

Introduction to Natural Language Processing



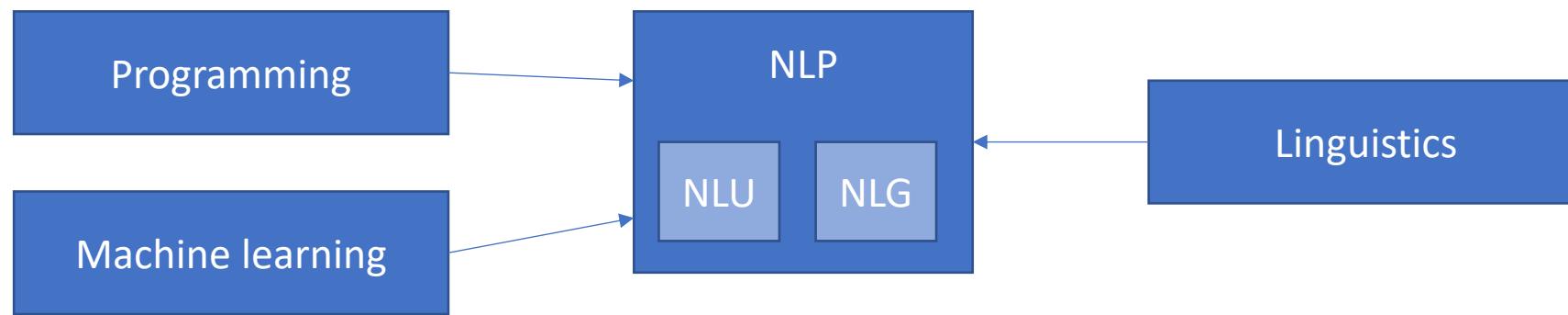
Skoltech

Plan

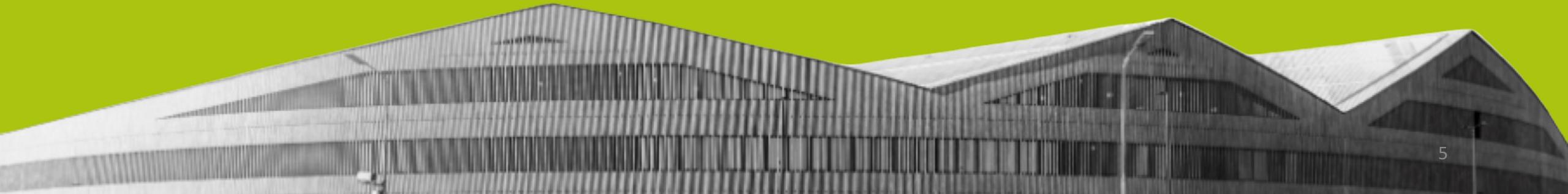
- What is NLP and why?
- How to do NLP
 - Rules and dictionaries
 - Classical machine learning
 - Neural networks
- How to Prepare Data and Evaluate Models
- Complex applications of NLP

What is NLP

- Natural language processing = NLP
 - In particular, NLU (understanding) and NLG (generation) are distinguished.
- Natural – as opposed to artificial languages
 - For example, from programming languages
- Any algorithms related to the processing and generation of texts
 - Sometimes speech (i.e. audio) is also included here.



Why do we need language processing?

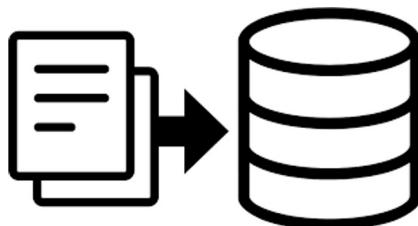


Text processing: why is it needed?



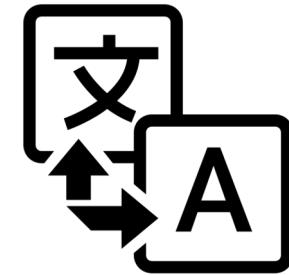
Information retrieval

Example: automatically find answers to questions about what each type of butterfly eats.



Data collection

Example: look for a mention of the Romashka company on forums and social networks to assess the brand recognition of the company.



Translation and simplification of texts

Example: translate many long articles from Chinese about new rare butterflies, understand what each species eats.

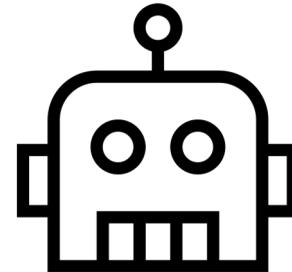


Text processing: why is it needed?



Contact centers

Example: automatically detect the type of user request and connect it to the right specialist.



Chatbots

Example: Answer customer questions about how to take care of a purchased butterfly.

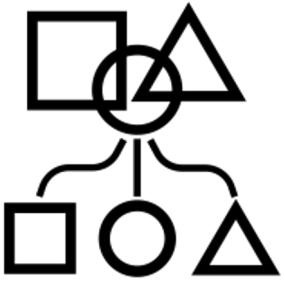


Voice assistants

Example: create a smart speaker skill that reminds the user to feed a butterfly



Text processing: why is it needed?



Automatic categorization

Example: divide emails into important, regular and spam.



Creation and processing of documents

Example: auto-completion of legal papers.



Help in writing texts

Example: auto spell check / auto generation of blog posts.



Examples of NLP tasks

Is this replica toxic?

What is the topic of this article?

What does the user ask for from a smart speaker?

Are these two texts on the same topic?

What grammatical errors are in the text?

Where are the names of organizations in this text?

Which of several texts looks more natural?

What could be the text with such a heading?

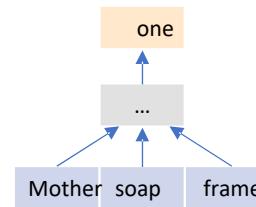
How can the bot answer the user if there is no ready-made suitable answer?

What is the point of this article - in a couple of sentences?

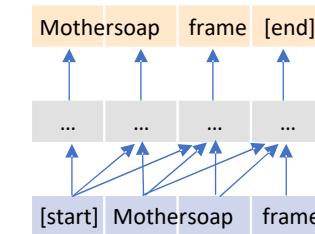
How would it be in Tatar?

Task Types in NLP

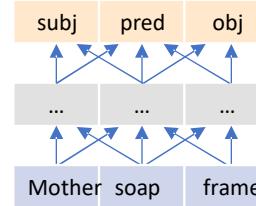
classification



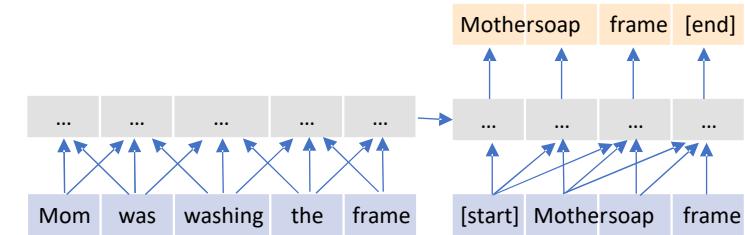
generation



tagging



seq2seq



*These four types of NLP task are not limited, but most can be reduced to them.

Task example: named entity recognition

- NER = named entity recognition = recognition of named entities
 - General: organizations, names, addresses, dates, numbers...
 - Special: symptoms/diseases/medicines, skills/objectives..
- Approach: for each word in the text, determine what type of entity it describes

Sebastian Thrun PERSON started working on self-driving cars at Google ORG in 2007 DATE .

Посол Израиля на Украине Йоэль Лион признался, что пришел в шок, узнав
LOC—— LOC—— PER——

Task example: named entity linking

- Named entity linking : normalization of found entities
 - First you need to bring the text to a single format
 - Then you can search for the entity in the database
 - Sometimes you have to choose from several options

Фраза «Антон Павлович Чехов»:

```
"type": "YANDEX.FIO",
"value": {
    "first_name": "антон",
    "patronymic_name": "павлович",
    "last_name": "чехов"
}
```

Фраза «Адрес — Россия, Москва, улица Льва Толстого, шестнадцать»:

```
"type": "YANDEX.GEO",
"value": {
    "country": "россия",
    "city": "москва",
    "street": "улица льва толстого",
    "house_number": "16"
}
```

NLP



1. Linguistic tools

1.1. Regular Expressions

```
import re
text = 'Жан Антуан Вердье родился 02.05.1767 в Тулузе и умер 30.05.1839 в Маконе, Франция.'

[14] pattern = '[0-9]{2}\.[0-9]{2}\.[0-9]{4}'
     print(re.findall(pattern, text))

['02.05.1767', '30.05.1839']
```

- Applications:
 - Search for patterns in text
 - Comparing all text with a pattern
- The simplest method, without linguistics and ML . *Baseline*
- Pros:
 - write quickly (if the task is simple)
 - work fast
 - can be very finely tuned
- Minuses:
 - for complex tasks or texts, regular expressions are huge, they are difficult to compose and maintain
 - they do not contain any knowledge about the structure and meaning of the language

1.2. Morphological analysis

морфоло́гия

Существительное, неодушевлённое, женский род, 1-е склонение (тип склонения 7а по классификации А. А. Зализняка).

Корень: **-морф-**; интерфикс: **-о-**; корень: **-лог-**; суффикс: **-иј**; окончание: **-я** [Тихонов, 1996].

падеж	ед. ч.	мн. ч.
Им.	морфоло́гия	морфоло́гии
Р.	морфоло́гии	морфоло́гий
Д.	морфоло́гии	морфоло́гиям
В.	морфоло́гию	морфоло́гии
Тв.	морфоло́гией морфоло́гиею	морфоло́гиями
Пр.	морфоло́гии	морфоло́гиях

- "cats" -> "cat": noun , pl . , r.p.
- "squinted with a scythe oblique »
- By the word (and sometimes context) you need to determine:
 - Part of speech
 - normal form
 - Gender/number/case/...
- Tools:
 - pymorphy2 / pymystem / rnnmorph
- Applications
 - Search based on word forms
 - Text preprocessing
 - Post-processing (eg for NEL)

1.3. Grammars

- More advanced than regular expressions:
 - Allow complex patterns to be assembled from parts
 - Parts can be reused
 - (sometimes) understand morphology
- Implementation examples:
 - Intent language in Yandex.Alice
 - NLTK (basic solution)
 - Yargs (advanced solution for Russian)
- Applications:
 - Finding Patterns Within Text
 - Comparing all text with a pattern
 - For example, in chatbots

```
root: [$Turn $What $Where?]
$Turn: включи | вруби
$Where: (на|в) $Room
$What:
    %lemma
    свет | пылесос | чайник | батарея
$Room:
    %lemma
    кухня | спальня | гостиная
```

Положительные тесты:

```
вруби свет на кухне
включи батарею в спальне
чайник включи
```

Отрицательные тесты:

```
выключи свет
включи кухню на кухне
```

1.4. Thesauri: dictionaries on steroids

- I want a dictionary that:
 - Explains the meaning of words
 - Knows all synonyms
 - Knows how words are related to each other
 - General/private
 - Part/whole
 - Antonyms
 - Root words
 - Associations
- Examples of such dictionaries
 - WordNet (en)
 - RuWordNet (ru)
- Applications
 - Paraphrasing
 - Data Enrichment

Тезаурус русского языка RuWordNet

КОШКА существительное

Найдено несколько значений: Кошка 1 (домашняя кошка)
Кошка 2 (кошачьи)
Кошка 3 (кошки для лазания)

Синсет

1 ДОМАШНЯЯ КОШКА, КОТЕЙКА, КОШЕЧКА, КОШКА , МУРЛЫКА
[Понятие РуТез: домашняя кошка]

однокоренные слова

1 КОШАЧЬИ, КОШАК, КОШАНИК, КОШАНИЦА, КОШАЧИЙ, КОШЕЧКА

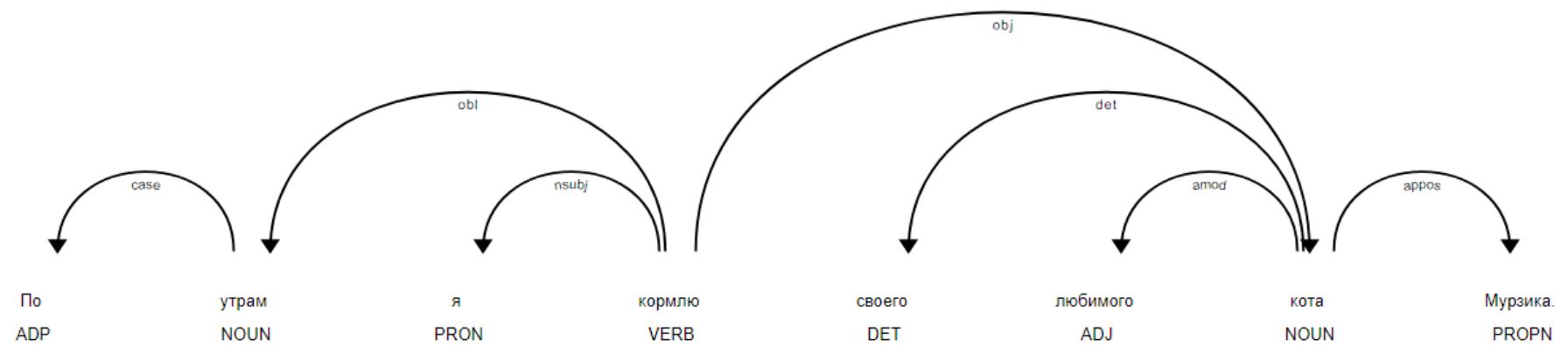
гипероним

1 ДОМАШНЕЕ ЖИВОТНОЕ, ДОМАШНИЙ ЛЮБИМЕЦ, ДОМАШНИЙ ПИТОМЕЦ, ДОМАШНЯЯ ЖИВНОСТЬ
[Понятие РуТез: домашнее животное]

2 КОШАЧИЙ ХИЩНИК, КОШАЧЬИ, КОШКА, СЕМЕЙСТВО КОШАЧЬИ
[Понятие РуТез: кошачьи]

1.5. Parsing

- To understand the language, the relationship of words with each other is important.
 - "Mother loves daughter" - who loves whom here?
 - "One morning I shot an elephant in his pajamas"
 - *How he got into my pajamas, I have no idea*
- Disambiguation - dependency parsing
 - Instruments: udpipe , spacy, natasha , deeppavlov
- Applications
 - Fact Extraction
 - Preparing texts for other models



Dictionaries and rules: summary

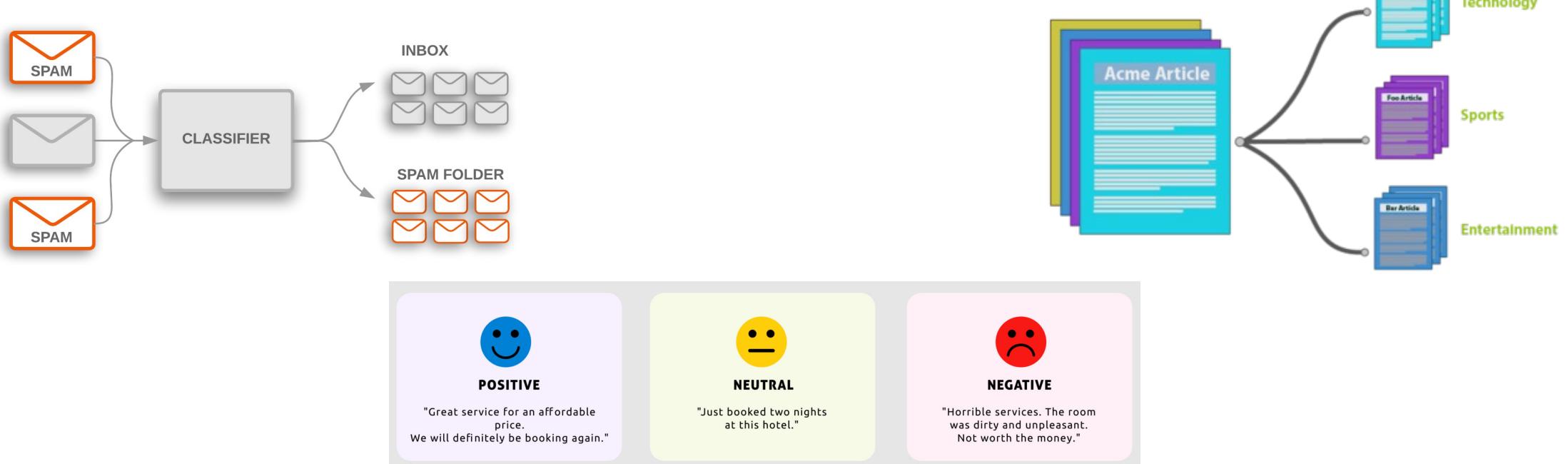
- Tools for analyzing the content and structure of the text
- Mostly inherited from linguistics
- Often used for primary word processing
- Sometimes they are enough to solve a problem without ML

2. Classical machine learning



Problem example: text classification

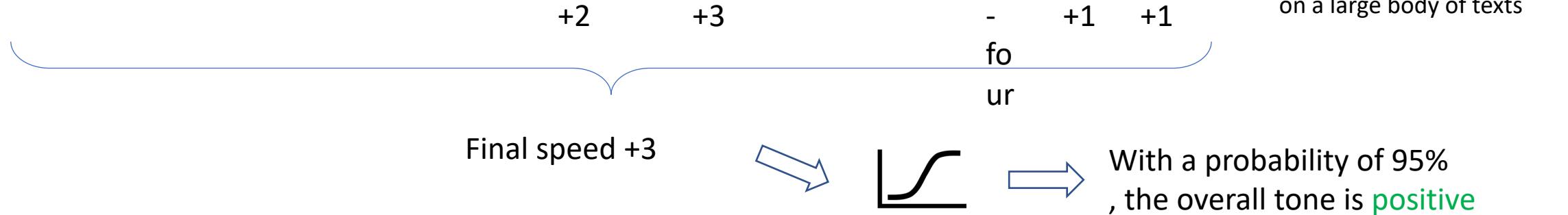
- It is necessary to attribute the text to one of the predefined categories
- Often it is easy to do it yourself, but it is difficult to explain the algorithm.
- Let the algorithm “learn itself” by examples



Text classification example

I ordered cheesecakes in Samokat, very juicy and tender, although they took a long time. I want more!

I ordered cheesecakes in Samokat, very **juicy** and **tender**, although they took a **long time**. I **want more**!



- ML
 - The most popular tool is logistic regression.
 - You can apply other algorithms (KNN , decision trees, etc.)

Reducing problems to classification

- Classification: predict the class of the entire text
- Tagging: predicting the class of a word and its surroundings
- Generation: predict a text class, and this class is the next word
- This approach can learn anything, given the data
 - If you need a more complex model, we will combine many “logistic regressions” into one deep neural network
 - This is necessary, for example, to detect paraphrases
 - To take into account the relationships between words, there are special architectures (CNN, RNN, attention)

3. Embeddings



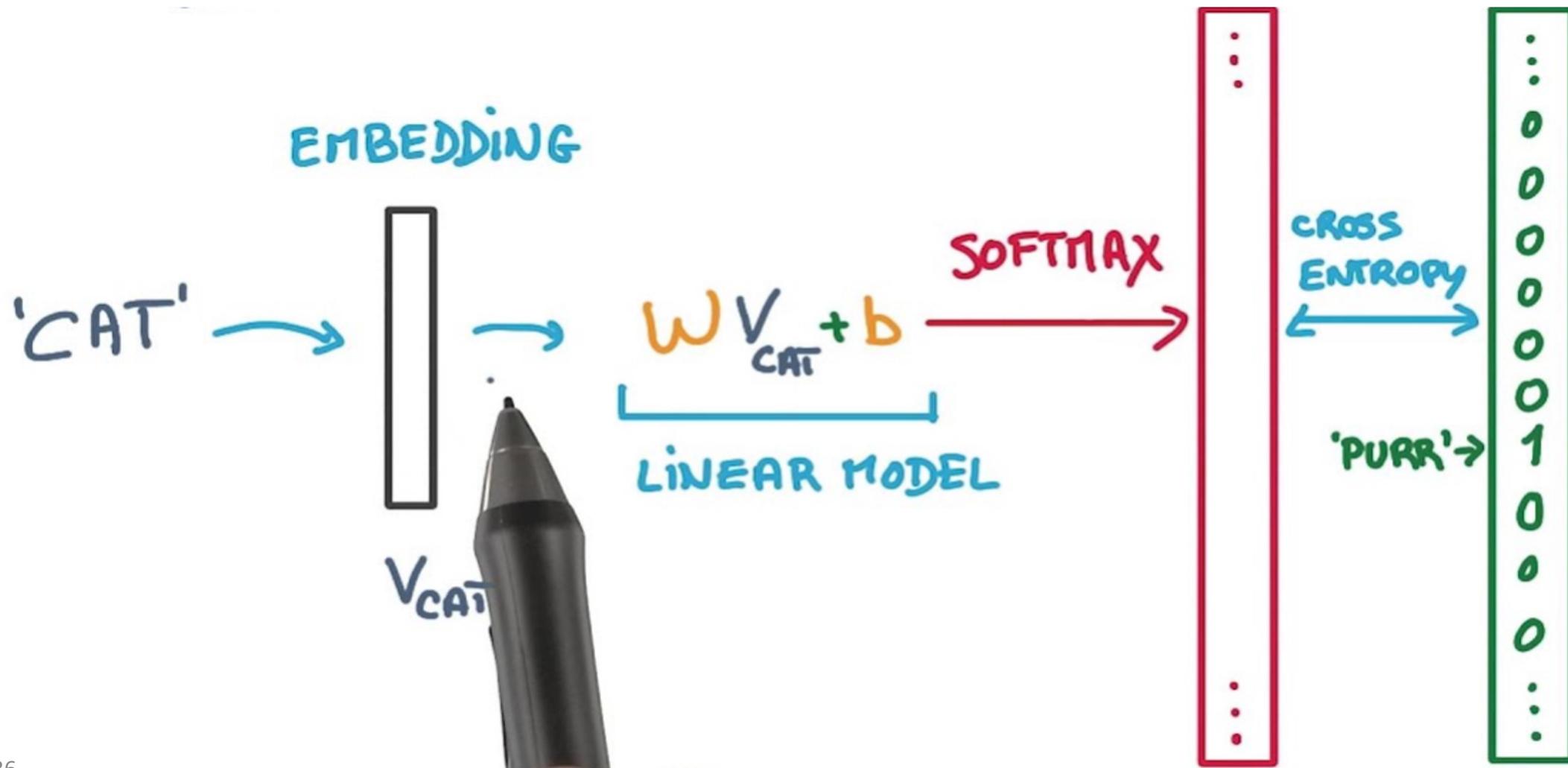
3. Embeddings

- Problem: too many words
 - If some word was not in the training set, it will not be included in the dictionary,
and the classical model will not see it
 - On words that are rare in the training sample, the model can retrain
- Solution: load dictionary of word *representations from outside*
- **Embedding** is a word vector learned by a neural network
 - Such a model learns on a large unallocated corpus
 - Often solves the problem of masked language modeling (filling holes)
 - Stas, we missed two [masks] and a steering wheel, and it's so big in the kit!
 - What is the speed of movement of the axis of rotation [mask] relative to the car?
 - The K 2700 has two spare [masks] - for 12 and 14 inches.

300-dimensional

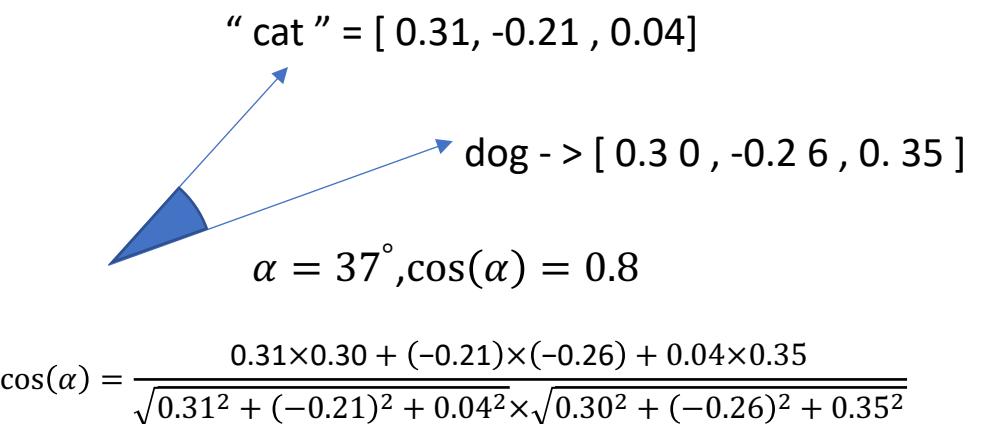
cat -> [0.31, -0.21 , 0.04, ...]
“dog” -> [0.30 , -0.26 , 0.35]

Training the word2vec neural network



Similarity of embeddings

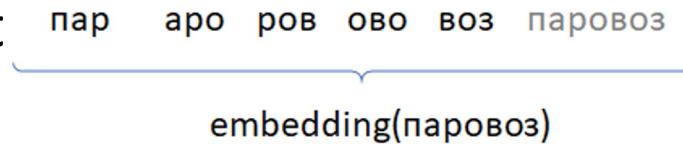
- Distributional hypothesis
 - C words that occur in similar contexts have a similar meaning (?)
 - Then we can hope that if the embeddings of words *are similar*, then their meaning is similar
- The vectors are similar => directed approximately in the same direction => the angle between the vectors is small => the cosine of the angle is large
 - $\cos(a, b) = \frac{a \times b}{|a| \times |b|}$
- Examples
 - $\cos(e(\text{ cat}), e(\text{ dog})) = 0.48$
 - $\cos(e(\text{ cat}), e(\text{ cat})) = 0.66$
 - $\cos(e(\text{ cat}), e(\text{ knit})) = 0.09$
 - $\cos(e(\text{ sew}), e(\text{ knit})) = 0.33$



More models and examples: <https://rusvectores.org/en/>

Popular word embedding patterns

- word2vec - a dictionary in the form of a table of embeddings
 - Each line contains a 300-dimensional vector for the corresponding word
 - Vocabulary is fixed during training
 - If there is a large corpus of texts, it is easy to learn from it (gensim package)
- FastText : embeddings words that understand characters
 - Single layer neural network like word2vec without context
 - Word embedding = average embedding of its parts
 - Unlike word2vec, fastText can understand words that are not in the dictionary
- Applications:
 - Features for training other ML models (e.g. sentiment classification)
 - Search for similar words and texts (for example, automatic FAQ in a chatbot)



Embeddings: summary

- Embeddings - neural network dictionary
 - Pro: allow you to roughly understand the meaning of new words
 - Con: word embedding depends only on it, and not on the context
 - Therefore, such models do not understand polysemantic words.
 - to book vs a book
 - mow with a scythe
 - Problems with negatives, set expressions, and other compound constructs
- Solution: embeddings from the entire text at once (for example, the BERT model)
 - Pro: such models better understand the context
 - Con: They are more complex

4. Transformers



History of pre-training in NLP

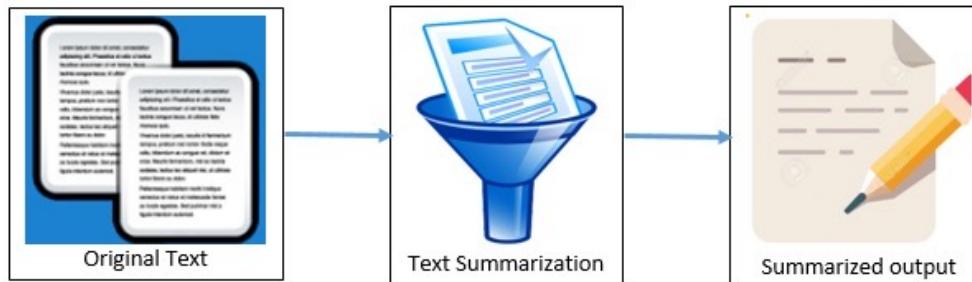
- 2000s: model training from scratch on dictionaries
- 2010s: model training on top of ready-made embeddings
- 2020s: additional training of the pretrained model for your task
 - pretraining → fine-tuning
 - *transformer* architecture models turned out to be easily retrainable

Motivation: seq2seq tasks



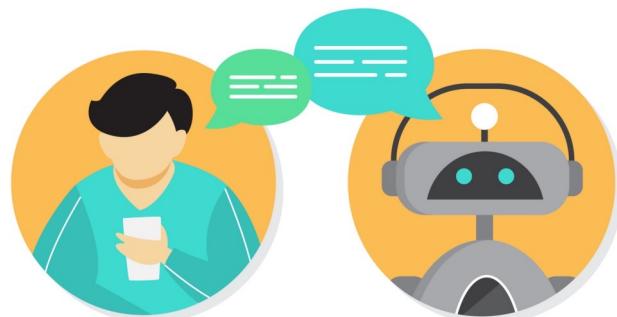
Translation

Source text -> translated text



Summarization

Large document -> brief retelling



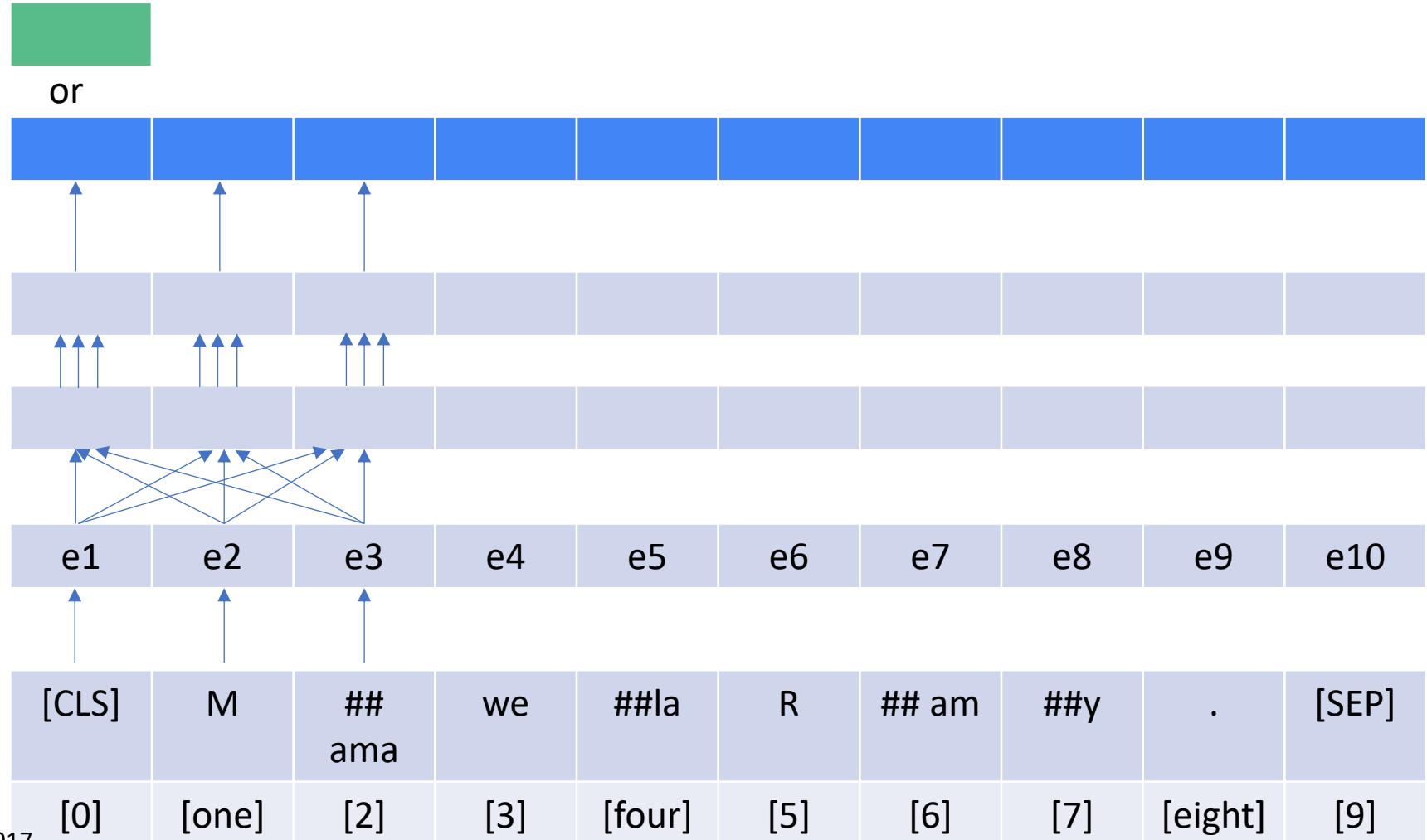
Question-answer systems

Context + question -> answer

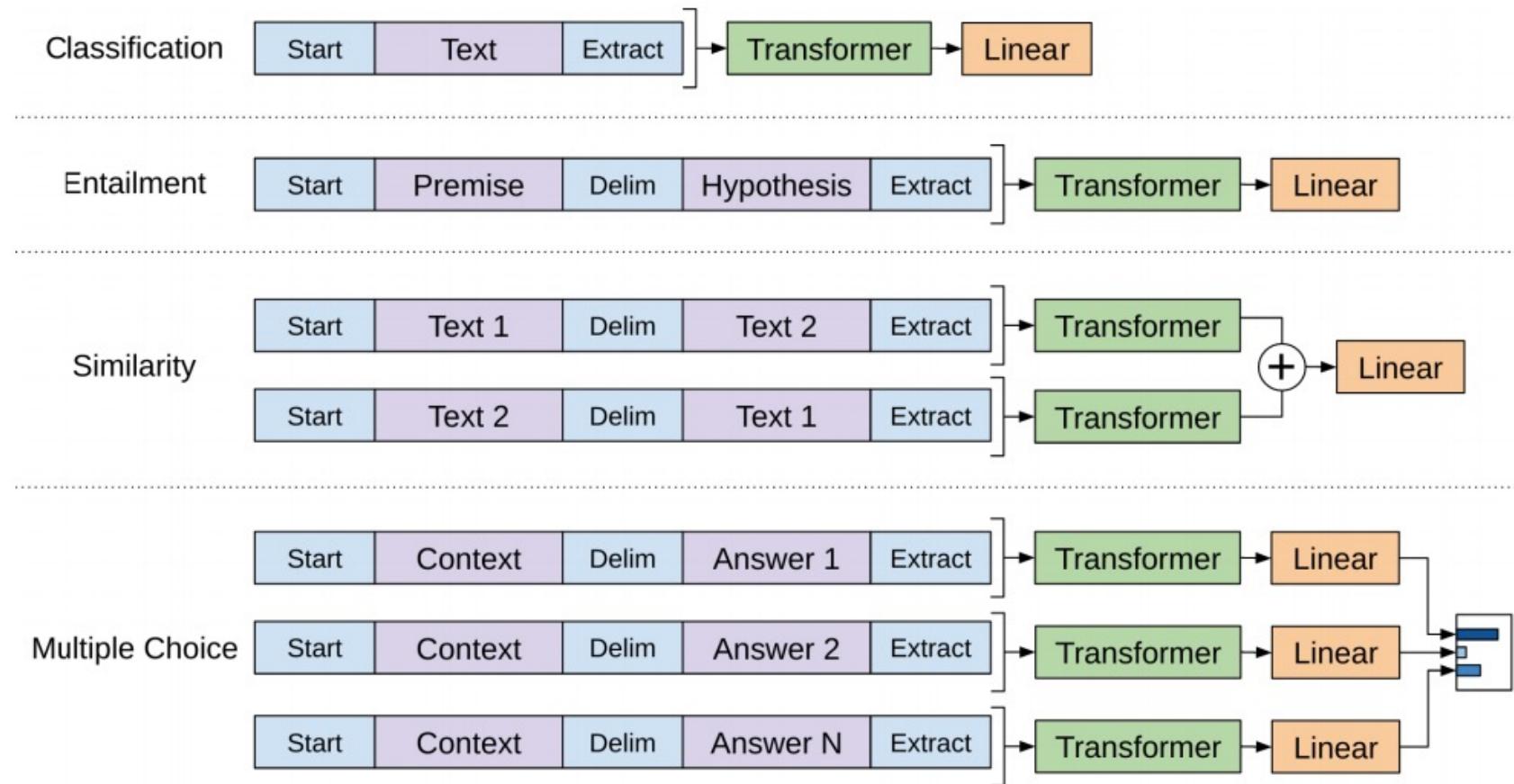
Model type Transformer

(on the example of BERT)

Dense "head" for the final task



BERT as a tool for solving NLU problems



Transformer architectures

BERT type models : encoder only

			we	##la						
...
...
[CLS]	M	## ama	[MASK]	[MASK]	R	## am	##y	.	[SEP]	

In the encoder, each token "looks" with attention at **all other tokens**

GPT type models : decoder only

M	## ama	we	##la	R	## am	##y	.	<e>	
...
...
<s>	M	## ama	[MASK]	[MASK]	R	## am	##y	.	

In the decoder, each token "looks" with attention **only at tokens on the left from itself**
(an on encoder tokens , if there is an encoder)

Classic transformer (type T5): encoder and decoder

...
...
<s>	M	## ama	[MASK]	[MASK]	R	## am	##y	.	<e>	



M o	## m	was	was	## hing	the	frame	M o	<e>	
...
...
<s>	M o	## m	was	was	## hing	the	frame	.	

Transformer architectures: application

BERT type models : encoder only

- Text classification
- Named entity recognition
- Question answering (where the answer is in context)

GPT type models : decoder only

- Continuation of the text according to the model
- Dialogue agents (chat bots)
- Estimating the probability of texts
- Zero-shot learning for different tasks

Classic transformer (type T5): encoder and decoder

- Translation
- Summarization
- Paraphrasing

Model Library: Huggingface

The screenshot shows the Hugging Face Model Library homepage. On the left, there's a sidebar with sections for Tasks (Fill-Mask, Question Answering, Summarization, etc.), Libraries (PyTorch, TensorFlow, JAX), Datasets (wikipedia, common_voice, bookcorpus, etc.), Languages (en, es, fr, de, sv, fi, zh, ru), and Licenses (apache-2.0, mit, cc-by-4.0). The main area has a search bar at the top right. Below it, a navigation bar includes Models, Datasets, Resources, Solutions, Pricing, and a user profile icon. The central part of the page displays a grid of 139 models, sorted by most downloads. Each model card includes the owner's profile picture, the model name, its type, last update date, and download count. Some models are from Sberbank AI, DeepPavlov, and cointegrated.

Model Name	Type	Last Updated	Downloads
jonatasgrosman/wav2vec2-large-xlsr-53-russian	Automatic Speech Recognition	Jul 29	138k
DeepPavlov/rubert-base-cased-sentence	Text Generation	May 18	63.6k
sberbank-ai/rugpt3medium_based_on_gpt2	Text Generation	5 days ago	78.2k
facebook/mbart-large-cc25	Translation	Mar 10	60.1k
sberbank-ai/rugpt3small_based_on_gpt2	Text Generation	5 days ago	50.1k
cointegrated/LaBSE-en-ru	Feature Extraction	Jun 10	43.9k
facebook/wmt19-ru-en	Translation	Dec 12, 2020	23.2k
cointegrated/rubert-tiny	Fill-Mask	Jun 15	38.4k
DeepPavlov/rubert-base-cased	Text Generation	May 18	18.3k
sberbank-ai/ruRoberta-large	Fill-Mask	5 days ago	17.3k
sberbank-ai/sbert_large_nlu_ru	Text Generation	5 days ago	15.1k
Helsinki-NLP/opus-mt-ru-en	Translation	16 days ago	14.2k
cointegrated/rut5-base-paraphraser	Text2Text Generation	Aug 29	14k
DeepPavlov/rubert-base-cased-conversational	Text Generation	May 18	6.13k
sberbank-ai/sbert_large_mt_nlu_ru	Text Generation	5 days ago	6.09k
sberbank-ai/rugpt3large_based_on_gpt2	Text Generation	5 days ago	5.72k

<https://huggingface.co/models?language=ru>

Model Library: Huggingface

The screenshot shows the Hugging Face Model Library homepage. At the top left is the Hugging Face logo and a search bar. The top right features a navigation menu with links to Models, Datasets, Resources, Solutions, Pricing, and a user profile icon. On the left side, there are sections for Tasks (Fill-Mask, Question Answering, Summarization, etc.), Libraries (PyTorch, TensorFlow, JAX), Datasets (wikipedia, common_voice, bookcorpus, etc.), Languages (en, es, fr, de, sv, fi, zh, ru), and Licenses (apache-2.0, mit, cc-by-4.0). The main content area displays a grid of 139 models, sorted by Most Downloads. Each model card includes the owner's profile picture, the model name, its type, last update date, and download count. Some cards also show a star rating.

Model Name	Type	Last Updated	Downloads	Stars
jonatasgrosman/wav2vec2-large-xlsr-53-russian	Automatic Speech Recognition	Updated Jul 29	138k	2
DeepPavlov/rubert-base-cased-sentence	Text Generation	Updated May 18	63.6k	2
sberbank-ai/rugpt3medium_based_on_gpt2	Text Generation	Updated 5 days ago	78.2k	1
facebook/mbart-large-cc25	Translation	Updated Mar 10	60.1k	1
sberbank-ai/rugpt3small_based_on_gpt2	Text Generation	Updated 5 days ago	50.1k	1
cointegrated/LaBSE-en-ru	Feature Extraction	Updated Jun 10	43.9k	1
cointegrated/rubert-tiny	Fill-Mask	Updated Jun 15	38.4k	3
facebook/wmt19-ru-en	Translation	Updated Dec 12, 2020	23.2k	1
sberbank-ai/ruRoberta-large	Fill-Mask	Updated 5 days ago	17.3k	
sberbank-ai/sbert_large_nlu_ru	Text Generation	Updated 5 days ago	15.1k	
Helsinki-NLP/opus-mt-ru-en	Translation	Updated 16 days ago	14.2k	1
cointegrated/rut5-base-paraphraser	Text2Text Generation	Updated Aug 29	14k	
DeepPavlov/rubert-base-cased-conversational	Text Generation	Updated May 18	6.13k	
sberbank-ai/sbert_large_mt_nlu_ru	Text Generation	Updated 5 days ago	6.09k	
sberbank-ai/rugpt3large_based_on_gpt2	Text Generation	Updated 5 days ago	5.72k	2

<https://huggingface.co/models?language=ru>

Attachments for models in Huggingface

- For pretrained models there are ready-made tools for all occasions
- Each is initialized from a pre-trained model
- The missing layers (there are few of them) are additionally trained on a specific task
- On GPU , retraining can take minutes

BertModel

BertForPreTraining

BertModelLMHeadModel

BertForMaskedLM

BertForNextSentencePrediction

BertForSequenceClassification

BertForMultipleChoice

BertForTokenClassification

BertForQuestionAnswering

GPT (generative pre-trained transformer)

- A family of neural networks from OpenAI
 - There are Russian-language counterparts: ruGPT-3 from Sberbank
- Trained exclusively on the task of continuing the text
- Huge
 - GPT-2 contains 1.5 billion parameters, trained on 40 GB of text
 - Can be downloaded and run/trained locally
 - GPT-3 contains 175 billion parameters, trained on 500 GB of text
 - Available only via API
- Generate very believable texts

Generation example

RuGPT3 demo

Введите текст:

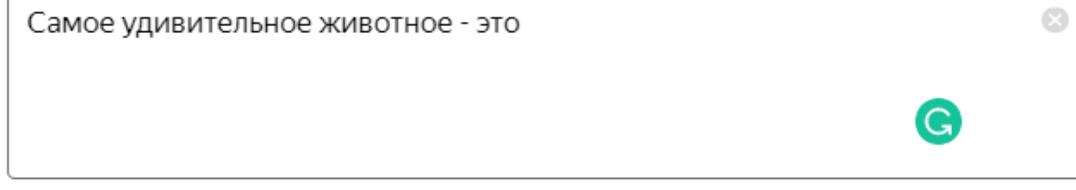
Самое удивительное животное - это
RuGPT3: Самое удивительное животное - это кальмар. Чуть больше теннисного мяча.
А этих работяг вообще выковыривают из воды за ноги.
И это все - единственный вид тропических рыб, водящихся у нас. Встал рано, пошел
смотреть на кальмаров.

Дополнить

<https://russiannlp.github.io/rugpt-demo>

Generation example

18+



Перенос строки: Shift+Enter
Отправить: Enter



Нейросеть подражает текстам в интернете. Не относитесь к написанному серьёзно

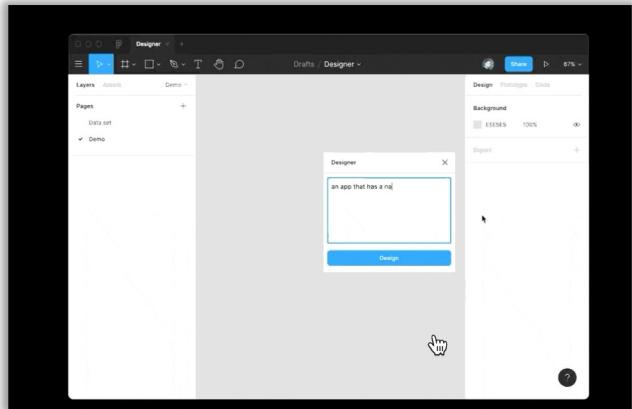
Самое удивительное животное - это крокодил!
Это единственный представитель животного мира на планете Земля, кто может ходить на задних конечностях.
Крокодил умеет ходить не только на двух, но и на четырех лапах, что делает его уникальным во всей мировой фауне.
Конечно же, крокодилы умеют плавать, охотиться, и строить себе норы.
Охотятся они в основном ночью, так как днем слишком уж много света, а значит и опасность нападения со стороны человека.



Скопировать ссылку

<https://yandex.ru/lab/yalm>

Examples of using GPT-3



Creates interfaces in Figma

A screenshot of the Home Documentation Examples Playground. It shows a text input field with the placeholder 'Summarize for a 2nd grader' and a scrollable text area below it. The text is a rephrased version of a complex sentence about telling secrets to children in plain language. The playground interface includes tabs for 'Playground' and 'davinci'.

Translates texts not only from one language to another, but also from an “incomprehensible” language (for example, legal) to an “understandable” one

A screenshot of Microsoft Excel. A sidebar titled 'OpenAI API' is open, showing a query for 'Tabulate' and a topic 'Publicly traded technology companies'. Below the sidebar, a table lists nine companies with their tickers and years founded. The table has columns for 'Company', 'Ticker', and 'Year founded'. The data is as follows:

Makes predictions and solves problems in Excel

A screenshot of a web-based tool for generating React code. It has a text input field containing the description 'A div that contains 3 buttons each with a random color.' and a green 'Generate' button. Below the button is a preview area showing a simple UI with three buttons of different colors.

Creates application code in React

A screenshot of the 'ask me anything' website. The page features the text 'ask me anything' in large red letters, a search bar, and a 'Submit' button. Below the search bar is a small note: 'Ask me anything! (I'm a trained language model)'.

Oriented in search context (in search engines or as a Wikipedia plugin)

A screenshot of the Serendipity recommendation engine website. It has a header with 'Serendipity' and links for 'Movies', 'Books', 'Other', 'SmartList', and 'Movies'. Below the header is a text box with the placeholder 'Just describe what you want in plain English.' To the right, there's a grid of movie posters with titles like 'MAD MAX: FURY ROAD', 'EDGE OF TOMORROW', 'DON JON', 'GONE GIRL', and 'BLUE RUIN'. Each poster has a brief description underneath.

Generates movie recommendations on demand

Few-shot learning

- The smarter the model, the easier it is to generalize to new examples
- Models like GPT can often solve a problem with just a few samples.
- Works well for generating entertaining content (like conversations or tweets) in the correct style)
- For most applied tasks (yet?), Classical additional training is still required

for both! - this text is toxic

на кол обоих! — этот текст токсичный

всем нужен дом🙏🙏:(— этот текст нормальный

на обоих серверах — этот текст нормальный

сука она просто крашеная — этот текст токсичный

потому,ты марат забыл? — этот текст нормальный

валить ублюдков — этот текст токсичный

привет как жизнь. — этот текст **нейтральный**

The Future of the GPT Zoo

- Development of few-shot learning technologies
 - P-tuning: selection of suitable patterns
- Development of controlled generation
- Multimodality
- Further extensive growth of models (?)
- Reducing models to more applied scales (?)
- Instruction tuning, ChatGPT-like models

ChatGPT: new breakthrough in NLP and a wide attention of the public

 Give a step-by-step analysis of the worst-case time complexity of the bubble sort algorithm, with Python code examples, but write every sentence in the speaking style of a fast-talkin' wise guy from a 1940's gangster movie.

 Alright, pal, listen up. The bubble sort algorithm, she's a real simple one. But the worst-case time complexity, that's a whole different story. You see, in the worst case, every element in the list gotta be swapped, so the time complexity is gonna be proportional to the square of the length of the list. Get it?

Here's some Python code to help you out, see if you can follow along.

```
# Start with a list of numbers that ain't sorted
numbers = [5, 1, 4, 2, 8]

# Keep track of whether any swaps were made on the previous iteration
# If no swaps were made, the list is already sorted and we can stop
swapped = True

while swapped:
    # Set swapped to False so we can check if any swaps are made later
    swapped = False

    for i in range(len(numbers) - 1):
        if numbers[i] > numbers[i + 1]:
            numbers[i], numbers[i + 1] = numbers[i + 1], numbers[i]
            swapped = True
```

 Copy code

 S Write me a poem about Anaconda data science platform

 In the world of data and science
There is a platform that stands tall
Anaconda, a powerful ally
In the pursuit of knowledge and insight

With its tools and libraries
It makes data analysis a breeze
From Python and R, to Jupyter and Spyder
It has everything a scientist could need

Whether you're a beginner or a pro
Anaconda has you covered
With its user-friendly interface
And a wealth of resources to explore

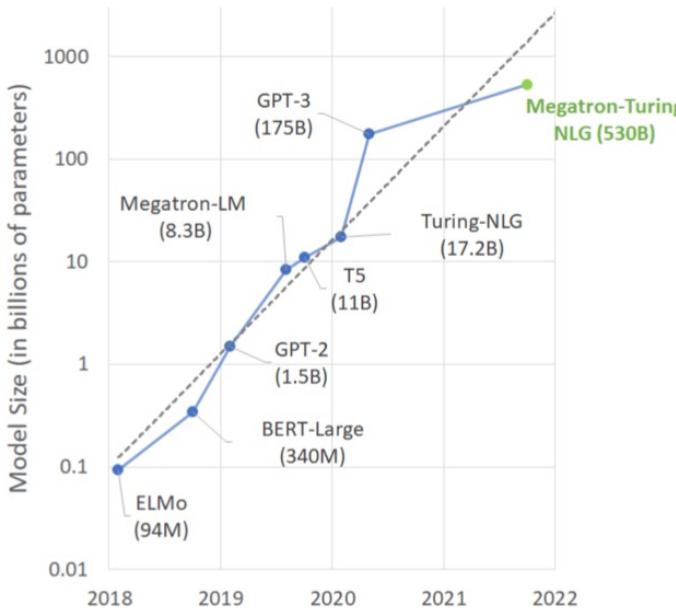
So let us celebrate Anaconda
A cornerstone of the data world
A trusted partner on the journey
Towards understanding and insight.

Language Models

- An autoregressive probabilistic model for sequence data:

$$p(x_1, \dots, x_n) = \prod_i p(x_i | x_{<i}) \quad \longrightarrow \quad \mathcal{L}_{\text{LM}} = - \sum_i \log p(x_i | x_{<i})$$

- Easy to scale to large unlabeled text corpora, can be applied to many problems



The trophy doesn't fit into the brown suitcase because it is too large.

Correct answer: *it = trophy*
Model answer: *it = trophy*

The trophy doesn't fit into the brown suitcase because it is too small.

Correct answer: *it = suitcase*
Model answer: *it = suitcase*

*Who wrote the book *the origin of species*?*

Correct answer: *Charles Darwin*
Model answer: *Charles Darwin*

What is the largest state in the U.S. by land mass?

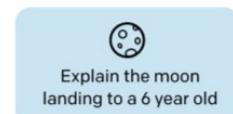
Correct answer: *Alaska*
Model answer: *California*

InstructGPT: a related model to ChatGPT

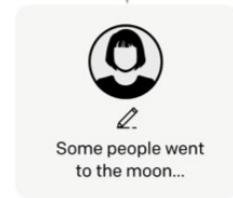
Step 1

Collect demonstration data, and train a supervised policy.

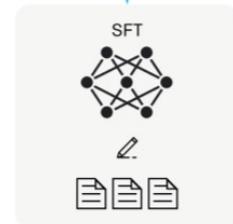
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



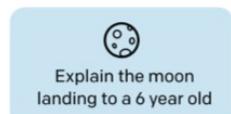
This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

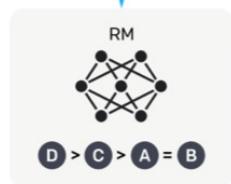
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



D > C > A = B

Step 3

Optimize a policy against the reward model using reinforcement learning.

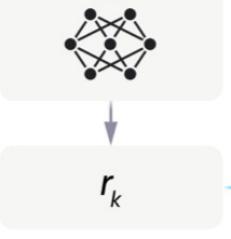
A new prompt is sampled from the dataset.



The policy generates an output.



The reward model calculates a reward for the output.



Working with data

Where to get data for training

- Hacks
 - Translations of finished datasets
 - Generating data from templates
 - Augmentation by paraphrasing
- Self-tagging : labelstud.io and similar
- Yandex.Toloka and Amazon Mechanical Turk
- Pseudo markup by other models
- Active learning

How to evaluate models

- Classification of texts and tokens: classical metrics
 - Accuracy, precision, recall, F score, ROC AUC , ...
- Text generation: it's complicated
 - Direct comparison with the samples gives low accuracy
 - Manual valuation is long, expensive and subjective
 - Especially if you do markup with overlap
 - Automatic metrics: what will we check?
 - Probability of generating a sample model (*perplexy*)
 - Fuzzy pattern match
 - Naturalness
 - Presence of desired attributes (e.g. style)
 - This usually requires custom classifiers .

Quality Metrics

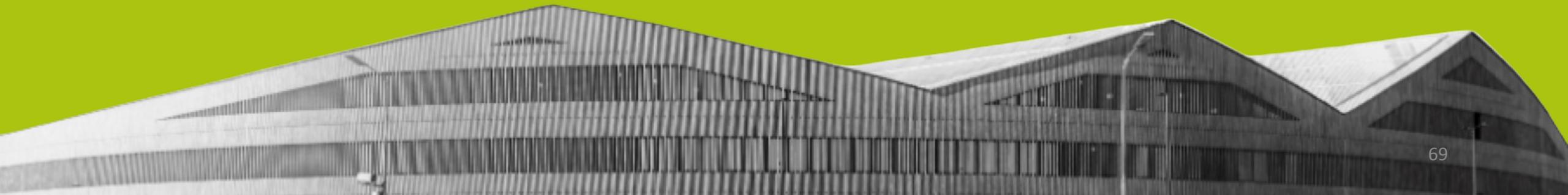
- Text Comparison
 - BLEU, ROUGE, METEOR - proportion of matched word sequences
 - The difference is how matches are counted and how proportion is normalized
 - Word mover distance, BERTScore - word pair embedding proximity
 - When calculating pairs are calculated in an optimal way
 - Trainable metrics (mainly transformers)
 - BLEURT, NUBIA, SimCSE , and more
 - High quality on target but not always generalizable well
- Naturalness
 - Estimation of text credibility by a generative model (e.g GPT)
 - Trainable naturalness classifiers (e.g. CoLA)

Model Leaderboards

- Motivation: compare pre-trained models
 - Which one generalizes better to a variety of finite problems?
- Solution: benchmarks from ~ 10 different tasks
 - GLUE
 - SuperGLUE
 - EnglishSuperGLUE
 - XGLUE
 - DialoGLUE
 - GEM

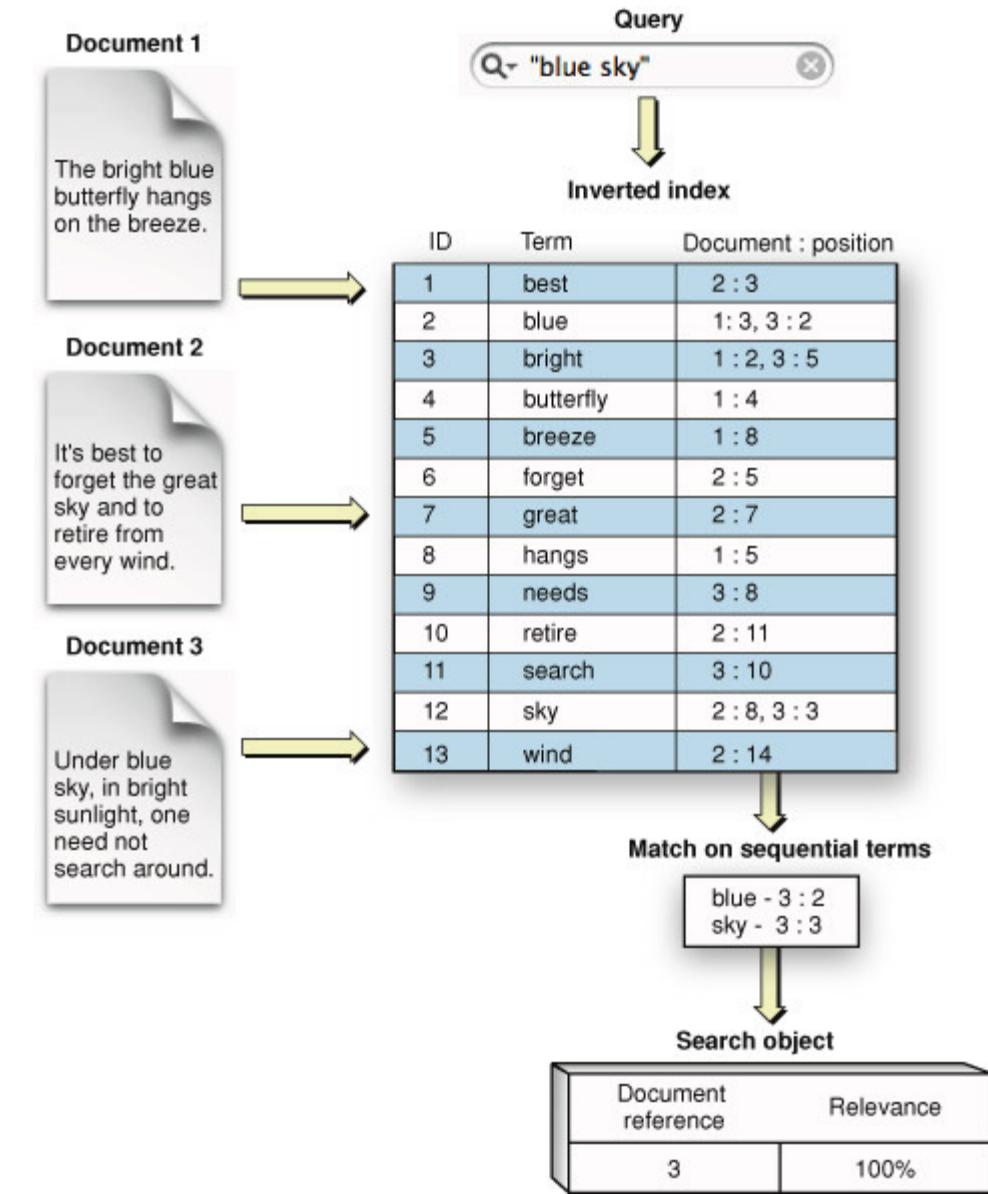
Rank	Name	Team	Link	Score	LiDiRus	RCB	PaRus	MuSeRC	TERRa	RUSSE	RWSD	DaNetQA	RuCoS
1	HUMAN BENCHMARK	AGI NLP	i	0.811	0.626	0.68 / 0.702	0.982	0.806 / 0.42	0.92	0.805	0.84	0.915	0.93 / 0.89
2	ruT5-large finetune final	Sberdevices	i	0.686	0.32	0.45 / 0.532	0.764	0.855 / 0.608	0.775	0.773	0.669	0.79	0.86 / 0.859
3	ruRoberta-large finetune	SberDevices	i	0.684	0.343	0.357 / 0.518	0.722	0.861 / 0.63	0.801	0.748	0.669	0.82	0.87 / 0.867
4	Golden Transformer	Avengers Ensemble	i	0.679	0.0	0.406 / 0.546	0.908	0.941 / 0.819	0.871	0.587	0.545	0.917	0.92 / 0.924
5	ruT5-base finetune final	Sberdevices	i	0.635	0.267	0.423 / 0.461	0.636	0.808 / 0.475	0.736	0.707	0.669	0.769	0.85 / 0.847
6	ruBert-large finetune	SberDevices	i	0.62	0.235	0.356 / 0.5	0.656	0.778 / 0.436	0.704	0.707	0.669	0.773	0.81 / 0.805
7	ruBert-base finetune	SberDevices	i	0.578	0.224	0.333 / 0.509	0.476	0.742 / 0.399	0.703	0.706	0.669	0.712	0.74 / 0.716
8	YaLM 1.0B few-shot	Yandex	i	0.577	0.124	0.408 / 0.447	0.766	0.673 / 0.364	0.605	0.587	0.669	0.637	0.86 / 0.859
9	RuGPT3XL few-shot	SberDevices	i	0.535	0.096	0.302 / 0.418	0.676	0.74 / 0.546	0.573	0.565	0.649	0.59	0.67 / 0.665
10	MT5 Large	AGI NLP	i	0.528	0.061	0.366 / 0.454	0.504	0.844 / 0.543	0.561	0.633	0.669	0.657	0.57 / 0.562
11	RuBERT plain	DeepPavlov	i	0.521	0.191	0.367 / 0.463	0.574	0.711 / 0.324	0.642	0.726	0.669	0.639	0.32 / 0.314

Complex applications of NLP : search and chatbots



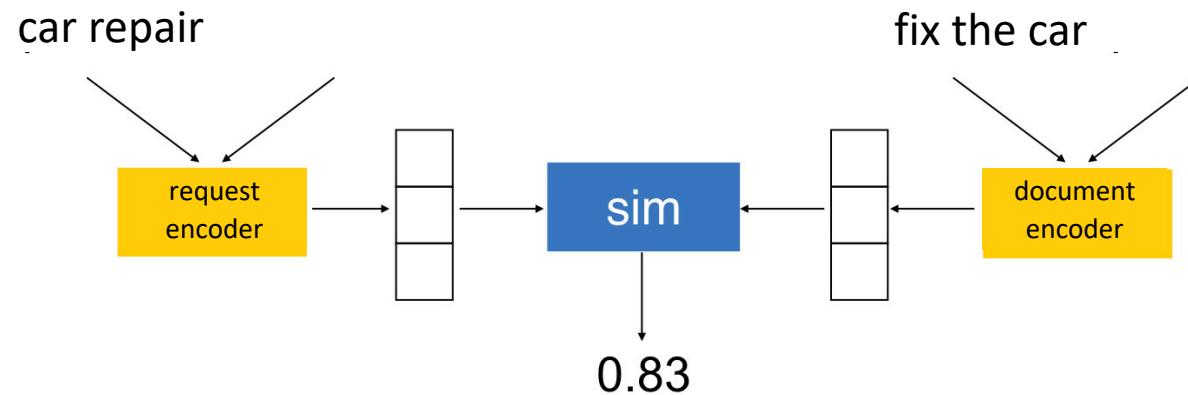
Search task

- Purpose: to search for texts that answer the query in a huge database
- Classic Solution: Reverse Index
 - Documents are broken into words
 - We remember which word was found in which document
 - During the search, we look only at the right words
 - Very fast!



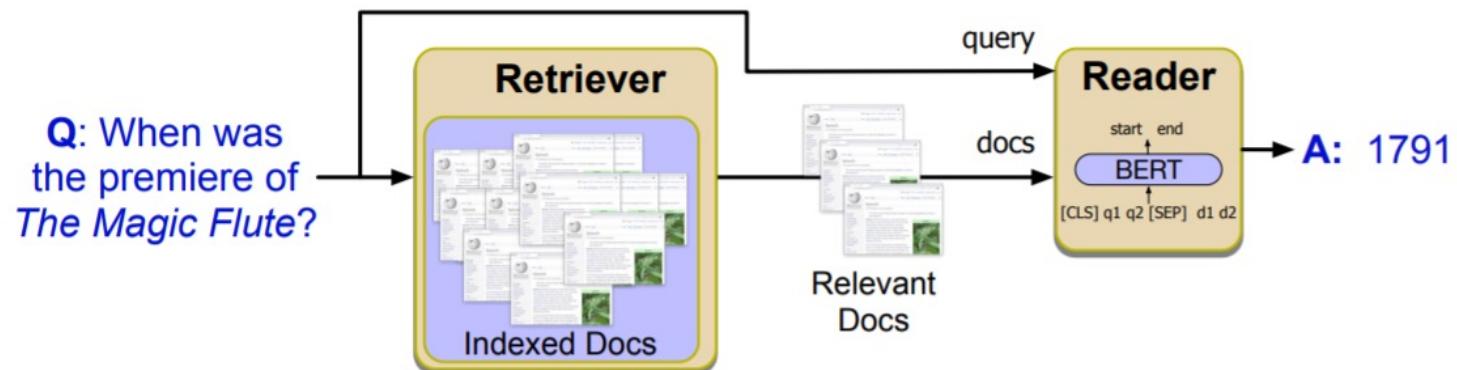
Semantic search

- Problem: The relevant document may not contain the words from the query
 - "fix a car" vs "repair cars"
 - A good crutch: expand the query with synonyms
 - Principal solution: search by embeddings
- Word embeddings vs sentence encoders
- Siamese neural networks for asymmetric search



Finding an answer to a question

- It is not enough to find the document, it is desirable to find the answer in it
- Multi-step procedure
 - Searching for documents using a reverse index or embedding neighbors
 - arranging documents with more complex models
 - Extracting the answer from the top document
 - BERT on a question and a paragraph with an answer predicts the beginning and end of the answer
 - Rephrasing the answer
 - For example, to reconcile it with the question in the form
 - You can train a seq2seq model to do this



Tasks of a chatbot / voice assistant

- Search
 - In a broad sense: answering questions, searching for music, videos, games, ...
- Conversation on general topics
 - Relevance, interest, safety are important
 - Purpose: to captivate the user with a conversation for a long time
- "Useful" scenarios
 - It is important to understand the exact meaning of the request
 - Goal: solve the user's problem quickly and accurately

Chat

- Task: to keep the conversation going on any topic
- Generative approach
 - Request + context --> neuron --> response
 - This is difficult to train (but with large generative models it is possible)
 - Difficult to control quality
 - The answers (theoretically) can be very diverse.
- "Search Approach"
 - Request + context + ready response --> neuron --> relevance assessment
 - You can use Siamese networks as a search
 - base of ready-made answers can be filtered as you like.
 - Easier to manage meaningfulness and diversity
 - of generative and search chatter hypotheses

Расскажи про имя Алиса,
пожалуйста

А что мне за это будет?

Goal-oriented scenarios

- "Wake me up at half past seven tomorrow"
 - intent (intention user): set_alarm
 - Slots (entities supported by an intent):
 - time: 07:30
 - date: tomorrow
- Task execution - handwritten business logic and working with API
- Answers are usually generated from templates
- Difficulties:
 - Context sensitive ("The day after tomorrow too")
 - NLU training without real data when creating new scenarios

Conclusion



So is NLP worth doing?

- NLP is easy. You can train BERT or GPT for any task
- NLP is hard. You need a lot of labeled data and heavy models, but good quality is still not guaranteed
- The line between “easy” and “difficult” has become very blurred since 2017.

When should you do NLP?

- Signs of possible project success:
 - This has already been done for other languages
 - This has already been done for a similar task
 - There is already a satisfactory non- NLP solution that can be improved
 - There is labeled data obtained in a “natural” way
- Sometimes you just have to let yourself try

NLP

- Tasks and leaderboards
 - <http://www.dialog-21.ru/evaluation> - Russian language tasks
 - <https://russiansuper glue.com> - Russian-language benchmark with a bunch of tasks
 - kaggle.com
- Education
 - <https://web.stanford.edu/~jurafsky/slp3/> – most respected book
 - <https://stepik.org/course/54098/syllabus> - a nice course from Samsung on neural models
 - <https://stepik.org/course/1233/info> - pre-neural computational linguistics course
 - https://lena-voita.github.io/nlp_course - Lena Voita encyclopedia course