed vector, to which the hyperbolic tangent was applied:

$$D(x,\theta_d) = \sigma((\tanh(\tanh(x \oplus \theta_d) \oplus \theta_d) \tag{7}$$

Following the minimax principle (sometimes called "game") of the Discriminator and Generator, let's define the target function (following the original GAN definition in [18]) using the averaging operator $\mathbb{E}$ over the input values and noise correspondingly:

$$\min_G \max_D V(D,G) = \mathbb{E}[\log(D(x,\theta_d))] - \mathbb{E}[\log(1 - D(G(z,\theta_g),\theta_d))] \tag{8}$$

The Decoder should solve the extreme problem of determining the best data type for the attribute X_i for the previously selected target function c_F:

$$t_i^m \in T^m \mid t_i^m = \underset{t \in T^m}{argmax}\, c_F(X_i,t) = \underset{t \in T^m}{argmax} \sqrt{\sum_{k=1}^{K_m} \sum_{j=1}^{l} TF(MF_i^m,t)}^2 \tag{9}$$

Additional tasks that are assigned to the Discriminator are the following: (a) making decisions on updating the set of features F^{m+1} based on the newly found latent ones, and (b) generating new data types T^{m+1}, by solving the suboptimal problem (3). The architecture of the modified adversarial generative autoencoder is presented in Diagram 1.

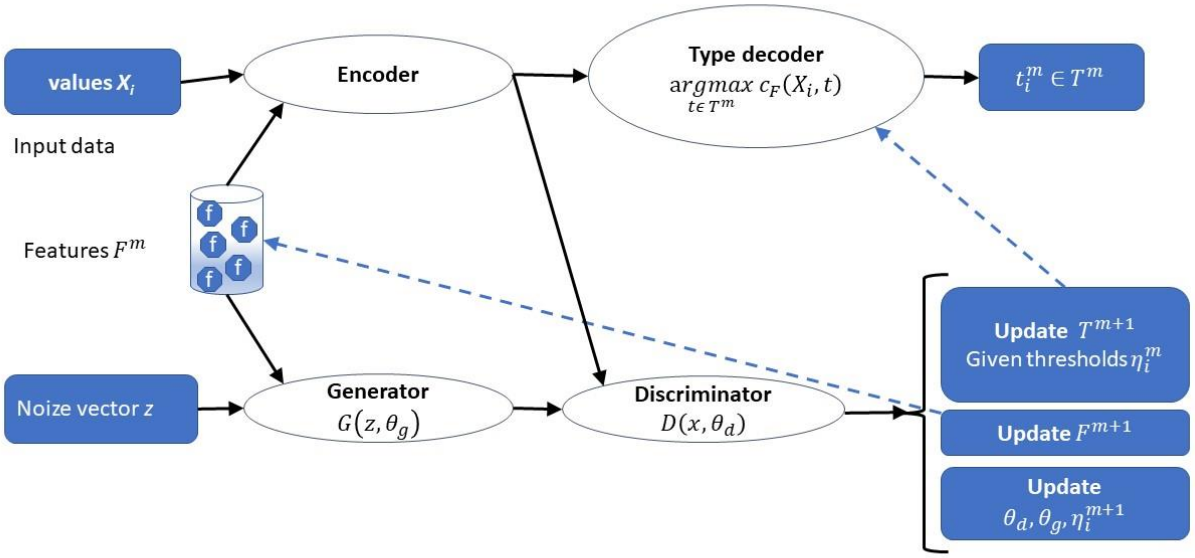

Diagram 1. Architecture of an adversarial autoencoder for the generalized typesetting problem

For the given architecture, it is possible to further configure separate learning mechanisms for the new type creation thresholds η_i^m at each iteration, but such complications may have a negative impact on the overall performance and require further analysis. When solving the problem of generating new data types, the technique of enumerators of suboptimal solutions [19] can also be used, which allows us to sort solutions of (3) according to the degradation of the target function with the help of effective algorithms for enumerators operations.

4. Neural network architecture for the problem of semantics of data attributes

This section investigates a problem of generating a data attribute meaningful description, which is a key one in data management [3,11,13] and its proposed solution turns out to be close to the architecture of the abovementioned generalized typesetting problem. Similar problems have been considered from different points of view previously, e.g. ontological approach, and linguistic ones, including using neural networks [20]. Recently, the use of generative neural networks becomes a relevant approach due to increased computing power.

Let's consider the architecture of a generative autoencoder that encodes the values of a vector X_i and the features measurements on that vector - MF_i^m, and then decodes it into a textual description S_i^m. Taking into account the fact that the description of the semantic part of an attribute is usually based on the set of its values, let's use some memory mechanism to store the "already read and processed" values. Different types of memory mechanisms as well as their advantages and disadvantages will be considered after the architecture itself.

This architecture turns out to be similar to the classic "Show, Attend and Tell" algorithm for the problem of determining captions for images. [21] proposed an architecture consisting of a set of convolutional networks with attention (to focus on individual fragments of the picture) and memory layer, which is presented in the Diagram 2 below.

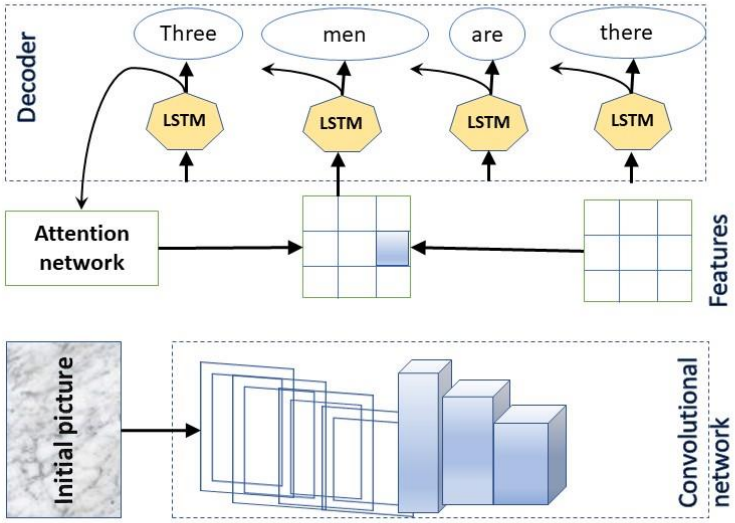

Diagram 2. Architecture of the Show, Attend and Tell model

To solve the problem of semantic description generation of an attribute let's modify the architecture from [21] and make the following changes:

1. In the problem being solved, it is not a picture that is analyzed, but a set of sequential attribute values X_i. Therefore, let us use layers of a simple unidirectional recurrent network (RNN, Recurrent Neural Network) instead of convolutional layers from Diagram 2. To simplify the problem, we assume that the order of the values is not important. However, in the case of streaming data and metadata changing over time, the values order becomes important and is a promising direction for further research;
2. As already noted, a long-term memory mechanism is needed to store the already processed values without regard to their order. One option is to use LSTM memory cells [22] for GAN networks [10]. However, they require quite a lot of computing resources, so the use of the GRU or MUT1 architecture [23] for memory cells looks more promising. During the implementation and experiments, one of the most modern memory architectures (SCRN) was used since its performance was proven with large amounts of data [24-25], and it turned out to be "cheaper" to train vs. LSTM. Analysis of various memory cell options[1]is out of scope of this publication and remains a subject for the further analysis and experiments.
3. Similar to [21], an attention mechanism is required to "focus" at the desired attribute values and "remove noisy" values. Unlike the typesetting problem, mechanisms with soft attention [18] are suitable here since the descriptive part can be "softer", namely a linear combination of values, rather than discrete. Let's design the attention mechanism as a separate neural network in order to cut off unnecessary values when generating the descriptive part of the semantics. Thus, a vector of read values $\overline{X_i}$ is supplied as an input for the recurrent network after removing noisy and atypical values. These values along with the measurements of the feature matrix MF_i^m are further passed through the attention network, where attention coefficients a_{ij}^m are obtained. Thus, a linear combination $\sum_j a_{ij}^m \, x_{ij}$ is received as the input for the Decoder memory cells.

[1]In addition to the already mentioned GRU, MUT1, the SRU (Simple Recurrent Unit) model may also be of interest.

Combining all of the above changes together based on the generative autoencoder proposed in [21], the following architecture is obtained:

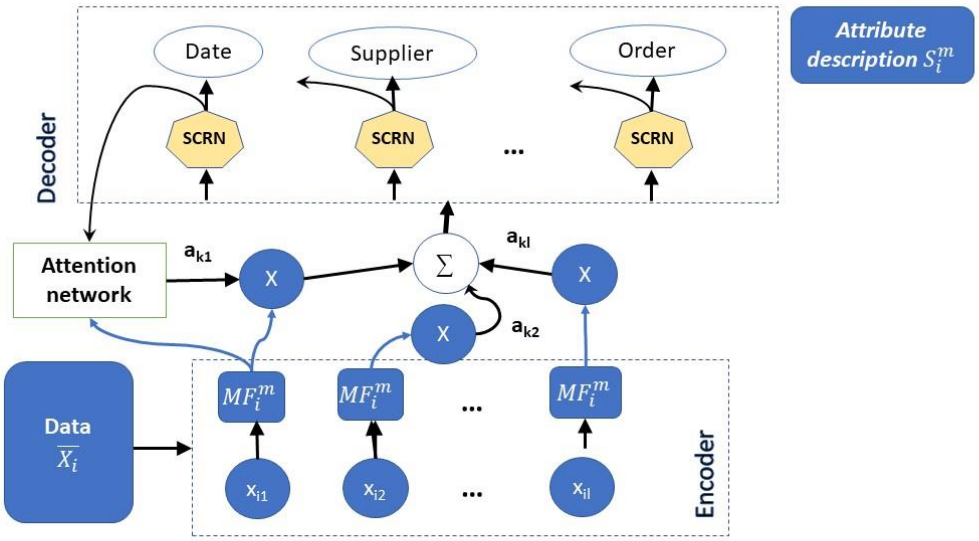

Diagram 3. Autoencoder architecture for the attribute semantics description generation problem

Diagram 3 shows the architecture of the deep network at iteration m, where any descriptive element can be changed to reflect the new value have read with every new iteration. Note, that the data type information is also relevant to the semantic description of the attribute. This observation and a similar AAE architecture can effectively solve both problems simultaneously within a single architecture.

5. A single neural network to solve both problems

Let us outline common elements of the neural network architectures proposed above for the generalized typesetting and the data attributes semantics generation problems: common cells of long-term memory SCRN, main recurrent network of the Encoder and the feature measurement matrix MF_i^m. At the same time, the attention mechanism from the architecture of the data attributes semantics generation is not useful in the typesetting problem, since it serves to search for the focal areas of the values of a particular attribute, which allows generating its descriptive part. Similarly, the Generator and Discriminator of the generalized typesetting problem cannot be reused due to the specifics of the additional tasks being solved by them. Let's combine these two architectures to optimize computing resources and learning time on the training data.

The adversarial autoencoder architecture of the generalized typesetting problem did not use long-term memory cells explicitly. However, they were used implicitly by the Discriminator as a "statistics cache" for the features list F^m to determine a new type. Considering that there is already a memory layer of SCRN cells, it is quite logical to use it to cache Discriminator's values. In this case, the task of determining new data types (3) will be solved much faster without additional recalculations. A good example is the new lookup type, where the memory layer stored in SCRN cells simply accumulates the values of the future lookup.

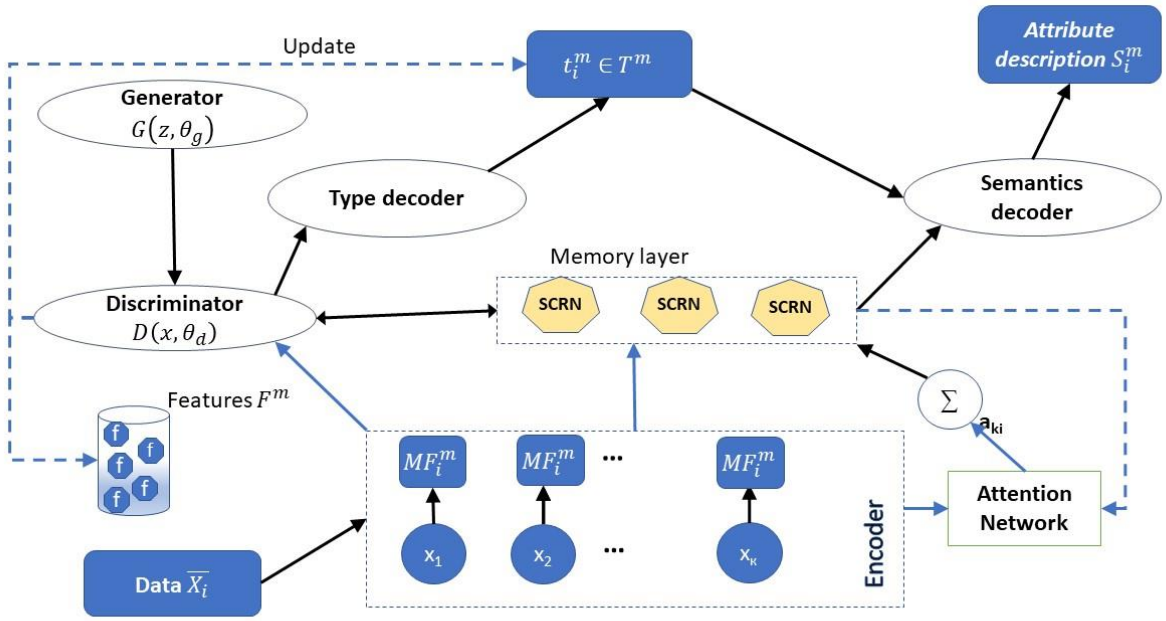

Diagram 4. Architecture of the combined neural network of the generalized typesetting and attribute semantics generation problems

6. Conclusion

Well-known industry metadata management problems for structured data and master data are considered, namely, generation of the meaningful description (semantics) of the attributes and identifying their data types, taking into account the necessity of creating additional reference data types. This paper proposes an algorithm to simultaneously solve these problems based on a single deep computing network, which is complemented by certain mechanisms for data types generation, and generation of the semantic descriptions of the data attributes.

A formal definition of the generalized problem of determining the data type (typesetting) is introduced in the form of a discrete optimization problem, for which the Bellman optimality principle is proven. To solve the problem, the architecture of a neural network based on an adversarial autoencoder (AAE) is presented, where the dynamic programming method is used within iterations to search for discrete suboptimums. Further combined architecture is proposed for the generalized typesetting and data attribute description generation problems, which uses shared layers and is supplemented by a unified attention network and a layer of long-term memory SCRN cells.

This work is based on the experiments with data from various industries, in particular smartphone log data [12] and online trading transaction data [26]. The program code of the specified architecture is implemented with the Python programming language using the open-source libraries TensorFlow and Keras. The main emphasis of the publication is the definitions of the problems in the forms of discrete optimization problems and the architecture of a unified deep learning neural network with well-known memory and attention blocks. At the same time, some directions for the future research are highlighted – e.g. dependent attributes identification, effective work with long-term memory, tuning the latent feature search, auto-tuning of the threshold values of the Discriminator.

References

[1] Kantorovich L. V. "Mathematical methods of organizing and planning production." Management science 6.4, 1960, pp. 366-422 (in Russian).

[2] Jain A. et al. Overview and importance of data quality for machine learning tasks //Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. – 2020. – pp. 3561-3562.
[3] International D. DAMA-DMBOK: data management body of knowledge. – Technics Publications, LLC, 2017.
[4] Kumar A., Boehm M., Yang J. Data management in machine learning: Challenges, techniques, and systems //Proceedings of the 2017 ACM International Conference on Management of Data. – 2017. – pp. 1717-1722.
[5] Thirumuruganathan S. et al. Data Curation with Deep Learning //EDBT. – 2020. – pp. 277-286.
[6] Pavia S. et al. Hybrid Metadata Classification in Large-scale Structured Datasets //J. Data Intell. – 2022. – T. 3. – №. 4. – pp. 460-473.
[7] Khan H., Wang X., Liu H. Handling missing data through deep convolutional neural network //Information Sciences. – 2022. – T. 595. – pp. 278-293.
[8] Stonebraker M. Inclusion of new types in relational data base systems //Readings in Artificial Intelligence and Databases. – Morgan Kaufmann, 1989. – pp. 599-606.
[9] Purandhar N., Ayyasamy S., Siva Kumar P. Classification of clustered health care data analysis using generative adversarial networks (GAN) //Soft Computing. – 2022. – T. 26. – №. 12. – pp. 5511-5521.
[10] Zhu G. et al. A novel LSTM-GAN algorithm for time series anomaly detection // 2019 prognostics and system health management conference (PHM-Qingdao). – IEEE, 2019. – pp. 1-6.
[11] Kuznetsov S., Konstantinov A., Skvortsov N. The value of your data, Alpina PRO Publishing House, 2022 (in Russian).
[12] Reyes-Ortiz Jorge, Anguita Davide, Ghio Alessandro, Oneto Luca, and Parra Xavier. (2012). Human Activity Recognition Using Smartphones. UCI Machine Learning Repository. https://doi.org/10.24432/C54S4K.
[13] Li J. et al. Feature selection: A data perspective //ACM computing surveys (CSUR). – 2017. – T. 50. – №. 6. – pp. 1-45.
[14] Chen, R. C., Dewi, C., Huang, S. W., & Caraka, R. E. (2020). Selecting critical features for data classification based on machine learning methods. Journal of Big Data, 7(1), 52.
[15] Bengio Y., Goodfellow I., Courville A. Deep learning. – Cambridge, MA, USA : MIT press, 2017.
[16] Romanovsky I.V. Algorithms for solving extremal problems. – 1977. (in Russian).
[17] Yu, Huizhen, A. Rupam Mahmood, and Richard S. Sutton. "On generalized bellman equations and temporal-difference learning." The Journal of Machine Learning Research 19.1. - 2018. - pp. 1864-1912.
[18] Goodfellow, I. NIPS 2016 tutorial: Generative adversarial networks. arXiv 2016. arXiv preprint arXiv:1701.00160.
[19] Kuznetsov S.V., Summation of enumerators in discrete optimization problems in the context of master data management // Differencialnie Uravnenia i Protsesy Upravlenia. – 2023. – No. 4. – pp. 42-52. (in Russian).
[20] Dudar Z. V., Shuklin D. E. Semantic neural network as a formal language for describing and processing the meaning of texts in natural language // Radioelektronika i informatika.– 2000. – No. 3 (12). – P. 72-76. (in Russian).
[21] Xu K. et al. Show, attend and tell: Neural image caption generation with visual attention //International conference on machine learning. – PMLR, 2015. – pp. 2048-2057.
[22] Gers F.A., Schmidhuber J, Cummins F. Learning to Forget: Continual prediction with LSTM // Neural Computation, 2000, vol. 12 no.10, pp. 2451-2471
[23] Jzefowicz R., Zaremba W., Sutskever I. An empirical exploration of Recurrent Network Architectures // Proc. 32nd ICML, 2015, pp. 2342 – 2350
[24] Mikolov T. et al. Learning longer memory in recurrent neural networks //arXiv preprint arXiv:1412.7753. – 2014.

[25] Lei T., Zhang Y., Artzi Y. Training RNNs as fast as CNNs. – 2018.
[26] Chen, Daqing. (2019). Online Retail II. UCI Machine Learning Repository. https://doi.org/10.24432/C5CG6D.