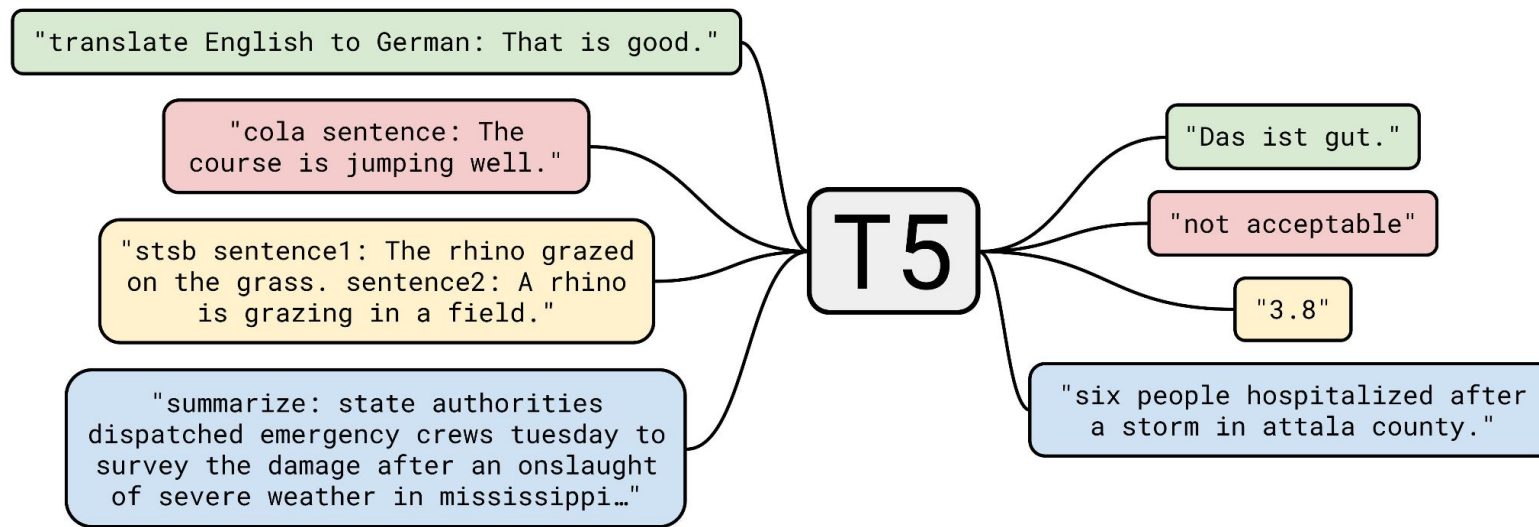


Text interfaces for sequential data

Student: *Andrei Filatov*
Research Advisor: *Ivan Oseledets*

Text interfaces

Text is an universal task layer, because it allows solve any task which have input in form of text and output in form of text

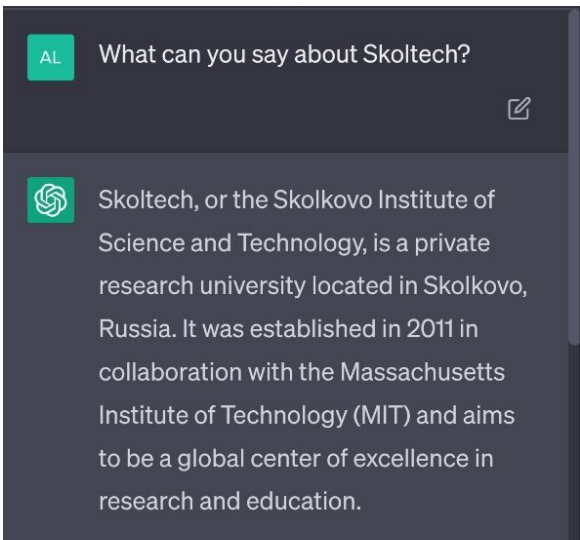


Picture from "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer"

Text interfaces

Through text we can construct text interfaces:

- Chatbots
- Visual-textual dialogues
- Multi-modal dialogues



Video Search

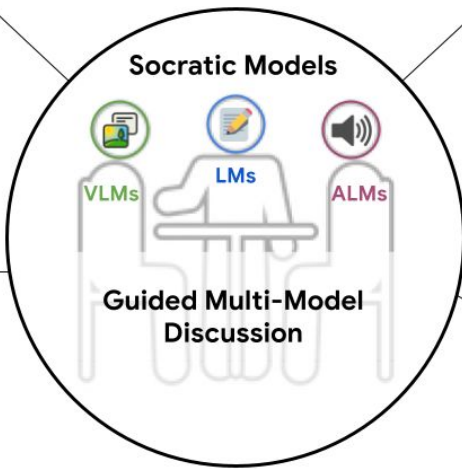
Q: Where did I leave my mug?



Image Captioning



Summary: I am watching netflix in a living room.

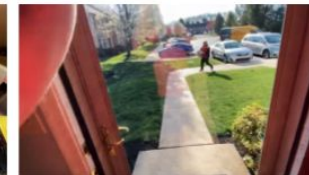


Free-form Video Q&A: Visual & Contextual Reasoning

Q: When did I wash my hands? Q: Why was I at the front porch? Q: Why was I chopping wood?



A: I last washed my hands at 3:38 PM.



A: I went to the front porch today to receive a package.



A: Because I needed to get a fire going in the fireplace.

Forecasting: Predicting Future Activities



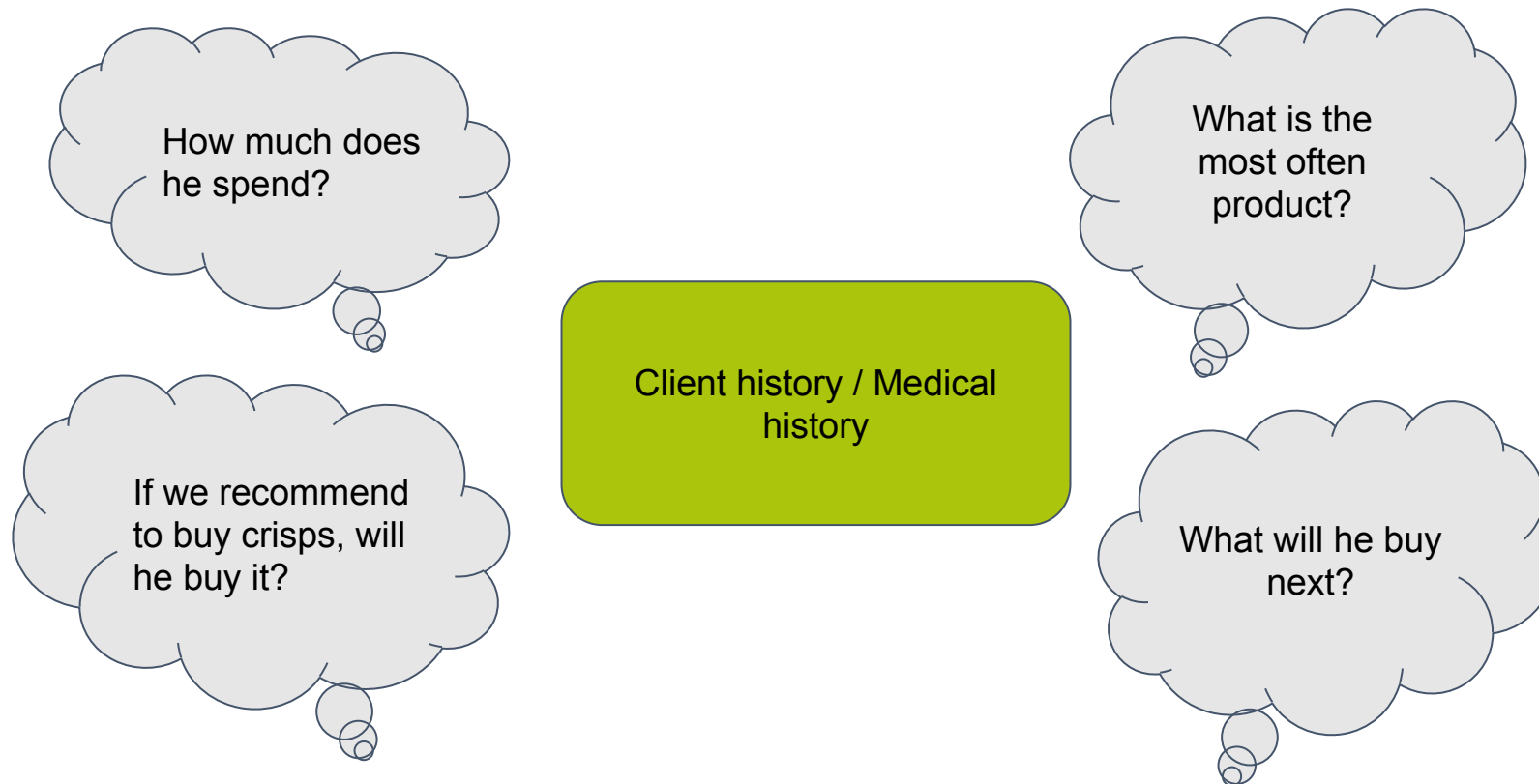
1:46 PM: I am eating a sandwich in a kitchen.
 2:18 PM: I am checking time and working on a laptop in a clean room.
 2:49 PM: I am buying produce from a grocery store or market.
 3:21 PM: I am driving a car.
 4:03 PM: I am in a park and see a playground.
 4:35 PM: I am in a home and see a television.

Text interfaces: advantages

- Allow to solve any number of tasks if they can be formulated in text form.
- Allow to construct multi-modal models
- Allow to do zero- / few-shot learning through in-context learning

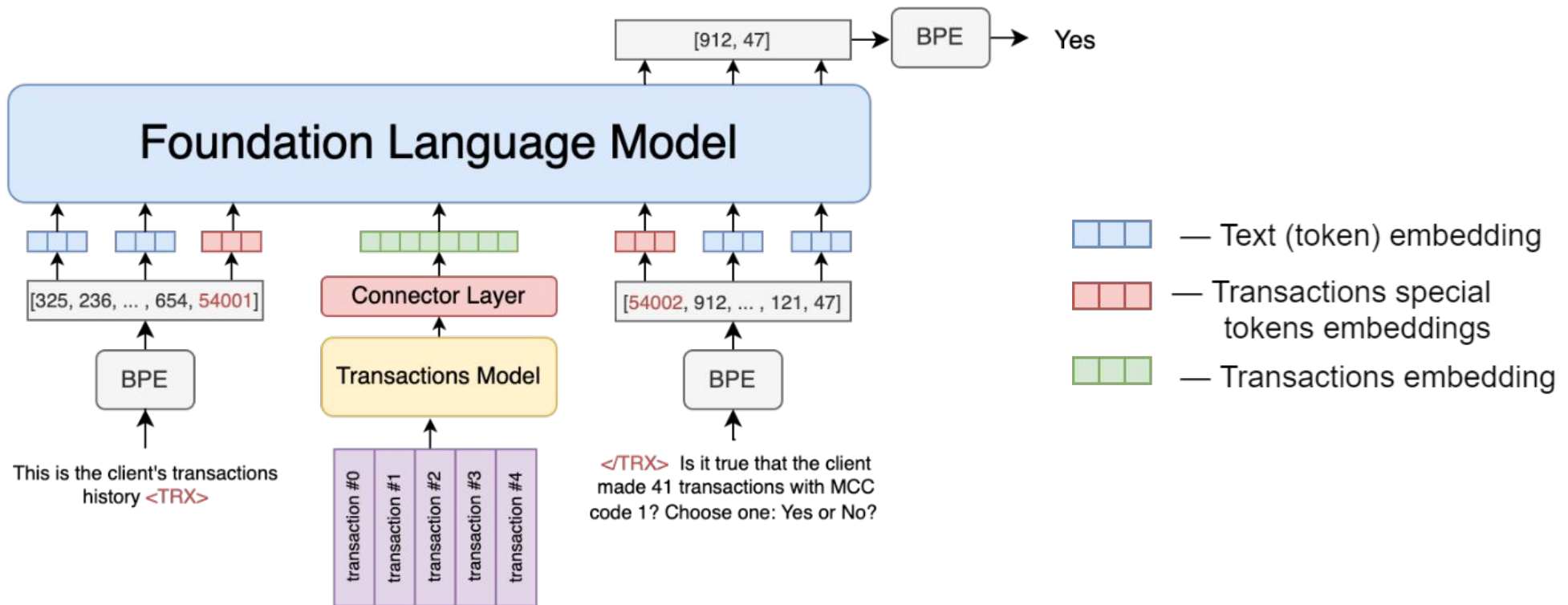
Text interfaces for sequential data

Aim: Construct of text interfaces for sequential data



Text interface: structure

We want to construct model of the following type:

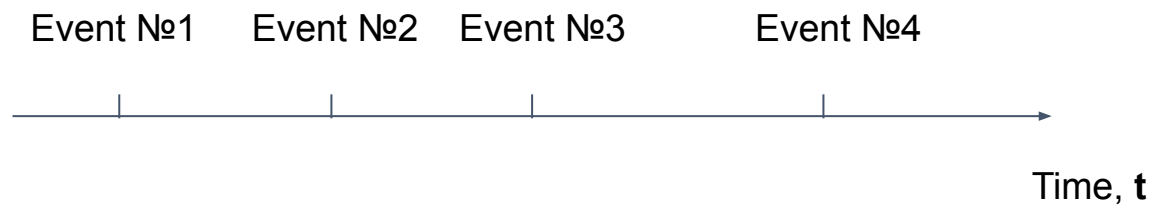


Objectives

1. Construction of representation over sequential data
2. Examine the basic text interface concept
3. Modifying text interfaces for solving several tasks simultaneously / multi-modality

Sequential data

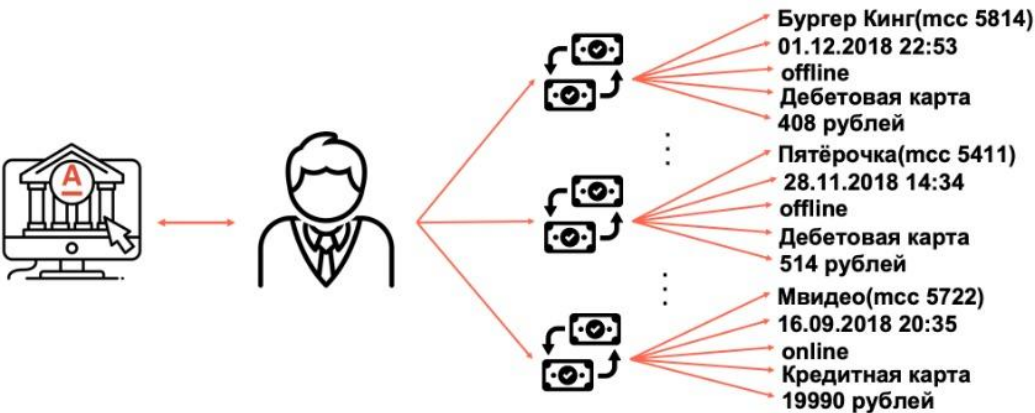
Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor	Home page, index page, contact info
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms



Dataset

AlfaBattle:

- **450 million bank** customer transactions, ~6 Gb.
- Transaction history depth up to **1 year**;
- **1.5 million** unique users;
- Transaction is a set of **19 attributes**;



	app_id	amnt	currency	operation_kind	card_type	operation_type
0	0	0.465425	1	4	98	4
1	0	0.000000	1	2	98	7
2	0	0.521152	1	2	98	3
3	0	0.356078	1	1	5	2
4	0	0.000000	1	2	98	7

Sample of transactional data

Representation model for sequential data

The representation model consists of 3 parts:

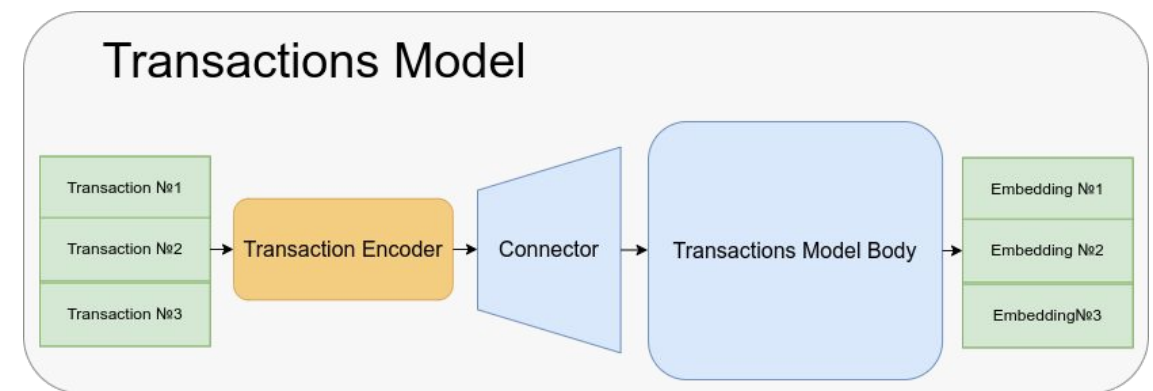
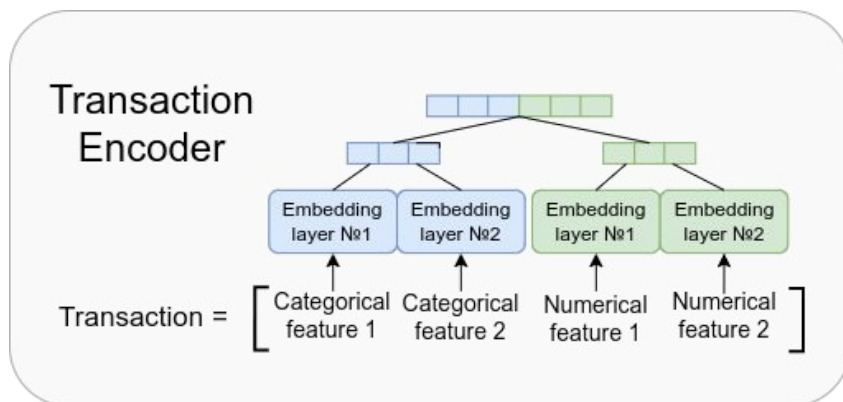
Transaction encoder: the block that converts transactions into embeddings.

Connector: the block that sets the dimensionality of the transaction's embedding to the dimensionality of the model body

Transactions Model Body: in our case it is a 4 x Transformer Decoder Layer.

Experiments:

- What transaction model architecture should we use? Encoder/Decoder/Encoder-Decoder?
- What pretraining method should we use?



Representation model: architecture

Motivation:

There is huge difference between how decoder/encoder architectures process a sequence:

- Decoder process sequence in natural way only looking at history.
- Encoder process sequence fully: mixing up the past and the future

Architectures:

- Decoder: GPT2 Base (124M), Medium (355M), Large (774M)
- Encoder: BERT Base (110M), Large (335M)
- Encoder-Decoder: T5 small (60M), T5 Base (223M)

Task / Metric:

- Default (AUC)

	Parameters	Scratch	Finetune (ALL)	Delta with SOTA
GPT2 Base	124M	0.7869	0.7745	-1%
GPT2 Medium	355M	0.7833	0.7798	-2%
GPT2 Large	774M	0.7747	0.7855	-1%
BERT Base	110M	Disconverge	0.769	-3%
BERT Large	335M	Disconverg	0.7652	-3%
T5 Small	60M	0.7721	0.7673	-3%
T5 Base	223M	0.7756	0.7731	-3%

Representation model: architecture

Motivation:

In the previous experiment we have found out that decoder architecture work the best. So, we decided to conduct experiment to decide what model size should choose and does transformer decoder outperform recurrent models?

Model:

- Transformer decoder: GPT2 base (100M), Whisper-tiny decoder (27M)
- RNNs: GRU small (0.3M), base (19M), ultra-big (39M)

Tasks / Metrics:

- Next item amount prediction (L1 loss)
- Next item time prediction (+- 12 hour interval hit accuracy)
- Next item MCC code prediction (accuracy)
- Various pair and triplets of tasks above. E.g., amount prediction in the case of next time + amount prediction task

Representation model: architecture

	Amnt	MCC	Time	Time 3 features	Amnt 3 features	MCC 3 features	Amnt Amnt + MCC	MCC Amnt + MCC	Amnt Amnt + Time	Time Amnt + Time	MCC Time + MCC	Time Time + MCC
Metric	L1 loss	Accuracy	Hit Accuracy	Hit Accuracy	L1 loss	Accuracy	L1 loss	Accuracy	L1 loss	Hit Accuracy	Accuracy	Hit Accuracy
Baseline	<u>0.0788</u>	<u>0.4075</u>	<u>0.5319</u>	<u>0.5319</u>	<u>0.0788</u>	<u>0.4075</u>	<u>0.0788</u>	<u>0.4075</u>	<u>0.0788</u>	<u>0.5319</u>	<u>0.4075</u>	<u>0.5319</u>
GPT2 Base (100M)	0.0632	0.4866	0.5986	0.5978	0.06667	0.48	0.6567	0.4888	0.069	0.6126	0.4812	0.5937
Whisper-tiny (37M)	0.0627	0.4801	0.5888	0.5947	0.0672	0.48	0.66	0.4861	0.068	0.5844	0.4812	0.593
GRU Small (0.3M)	0.0627	0.4842	0.5601	0.5616	0.677	0.47	0.67	0.4787	0.068	0.5656	0.4729	0.5588
GRU Base (19M)	0.0627	0.4808	0.5849	0.5728	0.6707	0.48	0.67	0.4805	0.067	0.5823	0.4805	0.5746
GRU-Ultra-big (35M)	0.0657	0.482	0.6162	0.5837	0.68	0.4789	0.6621	0.4815	0.068	0.622	0.4788	0.5806

Conclusion: GPT-2 model works the best, but it is too big so in the next experiments we will use whisper-tiny model

Representation model: pretraining

Motivation:

On the experiment we figured out that decoder models work the best. But as we don't want to pretrain transaction models on supervised tasks we should figure out on which unsupervised pretraining works the best?:

Pretraining methods:

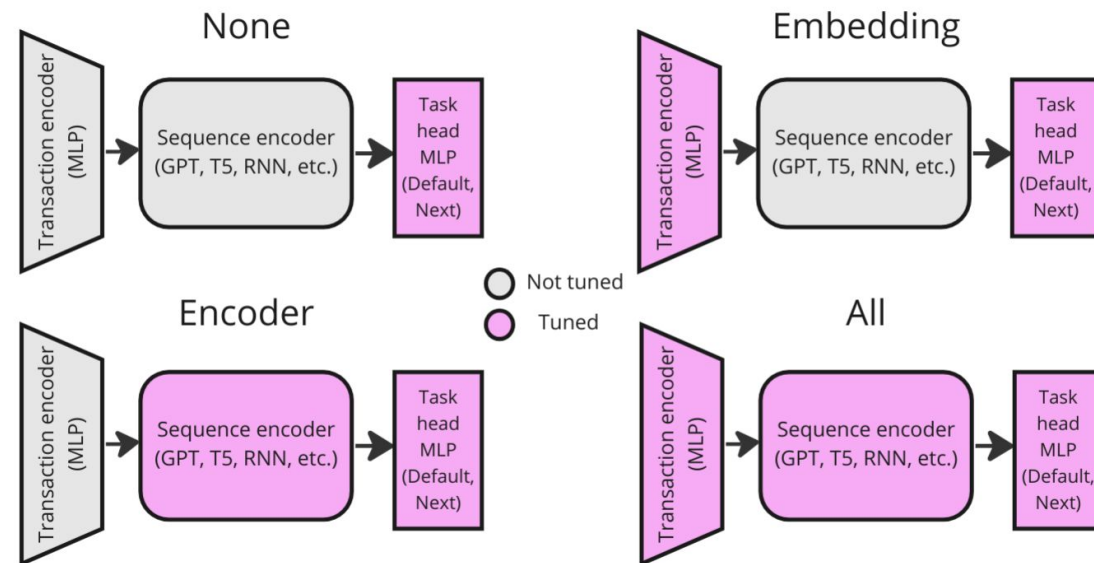
- CoLES
- Replace token detection:
- Contrastive predictive coding:

Finetuning methods:

- None
- Embedding
- Encoder
- All

Task / Metric:

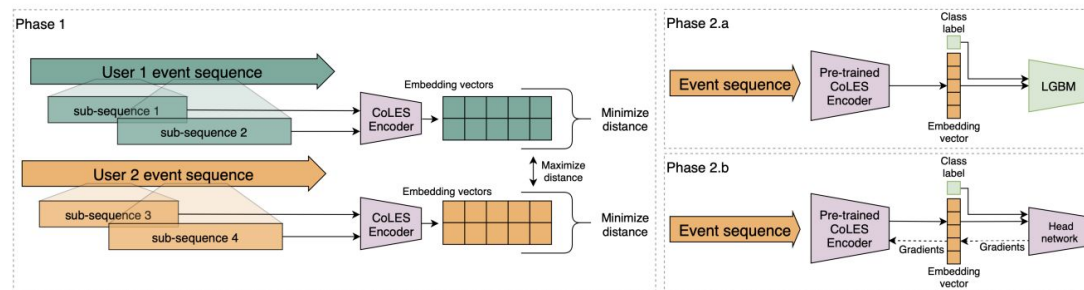
- Default (**AUC**)
- Next item prediction (**L1, Accuracy, Hit Accuracy**)



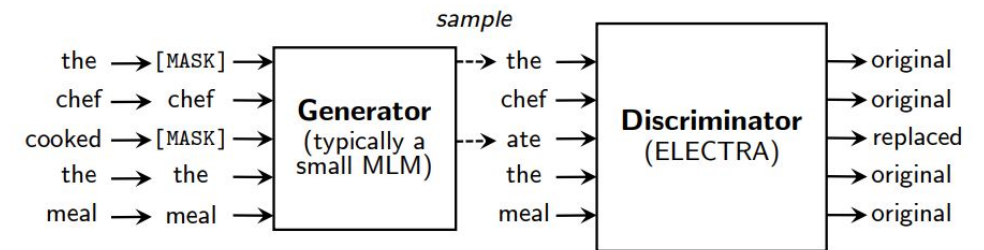
Representation model: pretraining

Pretraining methods:

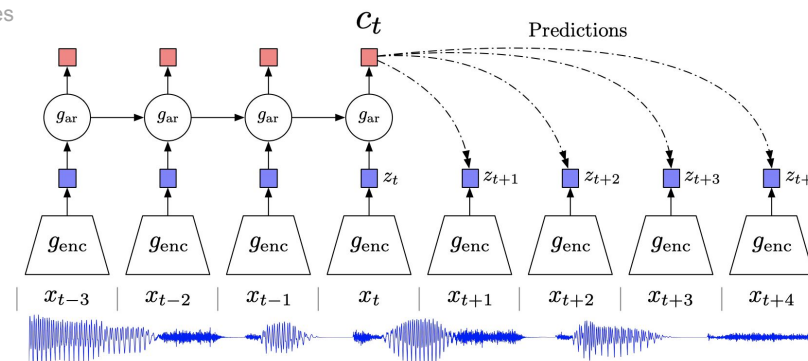
- **CoLES**. Contrastive learning method in which subsequence of the same user is considered as positive examples.
- **RTD**. Replace token detection:
- **CPC**. Contrastive predictive coding:



CoLES. Contrastive learning for event sequences



RTD. Replace token detection



CPC. Contrastive predictive coding

Representation model: pretraining

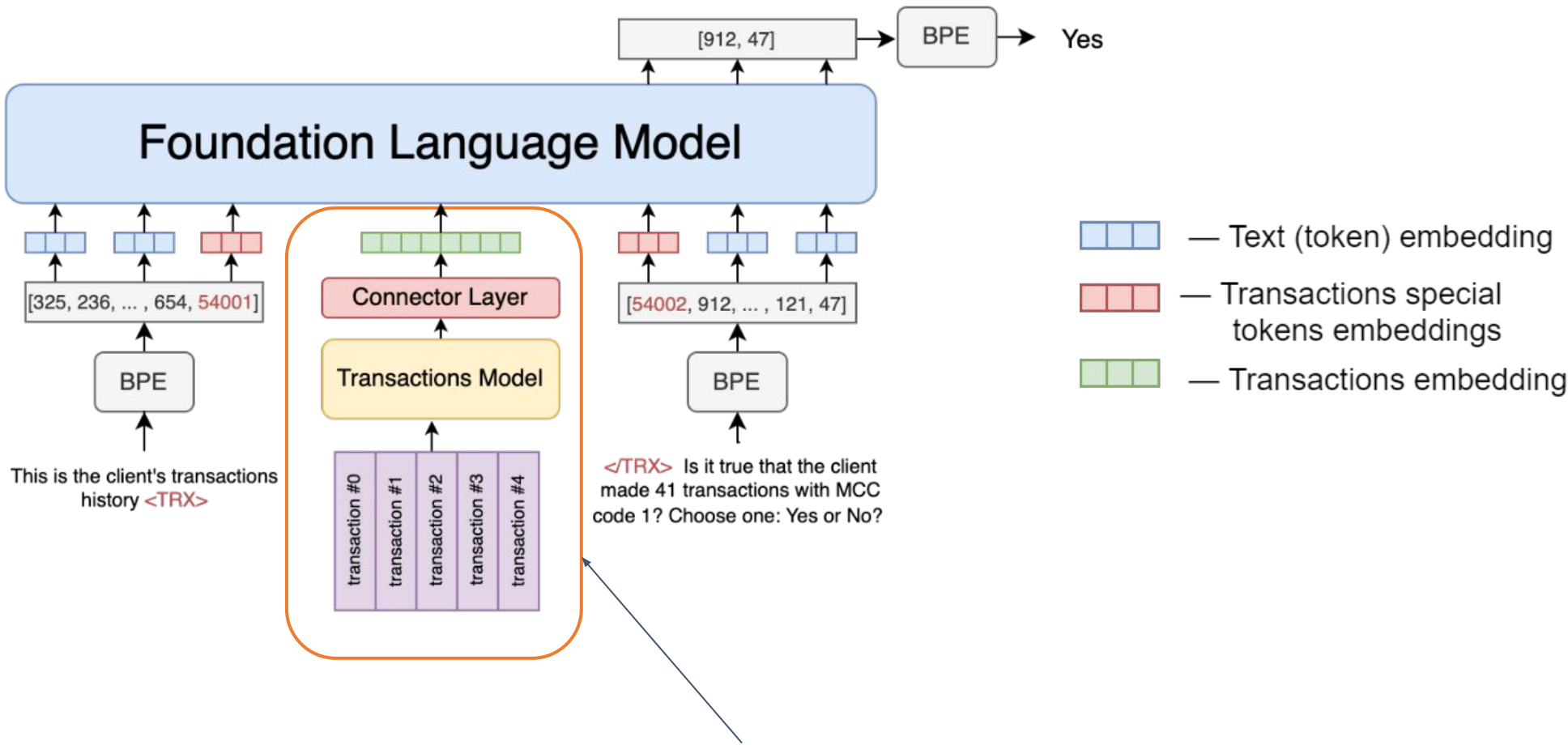
Method	Finetune all	Finetune encoder	Finetune embeddings	Finetune none
Metric	ROC AUC	ROC AUC	ROC AUC	ROC AUC
COLES-Whisper-small	0.718	0.7125	0.758	0.7039
COLES-GPT2-4	0.7799	0.7751	0.7049	0.5453
COLES-T5-8	0.7354	0.7448	0.6996	0.5645
COLES-GRU-1	0.7761	0.7714	0.7378	0.685
COLES-LSTM-1	0.7614	0.7537	0.6756	0.5755
COLES-LSTM-4	0.7578	0.7534	0.6911	0.5643
RTD-GRU-1	0.7732	0.768	0.6554	0.5792
CPC-GRU-1	0.7747	0.7695	0.7139	0.636

Representation model: pretraining

Method	Finetune all			Finetune encoder			Finetune embedding			Finetune none		
Task	Amount	MCC	24-hour acc	Amount	MCC	24-hour acc	Amount	MCC	24-hour acc	Amount	MCC	24-hour acc
Baseline	0.0788	0.4075	0.5319	0.0788	0.4075	0.5319	0.0788	0.4075	0.5319	0.0788	0.4075	0.5319
COLES-Whisper-small	0.06679	0.4857	0.6125	0.0666	0.485	0.6173	0.06924	0.4624	0.5575	0.07259	0.4517	0.5311
COLES-GPT2-4	0.06841	0.4779	0.6305	0.0712	0.4503	0.6324	0.06868	0.4775	0.6313	0.07442	0.4392	0.5045
COLES-GRU-1	0.06874	0.4746	0.6047	0.06998	0.4629	0.5419	0.06902	0.4726	0.62	0.07355	0.4528	0.4769
COLES-LSTM-1	0.06856	0.4754	0.6033	0.07047	0.4619	0.5174	0.06889	0.4724	0.6144	0.07657	0.4023	disconverge
COLES-LSTM-4	0.0677	0.4794	0.622	0.07484	0.4231	0.3107	0.0683	0.4789	0.6176	0.07488	0.4403	0.5328
RTD-GRU-1	0.06834	0.4747	0.5967	0.07045	0.4635	0.549	0.06916	0.7401	0.6089	0.07347	0.4464	0.4868
CPC-GRU-1	0.06848	0.4741	0.6033	0.07007	0.4642	0.5276	0.0687	0.4725	0.6122	0.07782	0.4259	0.611

Conclusion: We examined several methods of pretraining but none of them show significant difference so we decided to choose *whisper-tiny* pretrained on next item prediction in the next experiments.

Text interfaces: reminder



Done!

Text interfaces

Text interfaces requires - building dataset of questions and answers.

Questions type:

- Context: Questions asked about passed data.
- Predictive: Questions asked to predict the future.

Answer type:

- Binary: Yes/No questions?
- Multiple choice: On the input we pass several options to model choose between them
- Open-ended: Ask model to provide answer without any hints.

Language model as foundational model:

Flan-t5-family: small (80M), base (250M), large (780M)

Text interfaces: Question types

Question type has a two forms:

- Context questions
- Predictive questions

Context questions is much easier to answer because answer for them is provided directly in the input. These questions are used to check the concept of text interfaces. Because without good understanding of context it's impossible to provide good answers:

- What is the most popular type of transaction?
- What is the mean value of transaction amount?
- What is the least popular type of transaction?

Predictive questions are the questions about the future. These questions are equivalent to real-world tasks. Good performance on solving can show its efficiency in real-world applications.

- Will the client default?
- What type will the next transaction have?
- How many transactions will the client have in 30 days?

Text interfaces: Answer types

All questions begin:

This is the client's transaction history [*transactions_history*].

For each attribute, questions can be formulated in three statements:

1. **Binary:** the model is required to generate Yes/No tokens or True/False, depending on the wording of the question itself.

Example: This is the client's transaction history [*transactions_history*]. Is it true that the MSS transaction code 2 is the most frequent? Yes or No?

2. **Multichoice:** Models are given a choice of several options, and need to generate a numerical answer (from those given).





Example: This is the client's transaction history [*transactions_history*]. Which MCC transaction code is the most frequent? OPTIONS: 4, 18, 9, 2, 8, 13

3. **Open-ended:** the model question is asked in an open-ended form, without giving any options.

Example: This is the client's transaction history [*transactions_history*]. Which MCC transaction code is the most frequent?

Context questions: binary format

Metrics: Weighted Accuracy, weighted F1-score





	Fine-tune all		Fine-tune LM + Connector		Fine-tune Connector	
	accuracy	f1-score	accuracy	f1-score	accuracy	f1-score
flan-t5-small	0.964 (106 M. trainable params.)	0.963 (106 M. trainable params.)	0.946 (77 M. trainable params.)	0.945 (77 M. trainable params.)	 0.506 (197 K. trainable params.)	 0.01 (197 K. trainable params.)
flan-t5-base	0.948 (295 M. trainable params.)	0.947 (295 M. trainable params.)	<u>0.956</u> (247 M. trainable params.)	0.955 (247 M. trainable params.)	 (295 K. trainable params.)	 (295 K. trainable params.)

Context questions: multichoice

Metrics: Weighted accuracy, weighted F1-score



$$\text{Accuracy} = \frac{1}{N} \sum_i^N 1(y_i = \hat{y}_i)$$

+ weighted by each class support
(with ***n_classes*** = 108)

	Число опций	Fine-tune all		Fine-tune LM + Connector		Fine-tune Connector	
<i>Mempuku</i>		<i>accuracy</i>	<i>f1-score</i>	<i>accuracy</i>	<i>f1-score</i>	<i>accuracy</i>	<i>f1-score</i>
flan-t5-small	6	0.941 (106 M. trainable params.)	0.940 (106 M. trainable params.)	0.932 (77 M. trainable params.)	0.932 (77 M. trainable params.)	 0.118 (197 K. trainable params.)	 0.2 (197 K. trainable params.)
flan-t5-base	6	<u>0.961</u> (295 M. trainable params.)	0.957 (295 M. trainable params.)	0.953 (247 M. trainable params.)	0.947 (247 M. trainable params.)	 0.061 (295 K. trainable params.)	 0.01 (295 K. trainable params.)
flan-t5-base	12	-	-	0.921 (247 M. trainable params.)	0.901 (247 M. trainable params.)	-	-
flan-t5-base	24	-	-	0.917 (247 M. trainable params.)	0.903 (247 M. trainable params.)	-	-
flan-t5-base	108	-	-	0.926 (247 M. trainable params.)	0.993 (247 M. trainable params.)	-	-

Context questions: open-ended

Метрики: Weighted accuracy, weighted F1-score $\text{Accuracy} = \frac{1}{N} \sum_i^N 1(y_i = \hat{y}_i)$

	Fine-tune all	Fine-tune LM + Connector	Fine-tune Connector
	accuracy / f1-score	accuracy / f1-score	accuracy / f1-score
flan-t5-small	0.771 / 0.727 (295 M. trainable params.)	0.685 / 0.634 (77 M. trainable params.)	 (197 K. trainable params.)
flan-t5-base	0.736 / 0.682 (295 M. trainable params.)	0.764 / 0.721 (247 M. trainable params.)	 (295 K. trainable params.)

By question formulation:

flan-t5-small $\left(\begin{array}{l} \text{`Which MCC code is the most frequent?`} \rightarrow \text{Accuracy} = 0.747 \\ \text{`Choose the most frequent MCC code.`} \rightarrow \text{Accuracy} = 0.726 \end{array} \right.$

flan-t5-base $\left(\begin{array}{l} \text{`Which MCC code is the most frequent?`} \rightarrow \text{Accuracy} = 0.79 \\ \text{`Choose the most frequent MCC code.`} \rightarrow \text{Accuracy} = 0.763 \end{array} \right.$

Predictive questions: Binary format

Default: *Will the client default?*

MCC: *Will the MCC of the next transaction be 2?*



30 days: *Will the number of transactions in 30 days be greater than the median value?*

	Finetune: all			Finetune: LM+connector			Finetune: trx_encoder+connector		
Model / Task	MCC	30 days	Default	MCC	30 days	Default	MCC	30 days	Default
Metrics	Accuracy		AUC	Accuracy		AUC	Accuracy		AUC
classification model* (27 M. trainable params.)	0.7886	0.8530	0.792	0.7886	0.8530	0.792	0.7886	0.8530	0.792
flan-t5-small (106 M. trainable params.)	0.7897	0.8539	0.788	0.7853	0.8434	0.772	0.7803	0.8515	0.7794
flan-t5-base (295 M. trainable params.)	0.7886	0.8539	0.7822	0.7858	0.8429	0.773	⊘	⊘	⊘

* classification model = best model (GPT-like architecture) trained in a classical machine learning setting, trained in single task mode



Predictive questions: Multiple choice

MCC: What MCC will the following transaction have? OPTIONS: 18; 2; 7; 5; 27; 14;

	Finetune: all		Finetune: LM+connector		Finetune: trx_encoder+connector	
Model / Task	MCC	30 days	MCC	30 days	MCC	30 days
Metrics	Accuracy		Accuracy		Accuracy	
classification model* (27 M. trainable params.)	0.813	TBD	0.813	TBD	0.813	TBD
flan-t5-small (106 M. trainable params.)	0.8288	TBD	0.8254	TBD	0.7803	TBD
flan-t5-base (295 M. trainable params.)	TBD	TBD	TBD	TBD		

Predictive questions: Open-ended

MCC: What is the MCC for the next transaction?

	Finetune: all		Finetune: LM+connector		Finetune: trx_encoder+connector	
Model / Task	MCC	30 days	MCC	30 days	MCC	30 days
Metrics	Accuracy		Accuracy		Accuracy	
classification model* (27 M. trainable params.)	<u>0.70</u>	TBD	<u>0.70</u>	TBD	<u>0.70</u>	TBD
flan-t5-small (106 M. trainable params.)	0.6078*	TBD	0.6028	TBD	0.5953	TBD
flan-t5-base (295 M. trainable params.)	TBD	TBD	TBD	TBD		

Conclusion: Text interface type of model are able to solve tasks effectively even in some cases surpassing the classical type of model

* classification model = best model (GPT-like architecture) trained in a classical machine learning setting, trained in single task mode

Impact of the finetuning language model on overall NLI ability

Finetune of the language model gives an increase in the quality of answers to questions based on data of a different modality. However, in the fine-tuning phase for a particular task (especially a multimodal task), the Foundation Language model starts to "forget" its original language knowledge and starts to perform worse in standard language tasks.

Theory: Finetuning Foundation Language model reduces its overall ability to solve standard language problems

How to test:

1. Select a standard benchmark to test the punching ability of the LLM.
2. Perform experiments comparing the original model weights (before fine-tune) and the weights after finetune on a number of tasks.

Setting:

1. Model: FLAN-T5-small (77 M. parameters)
2. Benchmark: The Big-bench

The Big-bench (The Beyond the Imitation Game Benchmark)

- ~ 200 tasks, 100 languages (including programming languages)
- a set of **12 tasks** has been selected: *code_line_description*, *conceptual_combinations*, *known_unknowns*, *logic_grid_puzzle*, *misconceptions*, *misconceptions_russian*, *movie_dialog_same_or_different*, *novel_concepts*, *vitamin_c_fact_verification*, *winowhy*, *formal_fallacies_syllogisms_negation*, *logical_deduction*

Impact of the finetuning language model on overall NLI ability

	code line description		conceptual combinations		known unknowns		logic grid puzzle		misconceptions		misconceptions russian	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
flan-t5-small	31.67	55.07	28.15	55.04	50	50	33.4	33.4	51.60	51.60	2.04	8.87
flan-t5-small <i>(fine-tuned)</i>	31.67	55.07	28.15	55.04	50	50	33.4	33.4	51.60	51.60	2.04	8.87

	movie dialog same or different		novel concepts		vitamin C fact verification		winowhy		formal fallacies syllogisms negation		logical deduction	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
flan-t5-small	52.66	52.66	21.875	73.71	<u>50.5213</u>	<u>50.5213</u>	100	100	100	100	20.0	83.73
flan-t5-small <i>(fine-tuned)</i>	52.66	52.66	21.875	73.71	<u>50.5377</u>	<u>50.5377</u>	100	100	100	100	20.0	83.73

Text interfaces: multi-task setting

Objectives:

- Test text interfaces ability to deal with **more than one task at a time**, which is crucial for real world applications.
- Consider different schemes:
 - Finetuning language model + connector;
 - Finetuning transactional model + connector;

Setting:

Train a model to solve problems:

- Combination of tasks: prediction of MCC (type of transaction) + Default
- Combination of questions formats: (Binary + Open-Ended)

Multi-task learning

	Finetune: all		Finetune: LM+connector		Finetune: trx_encoder+connector	
Task	MCC + Default		MCC + Default		MCC + Default	
Metrics	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
Single task best	0.7897	0.7888	0.7897	0.7888	0.7897	0.7888
flan-t5-small (106 M. trainable params.)	TBD	TBD	<u>0.7829</u>	<u>0.7874</u>	TBD	TBD
flan-t5-base (295 M. trainable params.)	TBD	TBD	TBD	TBD	⊘	⊘

Conclusions

1. Found out the best representation for sequential data
2. Transformed tasks in text format and showed how one can use language model to solve them
3. Showed efficiency of text interfaces in solving multiple tasks, while preserving quality on language tasks.

Current Status

- Finishing experiments for multi-task learning setting
- Preparing text of the final draft

Outlook

1. Constructing text interfaces for multi-modal data
2. Increasing number of modalities on which text interfaces can work
3. Zero- / few-shot learning
4. Finetuning on new tasks / domains