Skoltech

Skolkovo Institute of Science and Technology

MASTER'S THESIS

Text interfaces for event sequences

Master's Educational Program: Data Science

Student

Andrei Filatov
Data Science
April 24

Research Advisor:

Ivan V. Oseledets
Full Professor

Moscow 2023

Skoltech

Skolkovo Institute of Science and Technology

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Текстовые интерфейсы для последовательности событий

Магистерская образовательная программа: Науки о данных

Студент

Андрей Филатов
Науки о данных
Апрель 24

Научный руководитель:

Иван Валерьевич Оселедец
Профессор

Москва 2023

Text interfaces for event sequences

Andrei Filatov

Submitted to the Skolkovo Institute of Science and Technology
on April 24

Abstract

In recent years, large language models (LLMs) have made significant progress in natural language processing, and it has been observed that these models may exhibit reasoning abilities when they are sufficiently large. One piece of evidence supporting this claim is the ability of LLMs to solve a wide variety of tasks, including making summaries, answering questions, sentiment classification, and more. This ability of LLMs has been further explored in prompt engineering, where a LLM can learn to solve new tasks in a zero- or few-shot format by directly typing a question and receiving the answer in text form. In this work, we introduce text interfaces for event sequence data. Text interfaces are interfaces that enable interaction between a user and a model using textual input and output. Through text interfaces, we were able to extract information from sequence data and use this data to solve different tasks. One benefit of this approach is that it is possible to solve multiple tasks simultaneously and perform a new task without additional training. The results produced by the text interface model are compatible with the best classical learning format models, which demonstrates its efficiency.

Keywords: prompt engineering, sequential data, large language models

Research Advisor:
Name: Ivan V. Oseledets
Degree: Dr
Title: Full Professor

Текстовые интерфейсы для последовательности событий

Андрей Филатов

Представлено в Сколковский институт науки и технологий
Апрель 24

Реферат

В последние годы языковые модели достигли значительного прогресса в обработке естественного языка, и было замечено, что эти модели могут проявлять способности к рассуждениям, если они достаточно велики. Одним из доказательств этого утверждения является способность языковых моделей решать широкий спектр задач, включая составление саммари, ответы на вопросы, классификацию настроений текста и многое другое. Эта способность языковых моделей получила дальнейшее развитие в промпт-инжиниринг, где языковая модель может научиться решать новые задачи без обучения в текстовой форме. В данной работе мы представляем текстовые интерфейсы для данных последовательности событий. Текстовые интерфейсы - это интерфейсы, которые обеспечивают взаимодействие между пользователем и моделью с помощью текстового ввода и вывода. С помощью текстовых интерфейсов мы смогли извлечь информацию из данных о последовательности событий и использовать эти данные для решения различных задач. Одним из преимуществ такого подхода является возможность одновременного решения нескольких задач и выполнения новой задачи без дополнительного обучения. Результаты, полученные моделью текстового интерфейса, сопоставимы с лучшими моделями классического формата обучения, что свидетельствует о ее эффективности.

Ключевые слова: языковые модели, последовательности данных

Научный руководитель:
Имя: Иван Валерьевич Оселедец
Ученое звание, степень: д.ф.-м.н.
Должность: Профессор

Contents

Chapter 1

Introduction

1.1 Event sequences

Event sequences [38] refer to a series of events that occur in a particular order, often with some sort of causal or temporal relationship between them. Event sequences can be found in various fields including computer science, natural language processing, and linguistics. In the context of computer science, event sequences are commonly used in the analysis of log data. Log data can contain a large amount of information about system behavior, user actions, and other events. By analyzing event sequences within this log data, it is possible to gain insights into patterns of behavior, identify potential issues or errors, and optimize system performance. Examples of such type of data are log entries, Internet of things telemetry, industrial maintenance, user behavior, travel patterns, medical history, transactional data, click through rates and other industrial and financial event sequences [2]. One technique for analyzing event sequences is machine learning [27]. Machine learning algorithms can be trained on large datasets of event sequences to identify patterns and make predictions about future events. For example, machine learning algorithms can be used to predict which products a customer is likely to purchase based on their past purchasing behavior. Event sequences are a powerful tool for analyzing behavior and understanding patterns in a wide range of fields. By analyzing event sequences, it is possible to gain insights into system behavior, language usage, and other complex phenomena.

1.2 Text as a universal task layer

Text is a universal task layer that serves as a fundamental building block for human communication and information exchange. From the earliest forms of writing to the latest digital technologies, text has been an essential tool for organizing and conveying information in a wide range of contexts, including education, business, science, and entertainment. At its core, text is a means of encoding information into a structured format that can be easily shared and understood. Whether it's a simple message written on a piece of paper or a complex scientific report, text provides a universal framework for organizing and transmitting information across diverse audiences and platforms. One of the key benefits of text is its accessibility. Unlike other forms of communication, such as spoken

language or visual media, text can be easily translated and shared across cultures and languages. This makes text an ideal tool for promoting cross-cultural understanding and collaboration, particularly in globalized contexts where diverse perspectives and communication styles are essential. Text is also a highly versatile tool that can be adapted to a wide range of tasks and contexts. From writing emails and reports to programming and data analysis, text provides a powerful medium for organizing and manipulating information in a structured and meaningful way. Whether it's a simple list or a complex database, text can be used to represent information in a way that is easy to understand and manipulate in a form of text interfaces.

1.3 Text interfaces

Text interfaces are a type of user interface that allows users to interact with computer systems through text commands. These interfaces have been used since the early days of computing and continue to be popular today due to their speed, efficiency, and flexibility. At their most basic level, text interfaces consist of a prompt, a cursor, and a text field where users can enter commands. Once a command is entered, the system processes it and returns a text-based response. This response can include information, feedback, or an error message, depending on the command and the system's capabilities. Text interfaces can be found in a variety of applications, including operating systems, programming environments, and software development tools. They are particularly popular among developers and power users who value the speed and precision of text-based commands. Some examples of text interfaces include the Windows Command Prompt, the Unix shell, and the Python interpreter. In recent years, there has been a renewed interest in text interfaces as more developers and users seek out lightweight and flexible tools. Many popular software applications, including code editors and development environments, now include built-in text interfaces that allow users to perform advanced tasks and automate repetitive workflows. However, text interfaces have traditionally been limited in their ability to understand and respond to human language in a natural way. Large language models [6] have changed this by enabling computers to understand and generate human-like text. These models are trained on massive amounts of data and can learn to generate text that is indistinguishable from human writing. This has made it possible to create text interfaces that can understand and respond to natural language queries in a more natural and human-like way.

1.3.1 Large language models as text interfaces

One of the most popular applications of large language models in text interfaces is in the creation of chatbots. Chatbots are computer programs that can simulate conversation with users through text or voice. They are often used by businesses and organizations to automate customer service

and support functions. Chatbots can answer frequently asked questions, provide information about products and services, and even help users troubleshoot technical issues. Large language models have made chatbots more effective by enabling them to understand and respond to user queries in a more natural and human-like way. This has led to an increase in user engagement and satisfaction with chatbot interactions. Chatbots powered by large language models can also learn from previous interactions with users and improve their responses over time. Another application of large language models in text interfaces is in virtual assistants. Virtual assistants are computer programs that can understand and respond to voice commands. They are often used to perform tasks such as setting reminders, making appointments, and controlling smart home devices. Large language models have made virtual assistants more effective by enabling them to understand and respond to natural language voice commands in a more human-like way. This has led to an increase in the adoption of virtual assistants and their use in a wide range of applications.

1.4 Application of text interfaces

The application of text interfaces doesn't stop on text domain - Visual Language Models (VLMs) has become increasingly popular in recent years, as it offers a more accessible way for users to interact with and control these models. This technology has numerous applications across a range of industries, including entertainment, gaming, marketing, and education. One of the primary benefits of text interfaces for visual language models is that they allow users to communicate complex ideas and concepts in a more natural and intuitive way. For example, in "Segment Anything" paper [23] authors proposed to use text for defining segmentation masks. Another example is Imagen work [32] where generation of images from text. This is particularly important for fields such as design, where visual communication is a crucial part of the creative process. By using a text interface to control a visual language model, designers can easily experiment with different visual elements and layouts, without having to spend hours manually creating and tweaking each individual design. For example, using VLMs it is possible to blend text and image or is is possible to edit image using text [15]. Another key benefit of text interfaces for visual language models is that they allow users to quickly and easily generate large amounts of content. This is particularly useful for marketers, who often need to create multiple versions of the same message for different audiences and platforms. By using a text interface to generate visual content, marketers can easily create and customize images and videos to fit their specific needs, without having to spend time and resources on manual design work.

1.5 Aim of research

The success of text interfaces on text and visual domain motivates us to explore possibilities of constructing text interfaces for different modalities. The main purpose of the research is an investigation of possibility to create text interfaces for a new modality. As a new modality we have chosen sequential data. This data is truely multimodal because is contains from different source and having different nature. Also, this is variable length data, which allows to investigate how text interfaces can work on other sequential modalities besides text. Our research has a several subgoals:

- **Creation proper representation for event sequence**. To pass data to text interfaces we first should preprocess it. So, we have to conctruct encoding model and conduct experiments to figure out which representation works the best, because the better representation will provide the better results.
- **Construction of question-answering format task on sequential data**. To get answers from text interface we should pass the text + embedding input. So, we should conduct experiments on how to transform task in a question.
- **Conducting experiments in several tasks**. The unique ability of text interfaces is an ability to solve many tasks simultaneously. So, we are interested in conducting experiments on possibilities of text interfaces to solve many tasks on sequence data.

1.6 Novelty of project

To the best of our knowledge it is the first work which creates text interfaces on sequential data. Also, it is the first work which creates a lot of tasks from sequential data and represent it in question-answering.

Chapter 2

Related works

2.1 Event sequence processing

There are several ways to construct event sequence in unsupervised manner representation over event sequence. The first is to apply masked language modelling [12] some of events in masked and the model is learning to reconstruct the missing tokens. In this task some percentage of tokens is masked and the task is to reconstruct them. The second task is replace token detection [10]. The essence of this task is replacing some events with random one. The purpose of the model is to detect which token was replaced. This task commonly considered more efficient because the gradient passes through all events not only masked one. Also, there are Contrastive Learning for event sequences is over method where contrastive learning is applied. The speciality of this method is how positive examples constructed. Positive examples is constructed using but splitting the sequence on several subsequences. The motivation is that the representation of the object doesn't change from time. That's why the embedding of different subsequence should the same. The full survey can be found in [31]. The main thought of this paper is that whether you use different pretraining scheme for better performance on final task one shoulde use targets information on pretraining stage.

The first thing is necessary to process the data is to transform data in numerical values. To transform categorical features classical approach is to apply one-hot encoding where category transform to vector representing ones on all places and zeros on other. In the case of huge amount of features this approach is not representive, because number of features is increase linearly with number of features. The other approach is target encoding. In this approach categorical features is replaced by probabilities of the target class. The most popular and universal approach is embedding approach. Each categorical value is mapped into some learnable embedding and passed further in model. For numerical values there several approaches to process them. The first is to apply logarithmic transformation, normalize them or apply clipping of values. As researchers found out that transformers perform poor on numerical data they propose to apply discretization and transform numerical features into categorical features. This approach harmonically treats outliers and properly works with numbers. But this approach has a limitation, by discretization we reduce ordering between numbers. In work [17] ordering preserved by introducing piecewise linear encoding, where

an embedding for a new value is constructed by summing embeddings for all previous values. Also, in this paper they propose a new scheme for discretization. In classical discretization approach bins created by using quantiles of the numerical features. In [17] they propose constructing bins using tree. This approach induce target information in embeddings and improve the quality of algorithms trained using this type of embeddings.

When, the representation for each feature is constructed it is necessary to combine them. The standard scheme is to concatenate them. But it can harm performance because the inter-feature relatioships in model aren't included into consideration. In work [20] embeddings of categorical features additionally processed with transformers because context is meaningful. This architectures was modified in the work [7] where multilayer perceptrons were replaced with gating multilayer perpeptron

The same trick is applied in [28] there feature embedding is processed and only then concatenated in transaction embedding.

There are different architecture which can process event sequences. In early recurrent neural networks were used to process event sequences. For example in the work [3] recurrent neural networks were applied to. Another work is [24]. But recently, with the rise of transformer models [37] they become a standard for sequence processing and event sequence is not exception In [18] author adapted ResNets and Multilayer perceptrons. Also, they have shown that ResNet are better imitate boosting trees than MLP. The close topic for event sequence is time series. They are event sequences too but have less data only value. The main problem of applying transformers to event sequence is that attention scales quadratically with number of inputs which in case of long sequence become big problem. The one idea is to process data sequentially by splitting data on accessible part of data, but in case of very long sequences doesn't work because model forget the context of previous windows. The trick to solve this problem proposed in [11]. They reuse intermediate outputs of previous model in order to keep information from previous stages. The next idea is to use sparse attention mechanism. In this case complexity of attention isn't $O(n^2)$ but $O(n \log n)$ or even $O(n)$. For linear attention it is proposed mechanism of linear attention. In this situation the linear attention is splitted onto scaled product, so the complexity of this attention is linear [22]. The other technique usually choose only some elements with each the attention is calculated. The example of such works are Informer [41]. In [42] the authors were able to reduce complexity to linear by random sampling constant number of Fourier frequencies.

2.2 Large language models

Large language models is arised in natural language processing and become de-facto a standard. They trained on large text corpora in unsupervised manner using Masked Language Modelling,

Causal Language modelling, Prefix Language modeling. Through this task models are able to solve variety tasks. The example of such models are GPT-3[6], BART [25], T5 [30], UL2 [36], PaLM [8], BERT [12]. They are shown incredible possibilities for solving task such reasoning, semantic parsing, sentence classification, text generation, question answering. Applied to event sequence Language models are shown good performance on data to text tasks. In [34] considered survey of different approaches. In [29] language models are applied to transform table to text. In [**?**] is shown that language models are able to solve tabular problems better than logistic regression, gradient boosting. In [4] done the same. In [5] is shown that language modelling can also generate realistic table. The main feature of this type of model that they easily be adapted to conditional generation and realistically imitate misssing data in the table. In [21] language modelling are used to to solve semantic reasoning problem. The idea of semantic reasoning is transform natural language query into structured query language which can be executed. In [21, 39] language model are efficiently could solve this task. In [13] semantic parsing was applied to simplify the original task. In [14] create joint model which can map data to text and vice versa.

2.3 Prompt engineering

The basement for text interfaces is given by prompt engineering technique. Creation of large language model takes a lot of time. So, finetuning on a new task will also take a lot of time. But, large language models are good in seeing patterns and working with text. In [6] it has been noticed that big model are able to solve tasks in a zero-/few-shot manner. So this concept will become interesting because it was doesn't necessary to learn model but just correctly construct prompts. The survey of prompt engineering technique is given in [26]. The idea of prompt engineering is the following large language models are so powerful so by correctly constructing the request the model will be to answer it. So, the request is constructed in the following form: [Input] text [Output]. And the purpose of the model is to correctly fill the output. There are some features on how to map the output of model in label space.

There are several techniques for prompt engineering on how much model is trained. First, is classical approach where pretrained language model is just finetuned onto downstream task. The second approach is Tuning-free Prompting. In this setting no parameters are trained and template is manually selected. The third approach is Fixed-LM Prompt Tuning. In this approach language model is frozen but additional training prompts are added to the model. The fourth approach is Fixed-prompt LM Tuning. In this approach the fixed prompt is chosen and language model is trained to solve downstream task. The last approach is Prompt+LM Fine-tuning. In this approach both language model and prompt parameters are trainable. It allows to achieve better performance, but tend to overfit and may do less impact.

2.4 General purposes interfaces

The concept of text interfaces is text domain. The majority of text task have both question and answers in the form of text. In [30]. [9] the model T5 were able to solve more than 2 thousands task on comparable. Next the concept of interfaces were developed in multimodal text, image domain. There object detection, image reasoning will be able to solved using text. In the work [19] the model propose to use the text interface for image and audio modalities. In the [40] the authors propose to combine different pretrained models in order to solve task in zero-shot manner using templates. In the work [1] authors were able to combined freezeed language model with visual encoder and able to solve tasks both in image and text domain. Also, in the work GPT-4 also has possibilities for multi-modal dialog. [35]. In [33] large language models were applied as application programming interfaces where the they were able to send emails or call calculator and solve mathematical tasks.

The similar approach were proposed in [16] where text-interfaces are applied to recommendation task. In this domain, model can provide recommendation on the basis of dialogue with the user, provide summary of reviews, make sequential recommendation, create reasoning for model prediction, ask does the user like the recommended film, or recommend film from the list, predict the rating of the model. Also, this model is able to solve task in zero-/few-shot manner.

There are also works which applied quaestion answering for graph data. Graph data. Also, there are tries to combine data, text and vision modalities. These type of models are possible.

Chapter 3

Methodology

3.1 Text interfaces

To understand in what there is the difference between large language model

Text interfaces as large language model requires large corpora for training. Additionally, text interfaces require a data. For you best knowledge, there are no joint dataset of text and sequential data in question-answering format. Also, there lack of datasets in multi-task format. So, we have to handle both problems. How we did will be described in the following sections. First, we discuss how to create multiple tasks on sequential data. Then, we discuss different form of questions of dataset.

$$\mathcal{L}_{AR} = -\sum_{i=1}^{n} \log p(\mathbf{y}_i|\mathbf{y}_{<i}, \mathbf{x}; \theta)$$

,

where $x_{<i}$ denotes the sequence of words preceding the i-th word, $p(x_i|x_{<i}; \theta)$ is the probability assigned by the model to the i-th word given the preceding words, and θ are the model parameters.

3.2 Task generation

For all experiments we use AlfaBattle 2.0 dataset. AlfaBattle2.0 is a large-scale dataset of anonymous 450M transactions for one and a half million users. For every user we have credit default label, whethere a client has default or not. We consider to possible type of tasks which can be generated — predictive and contextual tasks. Predictive tasks which utilize the data to predict the future - default, next item or agreggation of metric on next items. These tasks are very informative and used in different domains. Contextual tasks from the other side intended to learn model to solve tasks which are extracted from data: what is the most frequent feature value or the median value of the feature?. These task forces the model to learn representation very well in order to extract feature values.

3.2.1 Predictive tasks

For predictive tasks we propose the following tasks. First is to reuse the original task - default. Second - create task of prediction of next feature value. This task allows model to better predict the quality of metric. Third is to create task which aggregates feature values of future events and task is to predict this aggregate values. For example, prediction of number of transactions in the next seven days, or prediction of sum of all transactions in the next 30 days.

3.2.2 Contextual tasks

Contextual tasks are the tasks which targets could be directly extracted from the data. For example, the average time difference between transactions. For AlfaBattle we construct the following tasks. What is the most frequent type of transaction and what is the mean amount of transactions of task.

3.2.3 Question

To text interface work with data we have to reformulate the task in text form. For this, we consider formulating task in the form of question and prediction of the model in the form of text answer.

To create text answer we prepend ”This is the transaction history of the client”. We introduce to new tokens ”<trx>” and ”</trx>” to describe the beginning of transaction history and end of history correspondingly. Then, we create a special intro for every questions

There are three types of questions

- Binary. In this setting additional to the question we pass ”Yes or No?” to force model choose between this two options. As the output of the model we wait ”Yes” or ”No” words?
- Multiple choice. In multiple choice setting we pass ”OPTIONS”: and several variants. the model should choose between proposed answers. It is possible that there are no correct answers between proposed ones, so the model in the case should produce the answer ”Neither”.
- Open-ended. In this setting we ask model just a question without any hints and answer can be any form. The answer can be in any form, number, word, sentence, etc.

The construction of the question depends on the type of task. If task is numeric there is one way to construct the task. If a task in categorical, then there is another way of construction.

For binary numeric task we considered the following way of task construction. We construct binary task as prediction whether the value will be more than some threshold value. E.g, amount > 10k or number of events > 4. It is important to choose threshold properly in order not to get disbalanced task. So, for construction of threshold we use the following scheme. On train dataset

we calculate median of all possible values. Then we use this median value as a threshold. It allows to get balanced task which allows more properly assess the quality of the model

For binary categorical tasks we considered the following way of task construction. Categorical value can have limited set of values, so we decided to use subset of possible values and construct task as a check whether we predicted lying in chosen set. We chose this set to randomly to get balanced task. For the same reason as in previous task.

For numerical multiple choice task we should provide several options. But text model are bad with predicting numerical values, so we decided to apply the following scheme. We apply discretization procedure to get from numerical feature a possible intervals which model can predict. For example, we can split numerical feature on intervals: [0, 1000], [1000, 2000], [2000, 3000]. In this case the task from predicting exact number simplifies to predicting interval in which number is lying. We conducted this procedure with our numerical features. To get balanced interval in terms of number of values lying in it we chose quantiles as borders of intervals. This allows to get even predictions.

For categorical multiple choice task it's much easier to construct the number of options. So the idea is the following: sample the some classes and check if answer correct or there is no answer between this features.

For open-ended task the task is constructed by simply checking whether the predicted value is indeed equal to target value. This requires some sort of preprocessing, about it we will tell in the next section.

3.2.4 Mapping answer to label

As the answer of our model is just text to get predicted labels we should process text output of the model. Binary tasks is simple case. Here we need get only positive or negative answer. This can be achieved by using some positive words: Yes, True, Positive, etc.; and negative words: No, False, Negative, etc.; We can chose the answer as word with higher probability.

For multiple choice we have predefined set of variants so we can calculate probabilities of all varients and as predicted label choose the option with the highest probability.

3.2.5 Prompt Augmentation

To construct task in question-answering format we have to create a question. But only one question can be not enough - model can overfit for a certain pattern and on semantically similar question but syntatically different can provide random prediction. Also, using several forms of questions can use as an ensembling to get more diverse prediction. So, for all our experiments we use several forms of questions.

3.3 Creating representation for sequential data

To pass sequential data to embedding we should pass it to text interfaces. The simple idea is to use transform data to text format. E.g., ”the event #1 happened in 12 am 21 December 2022 with the following feature values, the event #2 happened i2 13 am 21 December 2022”. But this representation is too sparse and also to long - modern architectures as a transformer can’t handle sequence of thousands events. So, we propose to use embedding formulation. The idea is to transform sequence of events in to embedding representation. It can one-to-one transformation where for each event we provide independent embedding. Or it can be mapping all sequence in one or several embeddings. Both options are possible but in our research we focus on one-to-one mapping. To construct the embedding of event we use the following scheme. For numerical features we map it value to embedding with higher dimension. For categorical features we map it to embeddings. For sequence feature - features which occurs only once we map them to embedding and concatenate to embeddings of all events. Base embedding of the sequence is made of concatenation of embeddings numerical features, categorical features, sequence features. The dimension of embeddings are empirically chosen. But this representation is very restricted because it doesn’t count relationship between the features. 10 years dog is far from 2 month dog. So, we have to manage to consider relationship between the features. To consider we propose to apply the model on top of the embeddings which was pretrained on relevant tasks and as input for text interface model use these processed embeddings.

To conduct how to build the universal representation for sequence data we conduct series of experiment which will be described further.

3.3.1 Metrics

AUC

The AUC (Area Under the Curve) formula is commonly used in machine learning to evaluate the performance of binary classification models. It measures the area under the Receiver Operating Characteristic (ROC) curve, which is a graphical representation of the trade-off between the true positive rate and the false positive rate at different classification thresholds.

The AUC formula is defined as follows:

where TPR is the true positive rate, FPR is the false positive rate, and is the inverse function of the false positive rate.

In practice, the AUC value ranges from 0 to 1, with a higher value indicating better performance of the classification model.

Accuracy

The formula for Accuracy is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP represents the number of true positives, TN represents the number of true negatives, FP represents the number of false positives, and FN represents the number of false negatives.

In other words, Accuracy is the ratio of correctly predicted data points to the total number of data points. It gives us an idea about the overall performance of a model.

Chapter 4

Numerical experiments

4.1 Data

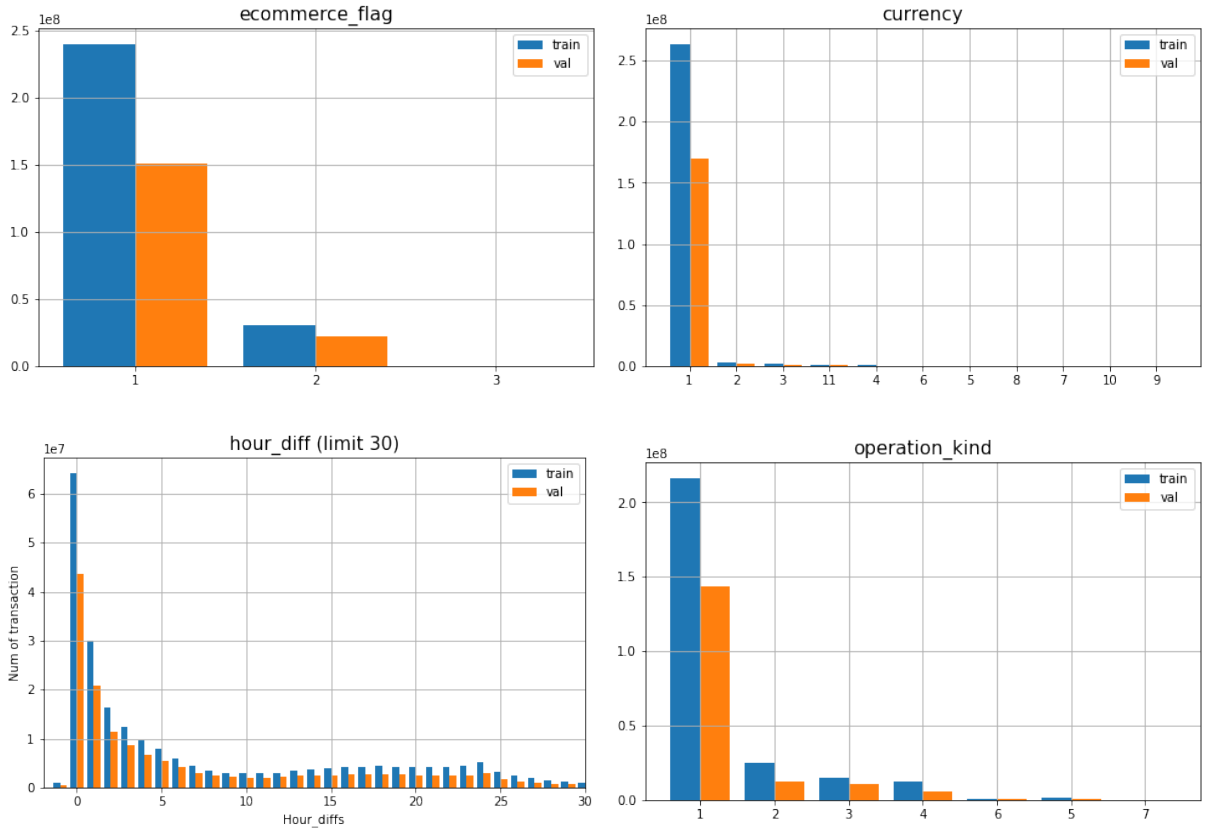
For training we use AlfaBattle dataset. AlfaBattle is a dataset of 400 million transaction of one and a half million users. The feature available for every transaction is presented in Table 4.1. Additionally, we data contains a target — whether a client will have default or not. The initial task of this dataset is to predict whether the client will have a default by using transaction history of clients

Feature name	**Description**	**Unique values**
currency	Transaction currency identifier	11
operation_kind	Transaction type identifier	7
card_type	Unique card type identifier	175
operation_type	Transaction type identifier	22
operation_type_group	Card transaction group identifier	4
ecommerce_flag	A sign of e-commerce	3
payment_system	Payment system type identifier	7
income_flag	Indication of debit/ deposit of funds	3
mcc	Unique point of sale type identifier	108
country	Transaction country identifier	24
city	Transaction city identifier	163
mcc_category	Transaction category identifier	28
day_of_week	Day of the week the transaction was made	7
hour	The hour when the transaction was made	24
weekofyear	The week of the year in which the transaction took place	53
days_before	Number of days before the loan	23
hour_diff	Number of hours since the last transaction	10
amnt	Normalised transaction amount	inf

Table 4.1: Decsription of features on AlfaBattle dataset

4.2 Explorative data analysis

On the figures one can see distribution of the features. Most of them a categorical and contains 16 categorical features and 3 numerical features.



4.3 Model

As the model for text interfaces we chose pretrained family of FLAN-T5 models [9]. T5 model is encoder-decoder model. We chose this model because it contains both benefits of context model as it have encoder and generative model as output is generated by decoder. As we want to train text interface to solve dozen of tasks it is necessary to model have good pretraining. Flan-T5 is a familty of language model which was finetuned on more than two thousands tasks and demonstrated significant gain in quality. So, adding new tasks for finetuning is important for our model.

There are three version of model which we utilize in our experiments: small, base, and large. The small version has 60 million parameters, the base version has 220 million parameters, and the large version has 770 million parameters.

4.4 Representation for event sequences

Constructing representation for sequential data requires both choice of pretraining scheme and model architectures. So we conducted experiment on both to figure out which representation is the best.

To construct universal distribution we conduct three experiments:

First, we investigate does it possible to reuse different text representation models for . To

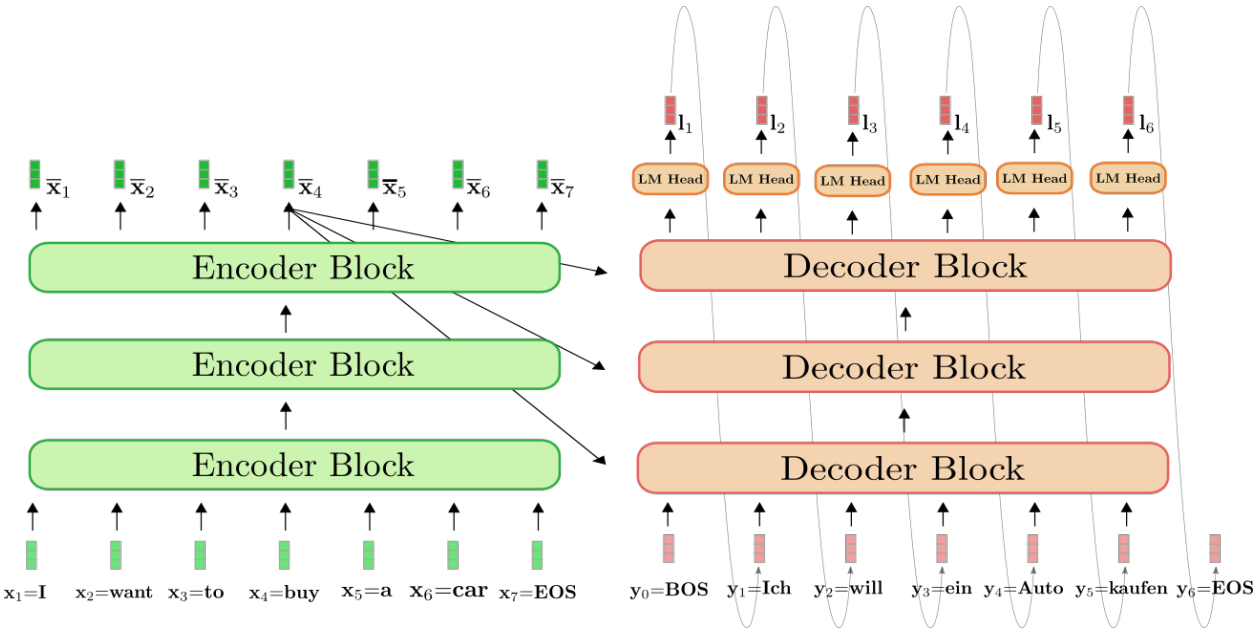


Figure 4.1: Caption

	Params	Learnable	Scratch	Finetune	Delta with SOTA
GPT2 Base	124M	0.12%	0.7869	0.7745	-1%
GPT2 Medium	355M	0.07%	0.7833	0.7798	-2%
GPT2 Large	774M	0.06%	0.7747	0.7855	-1%
BERT Base	110M	0.06%	Disconverge	0.769	-3%
BERT Large	335M	0.06%	Disconverge	0.7652	-3%
T5 Small	60M	0.18%	0.7721	0.7673	-3%
T5 Base	223M	0.07%	0.7756	0.7731	-3%

do this we conduct we take different pretrained architectures and finetune them on task from our benchmark. We considered several architectures. Bert (base and large), T5 (base, small, large) and GPT-2 (base, large).

4.5 Predictive tasks

For AlfaBattle dataset we were able to construct 6 different tasks

- Sequence classification
  - Default prediction
- Next feature prediction
  - Next amount prediction
  - Next merchant category code prediction
  - Next

Model \Task	Hour	Mcc	Amnt	Default*	Transactions-30	Amount-30
classification model* (27 M. trainable params.)	0.71	**0.762**	**0.713**	**0.791**	0.980	**0.965**
flan-t5-small (106 M. trainable params.)	0.657	0.736	0.683	0.762	0.970	0.945
flan-t5-base (295 M. trainable params.)	0.678	0.753	0.692	0.775	0.975	0.956
flan-t5-large (813 M. trainable params.)	0.691	0.761	0.710	0.784	**0.981**	0.964

- Next aggregation prediction
  - Number of transactions in the next 30 days
  - Aggregated amount for transactions in the next 30 days.

In the next table one can see the results of it. For all tasks, except default we report the accuracy score. For default we report area under curve, because it's very imbalanced tasks only 3% of objects have label 1, other 97% of users have.

Here we can see the growth in quality so we decided to examine how fast the model grows.

Chapter 5

Discussion and conclusion

In this work we constructed text interfaces for sequential data.

Summarize your work in this section.

1. Summary of the main results of the work that is consistent with the Aim and Objectives.
2. Overall position on the global research landscape.
3. Comparative critical analysis: what you have deduced from the findings and how these results relate to previous research or other studies.
4. Research limitations.

Acknowledgements

Thank you:

- Denis Dimitrov and Andrey Kuznetsov for suggesting idea of the project
- Elizaveta Goncharova for reviewing my work
- Maxim Zubkov for providing guidance on tokenization and model choice
- Irina Abdullaeva for helping with experiments

Bibliography

[1] Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198* (2022).

[2] Babaev, D., Ovsov, N., Kireev, I., Ivanova, M., Gusev, G., Nazarov, I., and Tuzhilin, A. Coles: Contrastive learning for event sequences with self-supervision. In *Proceedings of the 2022 International Conference on Management of Data* (2022), pp. 1190–1199.

[3] Babaev, D., Savchenko, M., Tuzhilin, A., and Umerenkov, D. Et-rnn: Applying deep learning to credit loan applications. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (2019), pp. 2183–2190.

[4] Bertsimas, D., Carballo, K. V., Ma, Y., Na, L., Boussioux, L., Zeng, C., Soenksen, L. R., and Fuentes, I. Tabtext: a systematic approach to aggregate knowledge across tabular data structures. *arXiv preprint arXiv:2206.10381* (2022).

[5] Borisov, V., Seßler, K., Leemann, T., Pawelczyk, M., and Kasneci, G. Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280* (2022).

[6] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems 33* (2020), 1877–1901.

[7] Cholakov, R., and Kolev, T. The gatedtabtransformer. an enhanced deep learning architecture for tabular modeling. *arXiv preprint arXiv:2201.00199* (2022).

[8] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[9] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).

[10] Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555* (2020).

[11] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860* (2019).

[12] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Drozdov, A., Schärli, N., Akyürek, E., Scales, N., Song, X., Chen, X., Bousquet, O., and Zhou, D. Compositional semantic parsing with large language models. *arXiv preprint arXiv:2209.15003* (2022).

[14] Duong, S., Lumbreras, A., Gartrell, M., and Gallinari, P. Learning from multiple sources for data-to-text and text-to-data. *arXiv preprint arXiv:2302.11269* (2023).

[15] Gandikota, R., Materzynska, J., Fiotto-Kaufman, J., and Bau, D. Erasing concepts from diffusion models. *arXiv preprint arXiv:2303.07345* (2023).

[16] Geng, S., Liu, S., Fu, Z., Ge, Y., and Zhang, Y. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems* (2022), pp. 299–315.

[17] Gorishniy, Y., Rubachev, I., and Babenko, A. On embeddings for numerical features in tabular deep learning. *arXiv preprint arXiv:2203.05556* (2022).

[18] Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems 34* (2021), 18932–18943.

[19] Hao, Y., Song, H., Dong, L., Huang, S., Chi, Z., Wang, W., Ma, S., and Wei, F. Language models are general-purpose interfaces. *arXiv preprint arXiv:2206.06336* (2022).

[20] Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678* (2020).

[21] Iida, H., Thai, D., Manjunatha, V., and Iyyer, M. Tabbie: Pretrained representations of tabular data. *arXiv preprint arXiv:2105.02584* (2021).

[22] Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning* (2020), PMLR, pp. 5156–5165.

[23] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. Segment anything. *arXiv preprint arXiv:2304.02643* (2023).

[24] Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval* (2018), pp. 95–104.

[25] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[26] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys 55*, 9 (2023), 1–35.

[27] Mitchell, T. M., et al. *Machine learning*, vol. 1. McGraw-hill New York, 2007.

[28] Padhi, I., Schiff, Y., Melnyk, I., Rigotti, M., Mroueh, Y., Dognin, P., Ross, J., Nair, R., and Altman, E. Tabular transformers for modeling multivariate time series. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2021), IEEE, pp. 3565–3569.

[29] Parikh, A. P., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D., and Das, D. Totto: A controlled table-to-text generation dataset. *arXiv preprint arXiv:2004.14373* (2020).

[30] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research 21*, 1 (2020), 5485–5551.

[31] Rubachev, I., Alekberov, A., Gorishniy, Y., and Babenko, A. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208* (2022).

[32] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems 35* (2022), 36479–36494.

[33] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761* (2023).

[34] Sharma, M., Gogineni, A., and Ramakrishnan, N. Innovations in neural data-to-text generation. *arXiv preprint arXiv:2207.12571* (2022).

[35] Surís, D., Menon, S., and Vondrick, C. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128* (2023).

[36] Tay, Y., Dehghani, M., Tran, V. Q., Garcia, X., Wei, J., Wang, X., Chung, H. W., Bahri, D., Schuster, T., Zheng, H. S., et al. Ul2: Unifying language learning paradigms, 2022.

[37] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[38] Yeshchenko, A., and Mendling, J. A survey of approaches for event sequence analysis and visualization using the esevis framework. *arXiv preprint arXiv:2202.07941* (2022).

[39] Yin, P., Neubig, G., Yih, W.-t., and Riedel, S. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314* (2020).

[40] Zeng, A., Wong, A., Welker, S., Choromanski, K., Tombari, F., Purohit, A., Ryoo, M., Sindhwani, V., Lee, J., Vanhoucke, V., et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598* (2022).

[41] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence* (2021), vol. 35, pp. 11106–11115.

[42] Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning* (2022), PMLR, pp. 27268–27286.