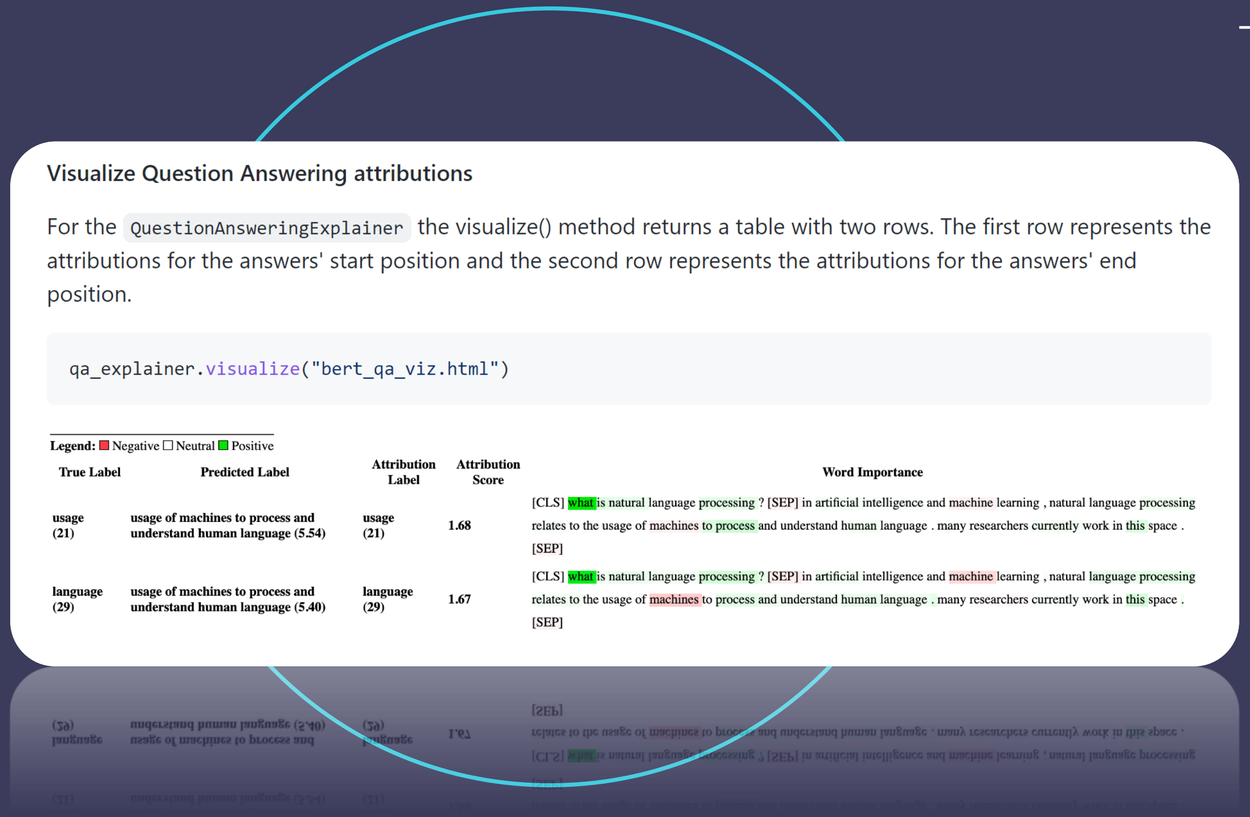
Victor Nosko, Zakhar Ponimash (Avatar Machine)

Explainit All: explainable and interpretable AI for generative neural network models transformer

Interpretable Natural Language Processing (INLP)

AGI-23 Workshop | 19 June, 2023 (Full-Day) | Stockholm, Sweden & Virtual

Task: Explainable AI



https://github.com/cdpierse/transformers-interpret

Text generation must become controlled

Objective:
1. Explain why the generation of transformers is such and what it depends on.
2. Evaluate the applicability of the model to a specific downstream problem.
(FSI Grant - CodeAi 2022 https://fasie.ru/press/fund/kod-ai-4-results/)

1. Challenge: Current libraries visualize attention by words - is not human-understandable
2. Generation is dangerous: no factual control, "unnecessary" topics, entities can appear
3. Can't trust ODQA question-answer system in complex and sensitive industries (medicine, law, etc.)
4. No metrics to assess the quality of interpretability of AI models

ExplainitAll Library: scientific novelty

ExplainitAll demo work

Research: interpretability techniques

The novelty of the proposed explainability methodology:

1. A methodology for the interpretability of large transformational models.
2. Methodology for evaluating the quality of interpretability depending on model parameters and architecture
3. New interpretability technique for question-answer systems with explainable parameters and completeness metrics

An example of use in the QA system:

Question: "What drug L in liquid form is acceptable to use to treat disease X given contraindication Y in this patient, age 5 years, with a fever of 37 ?“

Generative Model Answer: "Children ages 5-7 years old can take the following brands of antipyretics: [B,B] based on substance E. However, there are contraindications to Y. According to the sources in the articles [S,S,S]“

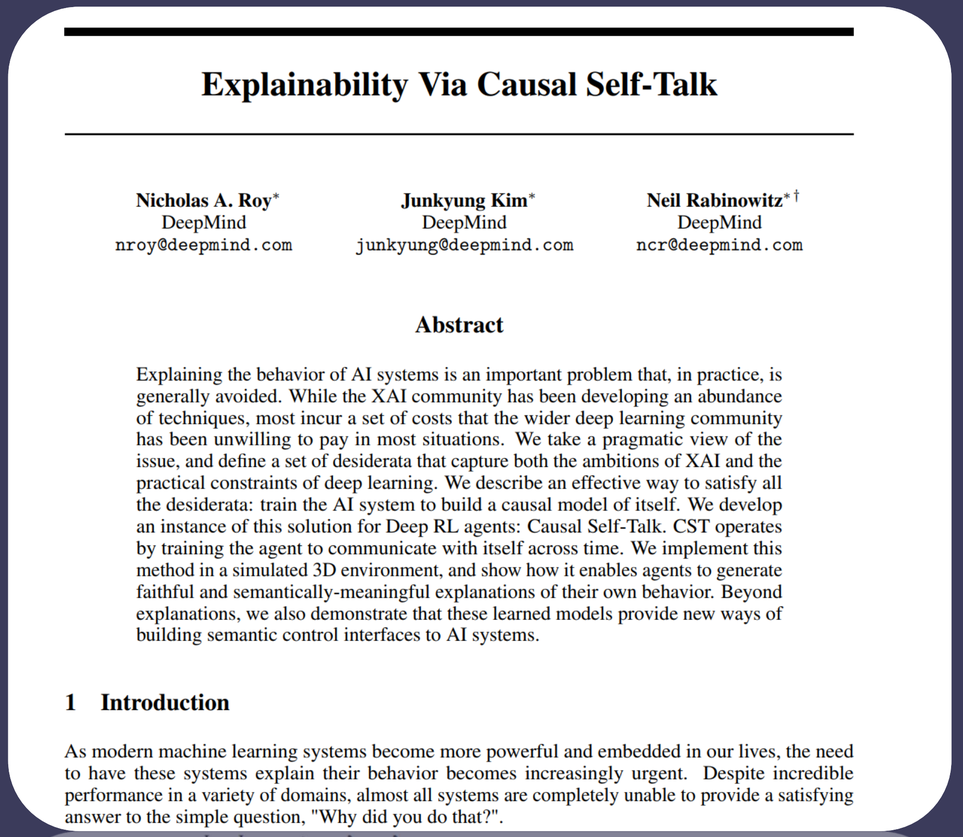
ExplainitAll result:

ExplainitQ interpretability quality metric: the confidence level of the answer is 80% based on the sources of S.

Interpretation: "Entities L and X occur with frequency F in sources [S,S,S], disease description tokens, temperatures are determinant in relative comparison to descriptions of other treatments" Set of combinations of tokens and their probabilities in table: ...

Explainit library: comparison with analogues

Explainability Via Causal Self-Talk



Explainability Via Causal Self-Talk

Nicholas A. Roy* DeepMind nroy@deepmind.com

Junkyung Kim* DeepMind junkyung@deepmind.com

Neil Rabinowitz*† DeepMind ncr@deepmind.com

Abstract

Explaining the behavior of AI systems is an important problem that, in practice, is generally avoided. While the XAI community has been developing an abundance of techniques, most incur a set of costs that the wider deep learning community has been unwilling to pay in most situations. We take a pragmatic view of the issue, and define a set of desiderata that capture both the ambitions of XAI and the practical constraints of deep learning. We describe an effective way to satisfy all the desiderata: train the AI system to build a causal model of itself. We develop an instance of this solution for Deep RL agents: Causal Self-Talk. CST operates by training the agent to communicate with itself across time. We implement this method in a simulated 3D environment, and show how it enables agents to generate faithful and semantically-meaningful explanations of their own behavior. Beyond explanations, we also demonstrate that these learned models provide new ways of building semantic control interfaces to AI systems.

1 Introduction

As modern machine learning systems become more powerful and embedded in our lives, the need to have these systems explain their behavior becomes increasingly urgent. Despite incredible performance in a variety of domains, almost all systems are completely unable to provide a satisfying answer to the simple question, "Why did you do that?".

https://arxiv.org/pdf/2211.09937.pdf

Causal Self-Talk (DeepMind)

Качественные характеристики:

1) CST – approach with the agent's communication with himself over time
2) CST-RL: The messages zt should allow earlier versions of the same agent to maximize return given the current environment state.
3) CST-MR: The messages zt should allow earlier versions of the same agent to reconstruct the next internal state, mt+1.
4) CST-PD: The messages zt should allow earlier versions of the same agent to recover the true current (and future) policy.

Clear and structured explanations

Analysis of the requirements for "explainability"

What Explanability should be:

1) Grounded
2) Flexible
3) Minimally-interfering - the use of an explanability model should have little effect on the underlying generation model.
4) Scalable
5) Faithful - truthful

2 Desiderata for explanatory models

Our goal is to build an explanatory model of an AI system, i.e. a mechanism that translates between its internal representations and/or computations, and some other representational form. We set out five key desiderata for such a model. We flesh these out in more detail in Section 3.

Grounded. For the model to be useful to external users, it should deploy a representation language that is semantically-grounded. This may require additional data to train.

Flexible. The method should be able to produce models of many different representational forms (e.g. classes, graphs, strings). Ideally it should also be sufficiently general to be agnostic to the base system's input/output modalities, and possibly even its overall architecture.

Minimally-interfering. Training and inference on the model should have as little impact as possible on the training and inference of the base system. Solutions which are disruptive to performance are highly unlikely to be adopted in practice.

Scalable. Once a solution is found, it is straightforward to apply it to any size network.

Faithful. The model should provide accurate descriptions of the underlying system. The gold standard is a *causal model* of the agent, which facilitates validation of its accuracy through interventions [38]. This is possibly the most challenging to achieve, and is discussed further in the specific context of decoders in Section 3.

Causal self-talk. To overcome this problem, we need to create an information asymmetry between speaker and listener. We introduce ***Causal Self-Talk*** **(CST)**: we encourage f_θ to be sensitive to the message z_t by creating an augmented data distribution where the speaker and listener diverge. Fortunately, we have a ready source of alternate listeners: internal states $m_{t'}$ for $t' < t$.

https://arxiv.org/pdf/2211.09937.pdf

Causal Self-Talk (DeepMind)

Learning approach: reStructured Pre-training

Teaching structure

2 reStructured Pre-training

2.1 Paradigm Shift in Modern NLP

The paradigm for solving NLP tasks is changing rapidly and is still ongoing. For example, as summarized by (Liu et al., 2021d), the development history of modern natural language technology can be abstracted into four paradigms. We revisit and extend it from a new perspective as summarized in Tab. 1.

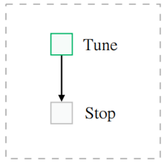
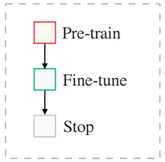
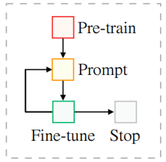
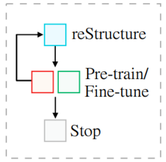
	Fully Supervised Learning	Pre-train, Fine-tune		Pre-train, Prompt Predict	reStructure, Pre-train, Fine-tune
Illustration					
Engineering	I: Feature	II: Architecture	III: Objective	IV: Prompt	V: Data reStructure
Example	SVM	Word2vec	BERT	GPT3	RST
Pre-training Data	-	ngram	plain text	plain text	reStructured text
Supported Signal	-	Limited	Limited	Limited	Unlimited
Transparency	-	✗	✗	✗	✓

Table 1: Five paradigms in NLP. There is no recurrent connection in paradigm I-III since models tuned for one task usually cannot be directly reused for the others due to the discrepancies between output layers. "Transparency" refers to, during the pre-training stage, whether the model stores data in a way that can be understood by downstream tasks so that they can access the required data efficiently.

https://arxiv.org/pdf/2206.11147.pdf

reStructured Pre-training (ExpressAI)

A different approach to training a Transformer, which are then easier to "understand":

1) Signal Definition Signal is the useful information that can serve for learning knowledge for specific tasks and instructs models for learning optimization. As the first step for restructured learning, we first need to figure out what signals naturally exist in the world and are collected and available.
2) Data Mine Identification In the real world, we have access to various data sources such as news websites, Wikipedia, knowledge bases, or even online videos. Data Mine refers to a collection of data that are rich in diverse types of signals. Once signals have been defined, searching for suitable data mines is expected
3) Signal Extraction How to effectively extract signals from data mines also matters for restructured learning.

Solution Overview: Shap

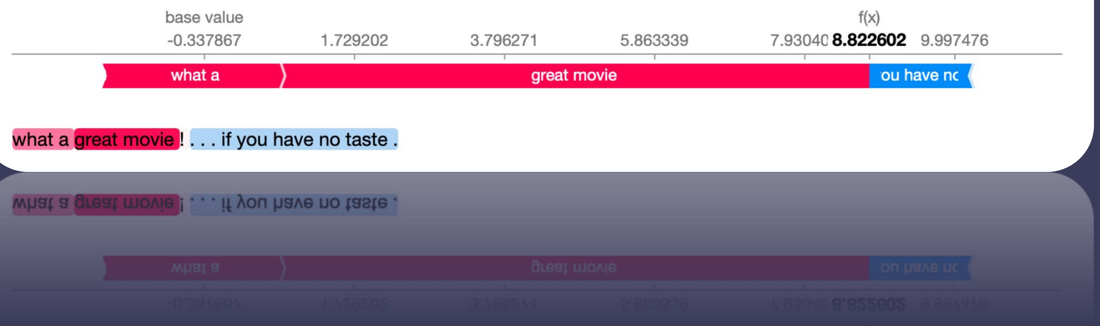
Shap for transformers

```
import transformers
import shap

# load a transformers pipeline model
model = transformers.pipeline('sentiment-analysis', return_all_scores=True)

# explain the model on two sample inputs
explainer = shap.Explainer(model)
shap_values = explainer(["What a great movie! ...if you have no taste."])

# visualize the first prediction's explanation for the POSITIVE output class
shap.plots.text(shap_values[0, :, "POSITIVE"])
```



SHAP has specific support for natural language models like those in the Hugging Face transformers library. By adding coalitional rules to traditional Shapley values we can form games that explain large modern NLP model using very few function evaluations. Using this functionality is as simple as passing a supported transformers pipeline to SHAP:

1) Similar to transformers-interpret
2) Applicable for classifiers, e.g. sentiment analysis tasks

https://github.com/slundberg/shap

Our solution for Captum

Библиотека Captum

```
attr = ablator.attribute(inputs=tokenization_result['input_ids'],
                         baselines=torch.full((13,), pad_token_id),
                         additional_forward_args=(tokenization_result['token_type_ids'],
                                                  tokenization_result['attention_mask'], ))

shape_a = attr.shape
enc_sh = encoded_input['input_ids'].shape
shape_2 = shape_a[0]//enc_sh[0]
shape_1 = shape_a[0]//shape_2
shape_3 = enc_sh[1]
new_a = attr.reshape(shape_1,shape_2,shape_3)

data = torch.sqrt(torch.sum(new_a**2, dim=1)/shape_2)
data = data.to('cpu').numpy()

lines = []

for line_num in range(len(data)):
    pairs = []
    toks = tokenizer.tokenize(sentences[line_num])
```

Implementing Ablation (Avatar Machine)

Ablation feature approach

Ablation - runs the sequence through sbert, replaces each token with a <pad> token, then counts the element differences between the original vector and the one obtained after replacement.

The greater the contribution of the word to the result, the greater will be the difference (sum of squares) for each token

https://captum.ai/

Our solution for Interpretation

Interpretation of Embedders (sbert)

```
In [3]: #Load AutoModel from huggingface model repository
        tokenizer = AutoTokenizer.from_pretrained("sentence-transformers/all-MiniLM-L6-v2")
        model = AutoModel.from_pretrained("sentence-transformers/all-MiniLM-L6-v2")

        ablator = FeatureAblation(forward_function)

In [4]: #Sentences we want sentence embeddings for
        sentences = ['This framework generates embeddings for each input sentence',
                     'The quick brown fox jumps over the lazy dog.',
                     'Sentences are passed as a list of string.',
                    'The quick brown fox jumps over the lazy dog.']

In [5]: emb_interp(sentences)

Out[5]: [[('this', 0.11327786, False),
          ('framework', 0.12338372, False),
          ('generates', 0.0891755, False),
          ('embeddings', 0.12026673, True),
          ('for', 0.055054933, False),
          ('each', 0.08235276, False),
          ('input', 0.10753967, False),
          ('sentence', 0.14927785, False)],
         [('the', 0.10460499, False),
          ('quick', 0.09024346, False),
          ('brown', 0.12647457, False),
          ('fox', 0.15491463, False),
          ('jumps', 0.13338014, False),
          ('over', 0.093634404, False),
          ('the', 0.06714732, False),
          ('lazy', 0.11681719, False),
          ('dog', 0.113603234, False),
          ('.', 0.065999806, False)],
         [('sentences', 0.15869598, False),
          ('are', 0.06814716, False),
          ('passed', 0.10024548, False),
          ('as', 0.06324373, False),
          ('a', 0.054334678, False),
          ('list', 0.11290424, False),
          ('of', 0.051345263, False),
          ('string', 0.12499759, False),
```

This set of sentences produces a list of important words (Avatar Machine)

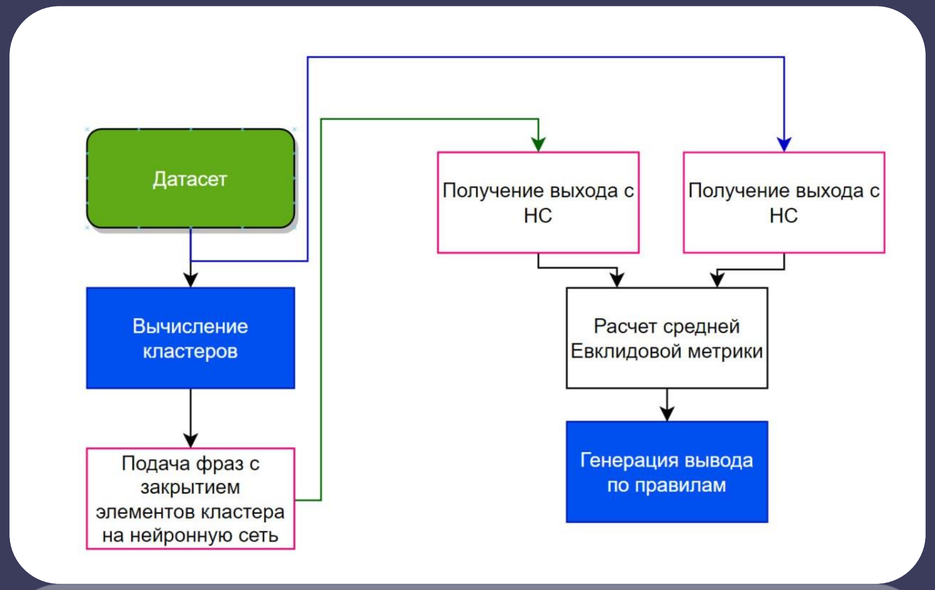
Analyzed the model from SBERT with Ablation and learned the importance of each word in the sentence.

Result: It is possible to test any model from Sbert.

1) ablator = FeatureAblation(forward_function)
2) A normalized Euclidean metric is used

Interpretation Model Algorithm

Interpretation of the model, scheme



$$d(X,Y) = \frac{1}{M}\sum_{m=1}^{M}\sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(X_n^{(m)} - Y_n^{(m)}\right)^2},$$

$$X^{(m)} \in X, Y^{(m)} \in Y,$$

$$X^{(m)}, Y^{(m)} \in \mathbb{R}^N$$

Interpretation Model Algorithm (Avatar Machine)

1. **We give text as input, there is an assumption that neural network should be sensitive to some semantic axis (cluster).**
2. **We form a cluster using word vectorization algorithm (w2v or other).**
3. **Then, for each cluster we count the total importance - how important it is for our neural network. Importance is counted in Captum + Average Euclidean distance. We count for the whole sample - how much this word changes the vector on average. And since the word is in the cluster - so we can understand how much weight a particular meaning cluster has.**

Consists of:
1. the Embedder.
2. the Embedder
3. Clusterizer, examples of clusters (drugs, animals, people, organizations, etc.)

Step 1: Cluster expansion

FAISS (for searching in a cluster)

```
get_words('бес_NOUN', top_k = 2000)
```

```
[('бес', 0.0),
 ('сатана', 363.84744),
 ('дьявол', 369.8022),
 ('демон', 437.05563),
 ('искуситель', 466.14645),
 ('диавол', 466.86288),
 ('геенна', 474.54553),
 ('нечестивец', 477.10327),
 ('блудник', 509.9815),
 ('шайтан', 513.1277),
 ('кикимора', 516.77466),
 ('богохульник', 518.09656),
 ('бражник', 520.74976),
 ('мудрование', 521.3508),
 ('пакостник', 522.25476),
 ('волхво', 523.3657),
 ('любодеяние', 526.59326),
 ('братий', 527.48303),
 ('ворог', 528.5243),
 ('нечестие', 528.8367),
 ('леший', 530.5376),
 ('помысл', 531.0069),
 ('чернокнижник', 533.01434),
 ('заклинатель', 536.5911),
 ('беззаконник', 536.7515),
 ('пещь', 537.0691),
 ('непотребство', 537.1166),
 ('аспид', 537.3877),
 ('бесенка', 538.10693),
 ('вопиять', 539.20154),
 ('мнози', 539.3369),
 ('святоша', 539.42163),
 ('кликуша', 541.27295),
 ('мы', 541.74615),
 ('фурия', 542.2231),
 ('нечестий', 542.22546),
 ('помело', 542.354),
 ('антихристов', 543.1851),
 ('-искуситель', 543.7405),
 ('помысол', 543.8076),
```

```
similar_words_etalon(('собака_NOUN', 'кошка_NOUN', "рыба_NOUN"), top_k = 10)
```

```
[('кот_NOUN', 0.6858417391777039),
 ('пес_NOUN', 0.6715787053108215),
 ('щенок_NOUN', 0.6543251276016235),
 ('дворняжка_NOUN', 0.6343124508857727),
 ('котенок_VERB', 0.6273977160453796),
 ('зверек_NOUN', 0.6264833807945251),
 ('крыса_NOUN', 0.6198749542236328),
 ('собака_ADV', 0.6190679669380188),
 ('дворняга_NOUN', 0.6161069869995117),
 ('кошка_PROPN', 0.594084620475769)]
```

```
k_similar_words_knn(('собака_NOUN', 'кошка_NOUN', "рыба_NOUN"), top_k = 10)
```

```
[('собака', 1.0),
 ('пес', 0.8104413350219728),
 ('дворняжка', 0.7073254377245829),
 ('кошка', 0.9999999999999999),
 ('кот', 0.8213909925416654),
 ('котенок', 0.7216601758447425),
 ('рыба', 1.0),
 ('карп', 0.7310271496913975),
 ('окунь', 0.7307456968993228)]
```

```
k_similar_words_faiss(('собака_NOUN', 'кошка_NOUN', "рыба_NOUN"), top_k = 10)
```

```
[('собака', 0.0),
 ('кошка', 0.0),
 ('рыба', 0.0),
 ('кот', 222.20175),
 ('пес', 277.53503),
 ('котенок', 320.07812),
 ('дворняжка', 388.07257),
 ('карп', 402.93842),
 ('окунь', 404.3291)]
```

Cluster extension using the FAISS method: helps the user to add words to the desired cluster.

The updated cluster is then fed into the interpretation method.

Step 2: Analyze the impact of the cluster on network output

Model interpretation, test

```
In [86]: cl_creator = ClusterCreator("Животные", ["Собака", "Кошка", "рыбы"])
         claster_anim = cl_creator.get_cluster(match_func)

         cl_creator = ClusterCreator("Лекарства", ["Пенициллин", "Антибиотик", "антидепрессанты"])
         claster_medicines = cl_creator.get_cluster(match_func)

         gr = get_g_atr(dataset)
         NNWTest(claster_anim, gr)
         NNWTest(claster_medicines, gr)

         net = RuleGenerate("rubert-base-cased-sentiment")
         net.calc_params([claster_anim, claster_medicines])

         print(net.test(claster_anim))
         print(net.test(claster_medicines))

         Сеть "rubert-base-cased-sentiment" чувствительна к кластеру сущностей "Животные"
         Сеть "rubert-base-cased-sentiment" не чувствительна к кластеру сущностей "Лекарства"

In [67]: claster_anim.r_mean_cluster

Out[67]: 0.06539037500894955
```

Calculation of the sensitivity of the Transformer model to a particular cluster (Avatar Machine)

Example of interpreter work: we give as input a list of clusters (each cluster is defined by a list of words), neural network, dataset.
The method evaluates neural network sensitivity to different semantic axes.

Advantage: ***Collect statistics on axes of meaning (semantic clusters) and not on a particular word***

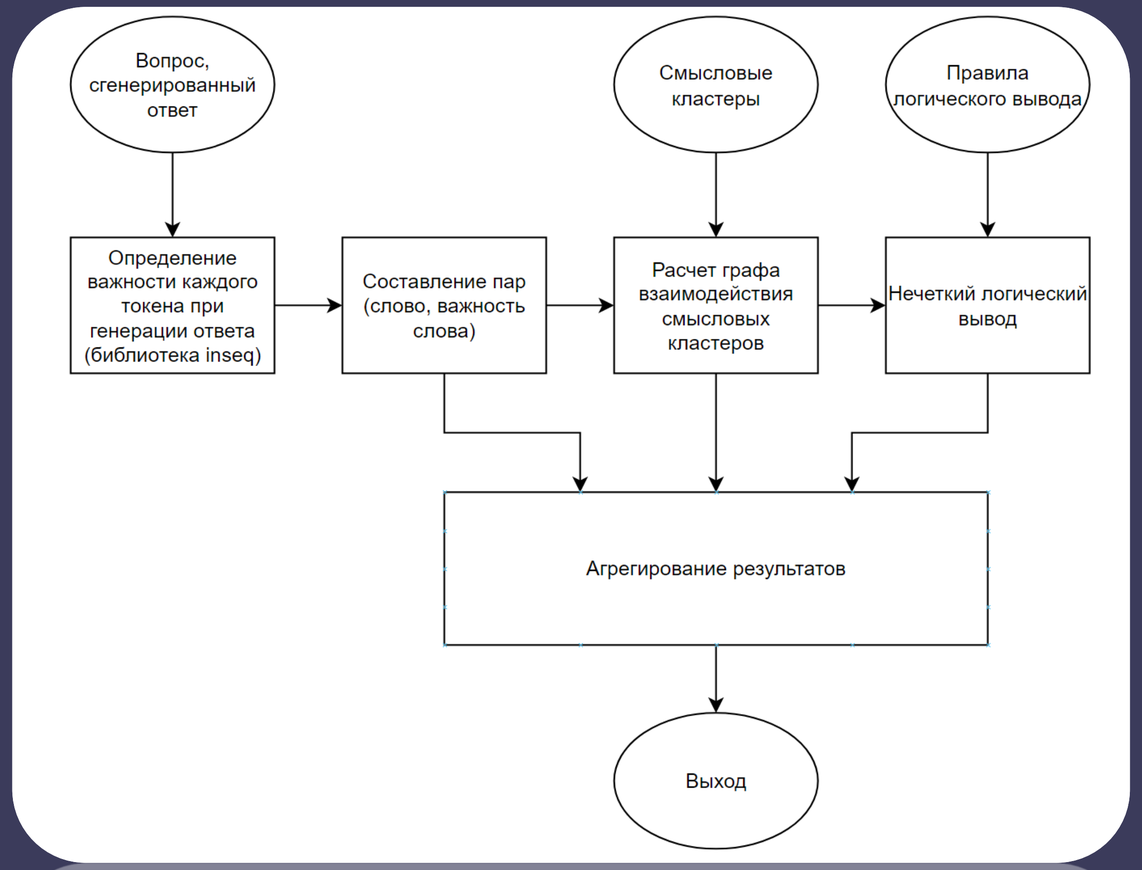
Work in progress:

1) 1.96 (threshold level of triggering)
2) Metric of "reliability" of a feature cluster
3) Resistance to different types of transformers: in this case the test for the rubert model

Generative neural networks

The user controls the semantic cluster

Scheme of the generative model interpreter



Interpreter Scheme

Schematic of the interpreter of generative question-answer models.

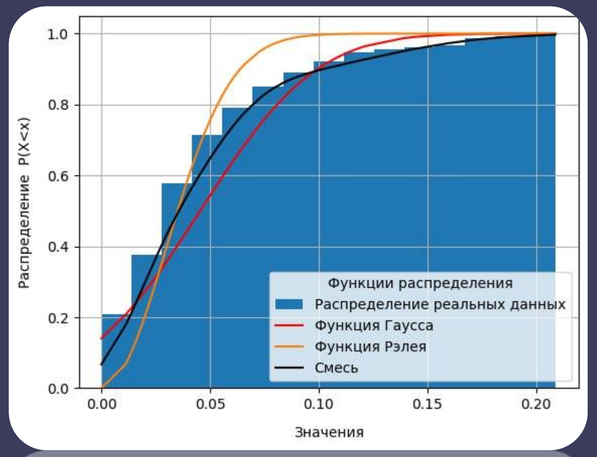
The system receives a question, a model-generated answer, and the generative model itself, e.g. gpt2.

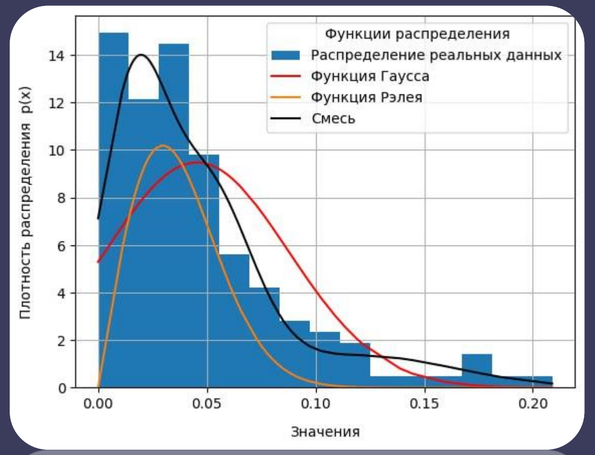
Advantage: interprets importance, outputs not only token-value, but also word-value, cluster-value

Work in progress:
Fuzzy logical inference will be added in the future.

Normal distribution

Gaussian distribution function





Three-component Gaussian mixture

$$\Phi(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x} \exp\left[\frac{1}{2}\cdot\left(\frac{x-\mu}{\sigma}\right)^2\right]dx$$

Through the same error function:

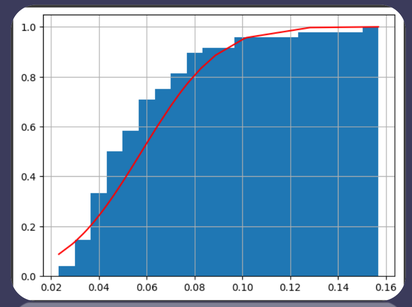
$$\Phi(x,\mu,\sigma) = \frac{1}{2}\left[1+\operatorname{erf}\left(\frac{x-\mu}{\sigma\cdot\sqrt{2}}\right)\right]$$

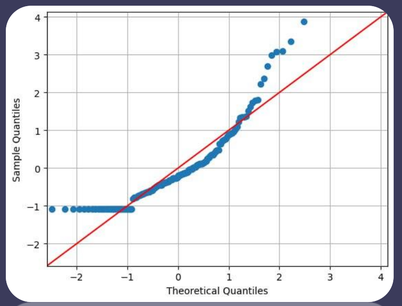
$$f(x) = \sum_{k=1}^{K} w_k \cdot \frac{1}{\sqrt{2\pi\cdot D_k^*}}\exp\left(-\frac{(x-\mu_k^*)^2}{2\cdot D_k^*}\right)$$

$$F(X) = \sum_{k=1}^{K}\left[w_k \cdot \int_{-\infty}^{X}\frac{1}{\sqrt{2\pi\cdot D_k^*}}\exp\left(-\frac{(x-\mu_k^*)^2}{2\cdot D_k^*}\right)dx\right]$$

$$F(X) = \sum_{k=1}^{K}\left[w_k \cdot \frac{1}{2}\left(1+\operatorname{erf}\left(\frac{x-\mu_k^*}{\sigma_k^*\cdot\sqrt{2}}\right)\right)\right]$$

Calculating the density and distribution function





Interpreter Scheme

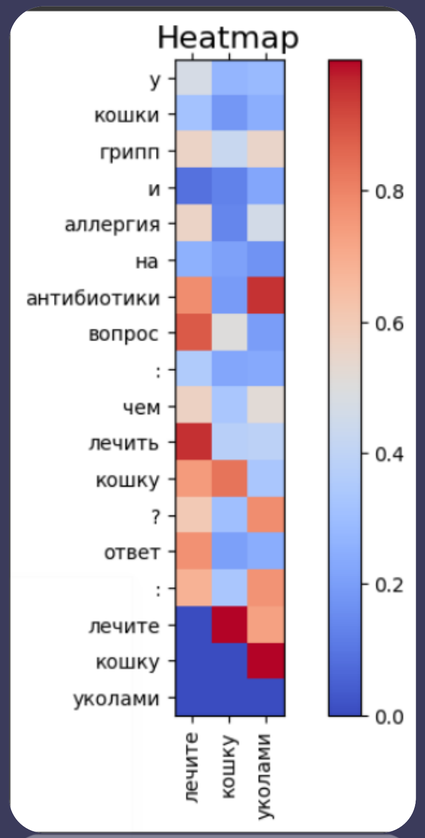
Scheme of the generative model interpreter

	-	0	1	2
0	-	лечите	кошку	уколами
1	у	0.056087	0.042276	0.043502
2	кошки	0.045771	0.034833	0.040392
3	грипп	0.061911	0.053355	0.061317
4	и	0.023337	0.028954	0.038474
5	аллергия	0.061961	0.030145	0.05554
6	на	0.041215	0.037297	0.033575
7	антибиотики	0.077689	0.035665	0.100846
8	вопрос	0.088842	0.058184	0.036294
9	:	0.048278	0.038568	0.0392
10	чем	0.062294	0.046986	0.059035
11	лечить	0.101858	0.04997	0.050664
12	кошку	0.074662	0.082715	0.046872
13	?	0.064651	0.044979	0.077569
14	ответ	0.076814	0.036999	0.040266
15	:	0.06998	0.046915	0.076264
16	лечите	NaN	0.156895	0.073449
17	кошку	NaN	NaN	0.128323
18	уколами	NaN	NaN	NaN

To (the sum in each column is approximately 1)

	-	0	1	2
0	-	лечите	кошку	уколами
1	у	0.470976	0.270211	0.286279
2	кошки	0.317147	0.183299	0.24643
3	грипп	0.56144	0.428774	0.552287
4	и	0.088217	0.128724	0.22343
5	аллергия	0.562212	0.138754	0.462493
6	на	0.256674	0.209944	0.170555
7	антибиотики	0.77959	0.192053	0.953052
8	вопрос	0.886165	0.503625	0.198834
9	:	0.352793	0.224531	0.231991
10	чем	0.567333	0.334236	0.516878
11	лечить	0.956801	0.377637	0.387972
12	кошку	0.742997	0.833259	0.332618
13	?	0.603197	0.306215	0.778197
14	ответ	0.769335	0.206605	0.24488
15	:	0.680733	0.333224	0.762754
16	лечите	0.0	0.999944	0.727489
17	кошку	0.0	0.0	0.997004
18	уколами	0.0	0.0	0.0

After



Visualization

We interpreted the integral gradient using a mixture of distributions.

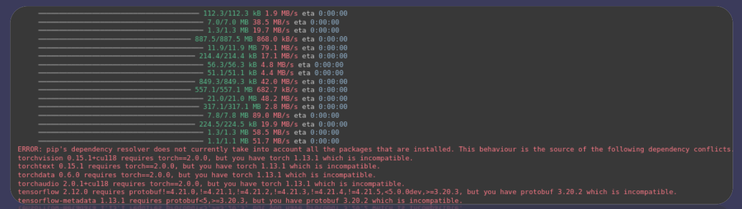
In each cell, we wrote down the probability that the random relation would be lower than the value in that cell

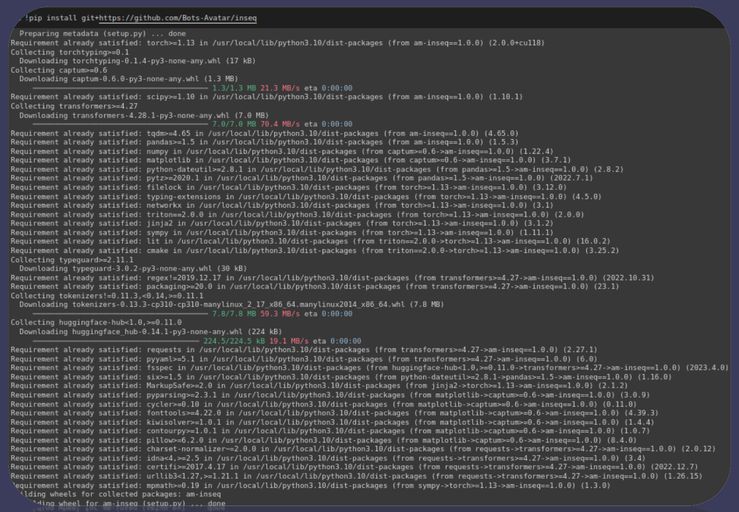
When generating the word "shots" the model looked at the word "antibiotics" and the number at the intersection is 0.1 (gradient), then look at the interpretation: line 7, the probability is 0.95 - that the connection will be random between these two words.

Improving the Inseq library

Works with the Russian language

1. **Installing the original Inseq takes 5 minutes on Colab, once fixed it takes 10 seconds**
2. **Inseq works correctly with the Russian language after being fixed**
3. **Inseq can give back an object with column names and values, or a pandas dataframe that is easy to use**





	ĠĐ»Đµ	ÑĪĐ¸ÑĤĐµ	ĠĐµĐµ	ĠÑĥĐ²Đ¼Đ»	Đ°Đ¼Đ¸
Ñĥ	0.059	0.05	0.05	0.05	0.045
ĠĐ°Đ¼ÑĪĐ°Đ¸	0.049	0.034	0.041	0.047	0.021
ĠĐ³ÑĢĐ¸Đ¿	0.067	0.037	0.049	0.066	0.028
Đ¿	0.016	0.011	0.014	0.012	0.019
ĠĐ¸	0.023	0.024	0.023	0.022	0.041
ĠĐ°Đ»Đ»ĐµÑĢ	0.066	0.043	0.047	0.069	0.018
Đ³Đ¸Ñı	0.02	0.018	0.02	0.018	0.015
ĠĐ½Đ°	0.038	0.028	0.029	0.036	0.03
ĠĐ°Đ½ÑĤĐ¸Đ±Đ¸Đ¼	0.083	0.048	0.052	0.109	0.02
ÑĤĐ¸Đ°Đ¸	0.03	0.027	0.023	0.021	0.011
ĠĐ²Đ¼Đ¿ÑĢĐ¾Ñģ	0.084	0.067	0.067	0.046	0.02
:	0.043	0.037	0.038	0.025	0.021
ĠÑĪĐµĐ¼	0.062	0.048	0.048	0.052	0.024
ĠĐ»ĐµÑĪĐ¸ÑĤÑĮ	0.104	0.102	0.067	0.057	0.023
ĠĐ°Đ¼ÑĪĐ°Ñĥ	0.079	0.049	0.08	0.072	0.024
?	0.055	0.048	0.033	0.027	0.039
ĠĐ¾ÑĤĐ²ĐµÑĤ	0.071	0.071	0.047	0.034	0.033
:	0.05	0.069	0.035	0.027	0.063
ĠĐ»Đµ		0.19	0.093	0.065	0.071
ÑĪĐ¸ÑĤĐµ			0.144	0.071	0.088
ĠĐµĐµ				0.076	0.175
ĠÑĥĐ²Đ¼Đ»					0.169
Đ°Đ¼Đ¸					

	-	ле	чите	ее	укол	ами
0	у	0.058619	0.049994	0.049727	0.049901	0.045417
1	кошки	0.049096	0.034235	0.041316	0.047159	0.020970
2	грип	0.066542	0.037406	0.049027	0.065975	0.028314
3	п	0.016028	0.011118	0.013710	0.011716	0.018948
4	и	0.023173	0.023844	0.023113	0.021786	0.041113
5	аллер	0.065910	0.042683	0.046733	0.069034	0.017998
6	гия	0.019914	0.018258	0.019920	0.017915	0.014776
7	на	0.038109	0.027618	0.028741	0.036145	0.029980
8	антибио	0.082750	0.047610	0.052231	0.109257	0.020265
9	тики	0.029704	0.027395	0.023152	0.020617	0.011348
10	вопрос	0.084305	0.066821	0.066649	0.045575	0.020213
11	:	0.043063	0.036507	0.038105	0.025346	0.020683
12	чем	0.062447	0.047860	0.047915	0.051804	0.024199
13	лечить	0.103880	0.101804	0.067480	0.056650	0.023167
14	кошку	0.079276	0.048635	0.080197	0.071967	0.023947
15	?	0.055327	0.048100	0.033248	0.026800	0.039162
16	ответ	0.071427	0.070671	0.046542	0.033776	0.033186
17	:	0.050432	0.069388	0.034862	0.027482	0.062972
18	ле	NaN	0.190053	0.093079	0.064793	0.071131
19	чите	NaN	NaN	0.144253	0.070699	0.088215
20	ее	NaN	NaN	NaN	0.075604	0.174612
21	укол	NaN	NaN	NaN	NaN	0.169383
22	ами	NaN	NaN	NaN	NaN	NaN

Before and after: c 300 -> 10 sec

https://github.com/Bots-Avatar/inseq

Example of use

Interpretation

```
[55]  1  model_name = "sberbank-ai/rugpt3small_based_on_gpt2"
      2  interp = InterpretationGPT2Like(model_name)

Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.

[56]  1  clusters_discr = [
      2      {'name':'Животные', 'centroid':['собака', 'кошка', 'заяц'], 'top_k':140},
      3      {'name':'Лекарства', 'centroid':['уколы', 'таблетки', 'микстуры'], 'top_k':160},
      4      {'name':'Болезни', 'centroid':['простуда', 'орви', 'орз', 'грипп'], 'top_k':20},
      5      {'name':'Аллергия', 'centroid':['аллергия'], 'top_k':20}
      6  ]
      7
      8  interp_data = interp.interp('Чем лечить собаку у которой насморк? Ответ:', 'Лечите ее антибиотиками', clusters_discr)

WARNING:inseq.models.attribution_model:Generation arguments {'max_new_tokens': 1900} are provided, but constrained decoding is ena
Attributing with integrated_gradients...: 100%|██████████| 18/18 [00:39<00:00,  5.65s/it]

      1  interp_data["cluster_imp"]
```

-	0	1	2
0	-	Неименованный кластер	Неименованный кластер	Лекарства
1	Неименованный кластер	0.849328	0.63492	0.660622
2	Неименованный кластер	0.859878	0.497953	0.596082
3	Животные	0.337307	0.807104	0.567378
4	Неименованный кластер	0.124738	0.09501	0.16686
5	Неименованный кластер	0.102928	0.15355	0.086209
6	Болезни	0.24049	0.30805	0.769098
7	Неименованный кластер	0.831561	0.390316	0.136212
8	Неименованный кластер	0.997754	0.552865	0.214022
9	Неименованный кластер	0.920351	0.268559	0.120488
10	Неименованный кластер	0.0	0.785057	0.751007
11	Неименованный кластер	0.0	0.0	0.834323
12	Лекарства	0.0	0.0	0.0

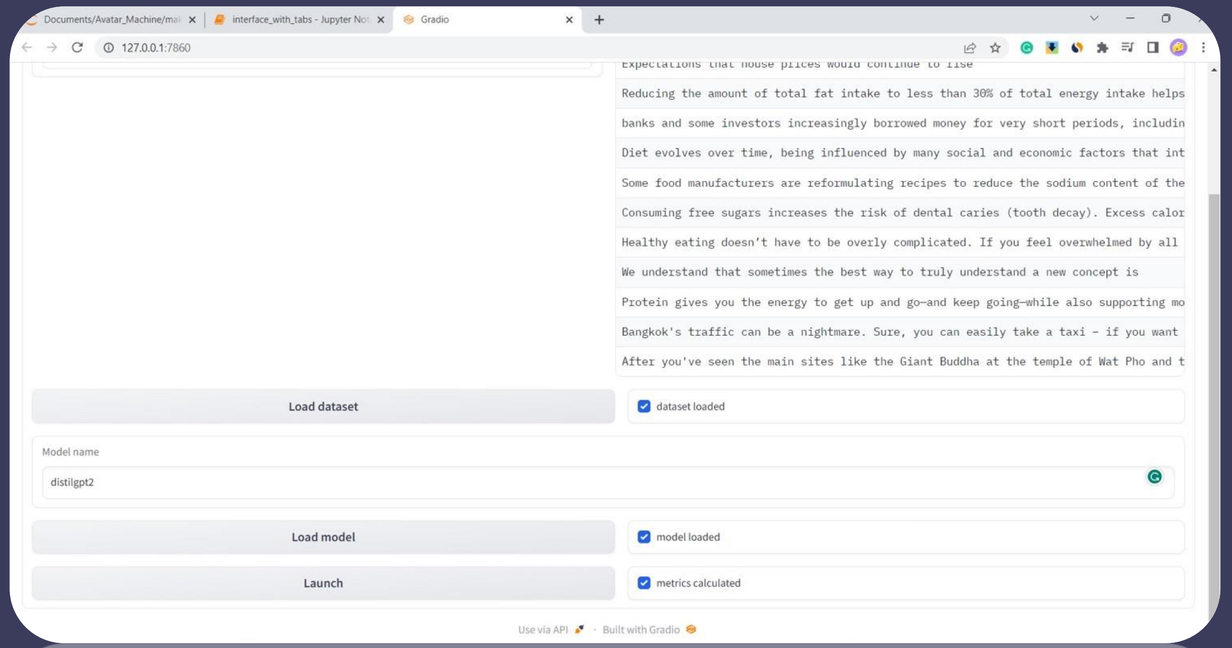
Checking the work of the interpretation

1. **Specified the model being tested**
2. **Created an interpreter**
3. **Described the clusters**
4. **We run the interpretation of the model: input the seed (question) and the result of the generation**
5. **Cluster visualization**

Explanation: the importance of the Disease cluster is 0.76, and the Animals cluster is 0.56.
The model "looked" at those clusters (thematic) during generation, which it would be logical to "look at" when answering the question about an animal's disease.

Example of use

Interface



Demo of the model test interface in the ExplainitAll library

Fiches:

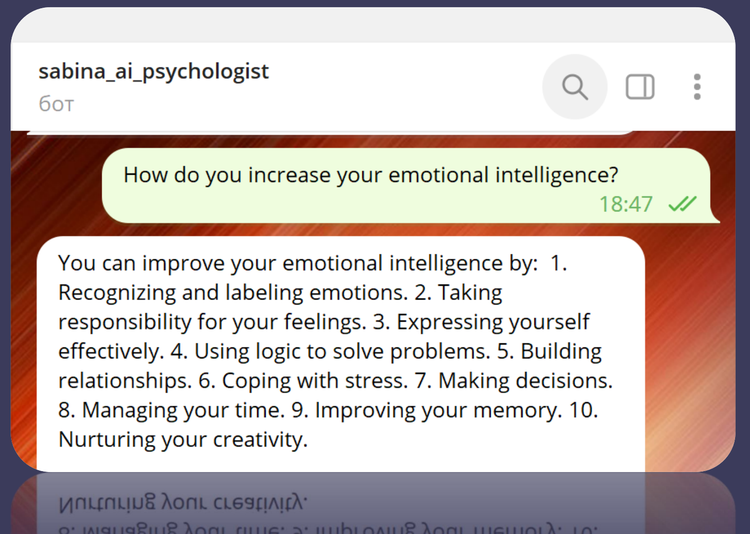
1. **Loads of different transformers**
2. **Loading your dataset**
3. **Testing metrics: ROUGE, PPL, our reliability metric**
4. **Works locally and in the cloud**
5. **Popular Gradio Library**

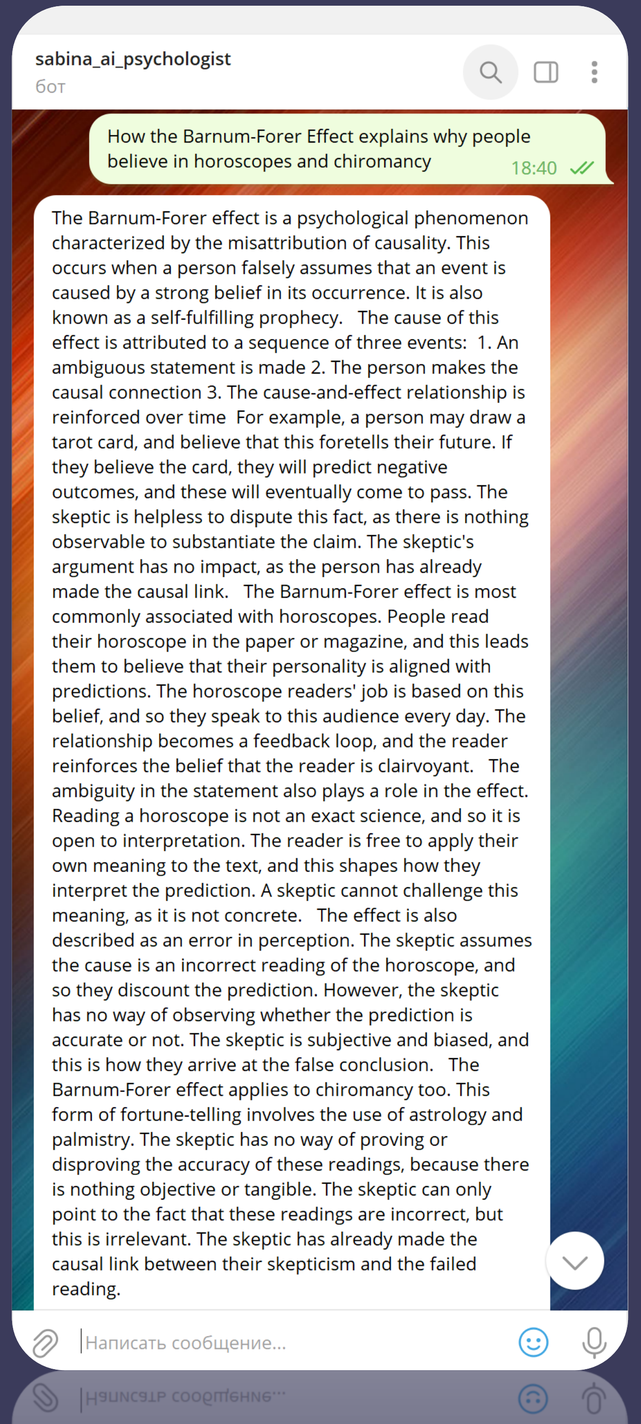
https://gradio.app/

Interpretable answers of QA systems

Explains:
1. Facts
2. Clusters of facts
3. NERs
4. NEL
5. MNLI task for sentences
6. Embeddings

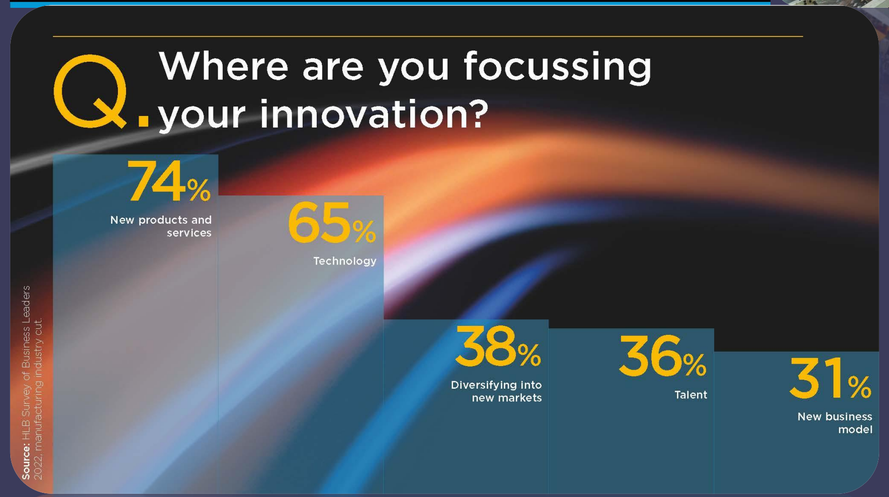
Which semantic clusters does the model look at and which should not have been looked at when answering?





Critical QA systems

Importance for critical industries



Explainable and controllable, reliable QA systems for industries:

1. **Pharmacy: QA systems to aid in diagnosis, contraindications analysis, personalized medicine.**
2. **Manufacture: chain-of-thought in manufacturing decision-making**
3. **Science: Analysis of Scientific Publications, Trends, Fakes, and Biases (Galactica)**

Perspectives. Hallucinations, generation control

ExplainitAll will be available opensource. Free, open source.

Solving the problem of trust in neural network models.

Skills.

Memory

Transformers modifications, hint model

Ability to remember facts, have short and long term memory when communicating

Emotions. Humor.

Explainable systems with ExplainableAi

Generative answers to difficult questions, read comprehension, ODQA, closed book QA

Products.

1. Robotics

Models explaining in words the behavior of complex systems (mechatronics)

2. Question-and-answer system

ExplainableAi explanatory systems. Any field: psychology, medicine, manufacturing, law

3. Text generator, chatbots

You set the keywords and the neural network writes a text with a guarantee of mentioning these words. Scope of creativity

Команда

A team of AI specialists and programmers with 5+ years experience, 10+ successfully completed projects. We have strong competences in Conversational Ai, NLU, use the most advanced Transformer neural networks.
Lead our own RnD, completed FSI Start 1, received a grant on the CodeAi (2022)

Developed Fast Experts Tuning, AIFramework AI_Free, Description Generator, Sabina Ai chatbot, speak at conferences and hackathons

Participation:
Ai Journey(2020), DataStart(2019), Conversations.ai(2021), OpenTalks.ai(2022), AGIconf 19 aug 2022, (INLP: AGI-22 | Interpretable Natural Language Processing (INLP)) Workshop (Seattle, 2022). SberCloud partners


РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2021681469

Интеллектуальная система скиллов и стилей цифровой личности на базе современных архитектур нейронных сетей Transformer с возможностью смыслового ранжирования сгенерированных ответов

Правообладатель: *ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ "АВАТАР МАШИНА" (RU)*

Авторы: *Носко Виктор Ильич (RU), Понимаш Захар Алексеевич (RU)*

Заявка № **2021681037**
Дата поступления **15 декабря 2021 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***22 декабря 2021 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

Zakhar Ponimash
Ai Engineer, author of **AIFramework AI_Free**, 6 years of experience in AI, winner of hackathons

Victor Nosko, CEO of Avatar Machine, NTI expert
5 years of experience in AI. Open AI visionary. Member of ALRII

Alexander Gusarin, 5 years of experience in AI. Data Science engineer and Python developer

Denis Ryabkin, 4 years of experience in AI. Data Science engineer and Python developer