Pretrained Coles model quility based on the input data amount (Machine Learning 2023 Course)

Alexander Zubrey [1] **Ksenia Kuvshinova** [1] **Anastasia Volkova** [1] **Olga Volkova** [1] **Anastasia Grigoreva** [1]

Abstract

This paper is dedicated to studying the effect of pre-train data size, model size on model accuracy and NN saturation. We applied a self-supervised learning method for embedding discrete event sequences called COntrastive Learning for Event Sequences (CoLES)(**?**), which is based on contrastive learning with a specific data augmentation strategy. We carried out several experiments using this method on different data sets, which showed that at about 40% of the pretrain data, the model reaches a "plateau" and the accuracy of the model changes insignificantly. And also when the model size was changed, NN saturation in self-supervised learning was achieved at half the value of the original model size.

Github repo: Github

1. Introduction

In recent years, there has been a growing interest in the analysis of event sequences in various fields, including medicine, finance, and social sciences. The classification of event sequences is a critical task in these fields, as it allows for the identification of patterns and trends in data that can inform decision-making processes. However, this task can be challenging due to the discrete nature of event sequences and the complex dependencies between events.

Deep neural networks have shown to be effective in various tasks related to event sequence analysis. However, training models based on neural networks requires a large amount of data, which can be challenging when there is a limited amount of labeled data available. This leads to the need for different methods to train models on unlabeled data. In this paper, we study the effect of data preprint size and model size on model accuracy and neural network (NN) saturation.

To embed discrete event sequences, we applied a self-supervised learning method called COntrastive Learning for Event Sequences (CoLES)(**?**), which is based on contrastive learning with a specific data augmentation strategy. Contrastive learning is a technique that involves determining whether two objects are from the same class or different classes. This task helps the model identify similarities and differences between objects. The majority of embedding methods have been primarily applied and studied in traditional machine learning areas such as natural language processing (NLP) with models like ELMO(**?**) and BERT(**?**), speech recognition with CPC(**?**), and computer vision with various methods.CoLES utilizes contrastive learning to embed discrete event sequences into a low-dimensional space, which can be used for further analysis.

CoLES is a self-supervised learning method, which means that it can learn from unlabeled data. This can be helpful when labeled data is scarce or expensive to obtain. CoLES also employs a data augmentation strategy, which helps to enrich the data and reduce noise, leading to higher quality models.

In this work, we evaluate the performance of CoLES on several benchmark datasets for event sequence analysis. Our goal is to investigate the influence of pre-train data size and model size on model accuracy and NN saturation.

The main contributions of this report are as follows:

1. For models with contrastive learning based on CoLES, we observed a "plateau" at about 40% of the pre-train data without a significant change in model accuracy when the pre-train data were further increased.

2. For CoLES-based contrastive learning models, when the model size was changed (changing the number of hidden layers), the NN saturation in self-supervised learning was achieved at half the value of the original model.

2. Related work

CoLES method is an adaptation of constructive learning to the problem of self-supervised learning on discrete event

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Alexander Zubrev <Alexander.Zubrey@skoltech.ru>.

Final Projects of the Machine Learning 2023 Course, Skoltech, Moscow, Russian Federation, 2023.

sequences. Previously constructive learning was used for audio and computer vision domains. Consider the self-supervised method Contrastive Prediction Coding (CPC) suggested for non-discrete sequential data.(**?**) This approach is able to learn useful representations achieving strong performance on four distinct domains: speech, images, text and reinforcement learning in 3D environments. The advantages of this method are the simplicity and low computational requirements to train the model, together with the encouraging results in challenging reinforcement learning domains when used in conjunction with the main loss. CPC combines autoregressive modeling and noise-contrastive estimation with intuitions from predictive coding to learn abstract representations in an unsupervised fashion.

In the article (**?**), was studied the problem of attributed sequence embedding, where the aim is to learn the representations of attributed sequences in an unsupervised fashion. This problem is core to many important data mining tasks ranging from user behavior analysis to the clustering of gene sequences. This problem is challenging due to the dependencies between sequences and their associated attributes. The authors identify the three types of dependencies in attributed sequences. They proposed a novel framework, called NAS, to learn the heterogeneous dependencies and embed unlabeled attributed sequences, that effectively boosts the performance of outlier detection tasks compared to baseline methods.

Language representation model called BERT is described in the paper(**?**). BERT is Bidirectional Encoder Representations from Transformers. BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pretrained BERT model can be fine tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications.

Another existing model is ELMo - Embeddings from Language Models(**?**). Extensive experiments demonstrate that ELMo representations work extremely well in practice.Authors show that they can be easily added to existing models for six diverse and challenging language understanding problems, including textual entailment, question answering and sentiment analysis. The addition of ELMo representations alone significantly improves the state of the art in every case, including up to 20% relative error reductions.

CoLES considers more complex sequences of events where an element of the sequence consists of several categorical and numerical fields, than the aforesaid works consider sequences of "items", where each element is an item identifier.

3. Algorithms and Models

3.1. Algorithms and Methods

Contrastive Learning of Event Sequences (CoLES) is a self-supervised sequence embedding method that uses an innovative augmentation algorithm to create sub-sequences of observed event sequences. These sub-sequences are then used as different high-dimensional views of the object (sequence) for contrastive learning. The generative process used by CoLES is specifically designed to address the observed interleaved periodicity in financial transaction event sequences, which is the primary application of the method. By using this process, CoLES can learn robust and meaningful sequence embeddings that can capture complex patterns and dependencies in the data.

The transactions data is the bench of discrete events sequences and for their processing we forced to use the special data augmentation strategy. CoLES method, which based on contrastive learning, was used for embeddings creating. Contrastive learning operates with representation $x \mapsto M(x)$ which brings similar objects closer to each other in the embedding space(positive pairs) and dissimilar objects (negative pairs) further.

The theoretical framework of contrastive learning proposes that each entity e is a latent class associated with a distribution P_e over its possible samples (event sequences). However, there are no positive pairs available, i.e. pairs of event sequences representing the same entity e. Instead, we only have a single sequence x_e available for each entity.

Each entity e is formally linked to a hidden stochastic process $\{X_e(t)\} = \{X_e(t)\}_{t\geq 1}$ and we can only observe a single finite realization $\{x_e\} = \{x_e(t)\}_{t=1}^{T_e}$ of this process. Our goal is to use the encoder M which is learned to map event sequences into a feature space R in such a way that the obtained embedding $\{x_e\} \mapsto c_e = M(\{x_e\}) \in R$ encodes the essential properties of e and disregards irrelevant noise contained in the sequence. The quality of the representations can be examined by downstream tasks in two ways: (1) c_e can be used as a feature vector for a task-specific model, and (2) the encoder M can also be fine-tuned jointly. This can be seen in Figure 1, where Phase 2a shows the use of c_e as a feature vector for a task-specific model, while Phase 2b shows the joint fine-tuning of the encoder M. By using this framework, CoLES can learn robust and meaningful sequence embeddings that capture the essential properties of the entities and disregard irrelevant noise contained in the sequence.

In cases where there is no access to the latent processes $\{Xe(t)\}$, synthetic augmentation strategies can be employed as a form of bootstrapping. However, most augmentation techniques proposed for continuous domains, such

as image displacement, color jitter, or random gray scale in CV, are not applicable to discrete events. Therefore, generating sub-sequences from the same event sequence $\{xe(t)\}$ can be used as a possible augmentation strategy. The proposed Random Slices sampling method applied in CoLES is motivated by the periodicity and repeatability observed in event sequences that represent lifetime activity. This method involves sampling sub-sequences $\{xe\}$ from a given sequence $\{xe(t)\}$ as continuous segments or "slices" using the following three steps: (1) selecting the length of the slice uniformly from admissible values, (2) discarding too short (and optionally too long) sub-sequences, and (3) uniformly choosing the starting position from all possible values. The overview of the CoLES method is presented in Figure 1 and Figure 2.

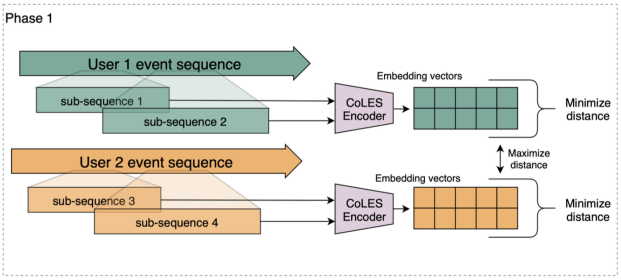

Figure 1. General framework. Phase 1: Self-supervised training. (**?**)

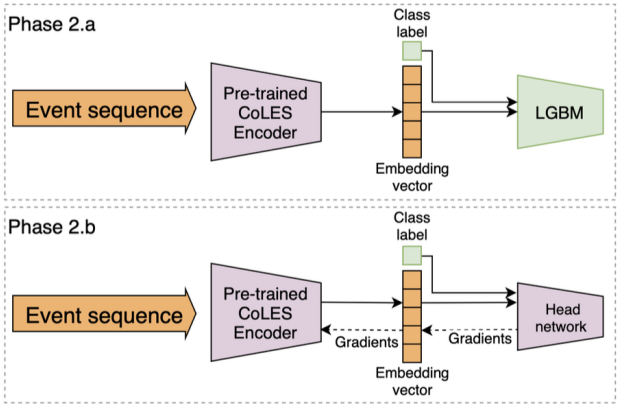

Figure 2. Phase 2.a Self-supervised embeddings as features for supevised model. Phase 2.b: Pre-trained encoder fine-tuning. (**?**)

The contrastive loss used in CoLES aims to minimize the objective function:

$$L_{uv}(M) = Y_{uv}\frac{1}{2}(u,v)^2 + (1-Y_{uv})\frac{1}{2}max\left\{0, \rho - d_M(u,v)\right\}^2$$

The function is minimized with respect to $M : X \to R^n$, where $d_M(u,v) = d(c_u, c_v)$ is the distance between embeddings of the pair (u,v), $c_* = M(\{\tilde{x}_*(\tau)\})$ is the embedding of the sequence, and Y_{uv} is a binary variable which sais is the pair (u,v) is positive. ρ is the soft minimal margin between dissimilar objects. The second term makes separation of embeddings in negative pairs to avoid of mapping the entities to the same point in the embedding space. In pairs with $Y_{uv} = 1$, the sequences $\{\tilde{x}_u(\tau)\}$ and $\{\tilde{x}_v(\tau)\}$ are obtained through random slice generation from the same observation $\{x_e(t)\}$. In pairs with$Y_{uv} = 0$, the sequences are sampled from $\{x_e(t)\}$ and $\{x_g(t)\}$, respectively, for $e \neq g$. $d(a,b)$ is Euclidian distance between two points $d(a,b) = \sqrt{\Sigma_k(a_k - b_k)^2}$.

To embed a sequence of events into a fixed-size vector, the individual events must first be encoded and then the entire sequence aggregated. In CoLES, the composite encoder model M takes the form $M(\{x_t\}) := \phi_{seq}(\{\phi_{evt}(x_t)\})$, where ϕ_{evt} and ϕ_{seq} are event- and sequence-level embedding networks, respectively. These networks are trained end-to-end to minimize the contrastive loss $L(M)$. The event encoder ϕ_{evt} takes the attributes of each event x_t and outputs its intermediate representation z_t in R^d. The encoder includes several linear layers for embedding one-hot encoded categorical attributes and batch normalization layers applied to numerical attributes. The outputs of these layers are concatenated to produce the event embedding z_t. The sequence encoder ϕ_{seq} takes the intermediate event representations $z_{1:T} = z_1, z_2, \cdots z_t$ and outputs the representation c_t of the sequence up to time t: $c_t = \phi_{seq}(z_{1:t})$. The last c_T is used as the embedding of the entire event sequence. In CoLES, the GRU is used as the sequence encoder, which is a recurrent network that performs well on sequential data. ϕ_{seq} is computed through the recurrence $c_{t+1} = GRU(z_{t+1}, c_t)$ starting from a learned c_0 In summary, CoLES includes three major components: the event sequence encoder, the positive and negative pair generation strategy, and the loss function for contrastive learning.

3.2. Data and Metrics

We operate on two datasets with bank clients transactions information. **Sber** dataset contains information about transactions of bank customers. About 27,000,000 million records in volume. Each entry describes one banking transaction. For each of the $\approx 20,000$ test id, the participants need to use the trained model to predict which of the age groups the Client falls into.

Training transactions_train.csv, in which the date, amount, type and id of the client are known for each transaction; Test transactions_test.csv containing the same fields: client_id is a unique client number, trans_date is the date of the transaction (it is simply the day number in chronological order, starting from the given date), small_group is a group of

transactions that characterize the type of transaction (for example, grocery stores , clothes, gas stations, children's goods, etc.), amount_rur - the amount of the transaction (for anonymization, these amounts were transformed without losing the structure). On the database of files, you can build various features that characterize age groups. The target variable for the training dataset is in the train_target.csv file. It contains information about the Client and the label of the age group to which it belongs: client_id – unique number of the Client (corresponds to client_id from the transactions_train.csv file), bins – age label. In the test.csv file, we need to predict the corresponding age group label for the specified client_id. We are also provided with an information file small_group_description.csv, which contains a breakdown of transaction types. Based on these data, it is necessary to carry out a multiclass classification (4 classes - from 0 to 3). The quality of the solution is calculated as the proportion of correctly guessed age labels for all test cases - accuracy.

Accuracy classification score is commonly used metric in multilabel classification tasks. It measures the proportion of correct predictions made by the classifier out of all predictions made: the set of labels predicted for a sample must exactly match the corresponding set of labels in true labels. A higher accuracy score indicates a better performance of the classifier in correctly classifying instances.

The **Rosbank** dataset contains information on about 520,000,000 million records. A record is a description of a particular customer's transaction.

We have with one training dataset train.csv, in which for each transaction the date, amount, transaction type and client id, date of commission, amount, target_flag and target_sum are known. test.csv containing the same fields: cl_id - unique client number, TRDATETIME - date of the transaction (number of the day in chronological order, starting from the specified date), trx_category - category of transactions characterizing the type of transaction, amount - transaction amount (for anonymization, these amounts were transformed without loss of structure). Based on the file database, it is possible to determine whether the customer will remain a user of the bank or not. The target variable for the training dataset is in the train.csv file. In the test.csv file, we need to predict the churn rate for the specified data. Based on these data, it is necessary to carry out a binary classification. The quality of the solution is calculated as the proportion of correctly guessed age labels for all test cases - AUC_ROC.

A ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate and False Positive Rate.

AUC is a commonly used metric in binary classification tasks. It measures the area under the receiver operating characteristic (ROC) curve, which plots the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds. A higher AUC score indicates a better performance of the classifier in distinguishing between positive and negative instances.

4. Experiments and Results

We utilized two datasets from data analysis competitions - one from Rosbank, where we aimed to predict customer churn after preferential card usage, and another from Sberbank, where we aimed to classify clients into age groups based on their spending habits. We preprocessed both datasets using pytorch-lifestream, with similar methods except for date formatting in the Rosbank dataset. We converted dates from string representations to ordinal numbers to facilitate analysis. Using the PandasDataPreprocessor module from pytorch-lifestream, we transformed the complete datasets (with and without targets) into the necessary format to receive embeddings. We then split the marked-up data into training and testing subsets and trained embeddings on a recurrent network as an encoder, which proved robust in handling sequential data. Subsequently, we performed embeddings inference on both subsets and applied classic machine learning models to them. In the case of Sberbank, we utilized average accuracy (as it was a multiclass classification task) as the target metric for the competition, whereas we used ROC-AUC (as it was a binary classification task) for Rosbank. The graph shows that the plateau was evident for Sberbank, whereas Rosbank was more complex, but we observed a clear trend. As a result, we achieved high scores with light gbm, which ranked us close to the top of the leaderboard. To surpass this score, we might need more intensive parameter tuning.

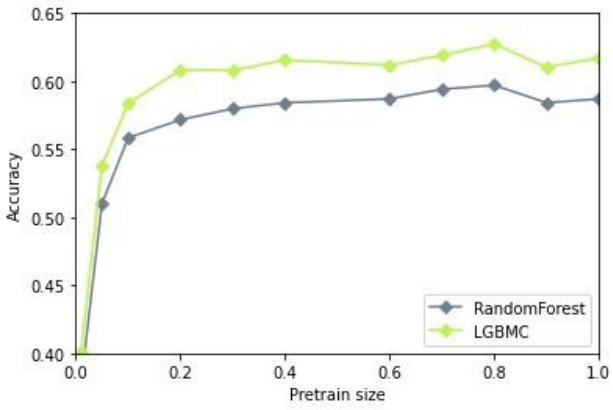

Figure 3. Model performance with different pretrained dataset sizes for Sber.

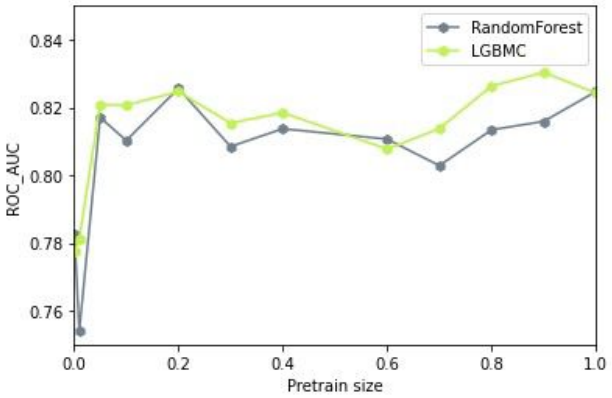

Figure 4. Model performance with different pretrained dataset sizes for Rosbank.

As for the hidden size, we obtain the following results:

Table 1. Model performance for different encoder sizes (Sber, accuracy).

HIDDEN SIZE	RANDFOREST	LGBM
64	0.569	0.603
128	0.592	0.617
256	0.587	0.612
512	0.584	0.623

Table 2. Model performance for different encoder sizes (Rosbank, ROC-AUC).

HIDDEN SIZE	RANDFOREST	LGBM
64	0.799	0.814
128	0.807	0.823
256	0.811	0.808
512	0.816	0.834

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Anastasia Volkova (20% of work)

- Experimenting with model size and pre-train size on Sberbank and Rosbank datasets
- Preparing the Section Algorithms and Models of this report
- Preparing slides Description of CoLES method for project presentation
- Presentation
- Performing the k-top recall metric calculation

Olga Volkova (20% of work)

- Experimenting with model size and pre-train size on Sberbank and Rosbank datasets
- Preparing the Sections Abstract and Introduction of the report
- Preparing slides Problem Statement for project presentation
- Presentation
- Graphs plotting

Kseniia Kuvshinova (20% of work)

- Implemented code for experiments
- Prepared Experiments and Results section

Alexander Zubrey (20% of work)

- Implemented code for experiments
- Prepared Algorithms and Methods section

Anastasia Grigoreva (20% of work)

- Presentation
- Report writing

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: 40% of code was written

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: see chapter Algorithms and models

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   Yes.
   ☑ No.
   ☐ Not applicable.

   Students' comment: See Github repo

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

9. The exact number of evaluation runs is included.

   ☐ Yes.
   ☑ No.
   ☐ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☐ Yes.
    ☐ No.
    ☑ Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    ☐ Yes.
    ☐ No.
    ☑ Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

    - ☐ Yes.
    - ☐ No.
    - ☑ Not applicable.

    Students' comment: None