Text generation with transformers

Outline

- Language models
- GPT and related transformer decoders
- Text generation methods
- Controllable text generation
- ChatGPT & InstructGPT: Reinforcement Learning from Human Feedback (RLHF)

Outline

- Language models
- GPT and related transformer decoders
- Text generation methods
- Controllable text generation

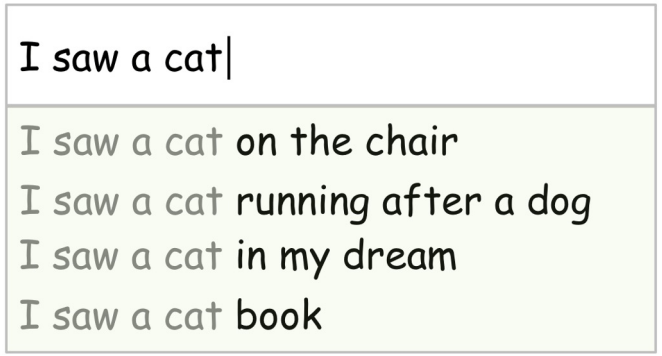
Language models

Language modeling

- Modeling something means to **predicting probability** of some action in a **given context**.

- Language Models (LMs) estimate the probability of different linguistic units: symbols, tokens, token sequences.

Web search engine / ...



We deal with LMs every day!

- We can choose one option between the same sounding (similar) phrases:

Я хочу назвать моего кота наполеон.

Я хочу назвать моего кота на поле он.

Я хочу назвать моего кота Наполеон.

- For automatic models, the probability of the sentence will help in the solution.

Probability of sentence: Intuition

- "**Probability of a sentence**" = as much as possible in natural language.

- Only a specific language is considered:

P(Кот лежит на диване) > P(На диване кот лежит)

P(Красивая девочка играла в мяч) > P(В мяч играла девочка красивая)

Based on https://github.com/king-menin/mipt-nlp2022/blob/master/lectures/L4.LMs_Intro_and_NER.pdf

LMs in NLP

- It is very difficult to know the real probability of a sequence of tokens.

- But we can use a language model to approximate this probability.

- Like all models, language models “behave well” in some cases and “badly” in others.

Based on https://github.com/king-menin/mipt-nlp2022/blob/master/lectures/L4.LMs_Intro_and_NER.pdf

Proper language modelling

The goal of language modelling: **assign a probability to any text (aka sequence of words/tokens)**

Based on https://github.com/king-menin/mipt-nlp2022/blob/master/lectures/L4.LMs_Intro_and_NER.pdf

Proper language modelling

The goal of language modelling: **assign a probability to any text**

What for?

- “More probable” sometimes means “better”
  - Can be used for spelling correction, translation, speech recognition etc.
- By sampling from this distribution, one can generate texts

Based on https://github.com/king-menin/mipt-nlp2022/blob/master/lectures/L4.LMs_Intro_and_NER.pdf

Proper language modelling

The goal of language modelling: **assign a probability to any text**

How?

Solution: chain rule

$$
\begin{aligned}
&P(\text{Я хочу назвать кота Наполеон}) = \\
&= P(\text{Наполеон}|\text{Я хочу назвать кота}) \times P(\text{Я хочу назвать кота}) = \\
&= P(\text{Наполеон}|\text{Я хочу назвать кота}) \times P(\text{кота}|\text{Я хочу назвать}) \times \\
&P(\text{назвать}|\text{Я хочу}) \times P(\text{хочу}|\text{Я}) \times P(\text{Я})
\end{aligned}
$$

Based on https://github.com/king-menin/mipt-nlp2022/blob/master/lectures/L4.LMs_Intro_and_NER.pdf

LMs in NLP

- Language models can be divided into two types:
  - **Count-based Models** (Statistical) - statistical language models.
  - **Neural Language Models** - language models based on neural networks.

Based on https://github.com/king-menin/mipt-nlp2022/blob/master/lectures/L4.LMs_Intro_and_NER.pdf

n-gram language models

$$P(text) = P(w_0 w_1 \dots w_m) = \prod_{i=0}^{m} P(w_i | w_0 \dots w_{i-1})$$

Source of the image: https://lena-voita.github.io/nlp_course/language_modeling.html#intro

n-gram language models

$$P(text) = P(w_0 w_1 \dots w_m) = \prod_{i=0}^{m} P(w_i | w_0 \dots w_{i-1})$$

Markov assumption: we can predict the probability of some future unit without looking too far into the past (on a current step distant words do not matter much)

$$P(w_i | w_0 \dots w_{i-1}) \approx P(w_i | w_{i-n} \dots w_{i-1})$$

Source of the image: https://lena-voita.github.io/nlp_course/language_modeling.html#intro

n-gram language models

$$P(text) = P(w_0 w_1 \dots w_m) = \prod_{i=0}^{m} P(w_i | w_0 \dots w_{i-1})$$

- Simplifying assumption: distant words do not matter much

$$P(w_i | w_0 \dots w_{i-1}) \approx P(w_i | w_{i-n} \dots w_{i-1})$$

How to estimate *P*:

- P= C(n-gram)/∑C(n-gram)

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\text{I}|\text{<s>}) = \frac{2}{3} = .67$ $P(\text{Sam}|\text{<s>}) = \frac{1}{3} = .33$ $P(\text{am}|\text{I}) = \frac{2}{3} = .67$

$P(\text{</s>}|\text{Sam}) = \frac{1}{2} = 0.5$ $P(\text{Sam}|\text{am}) = \frac{1}{2} = .5$ $P(\text{do}|\text{I}) = \frac{1}{3} = .33$

Source of the image: https://lena-voita.github.io/nlp_course/language_modeling.html#intro

n-gram language models

Before

P(I saw a cat on a mat) =

P(I)
· P(saw | I)
· P(a | I saw)
· P(cat | I saw a)
· P(on | I saw a cat)
· P(a | I saw a cat on)
· P(mat | I saw a cat on a)

After (3-gram)

P(I saw a cat on a mat) =

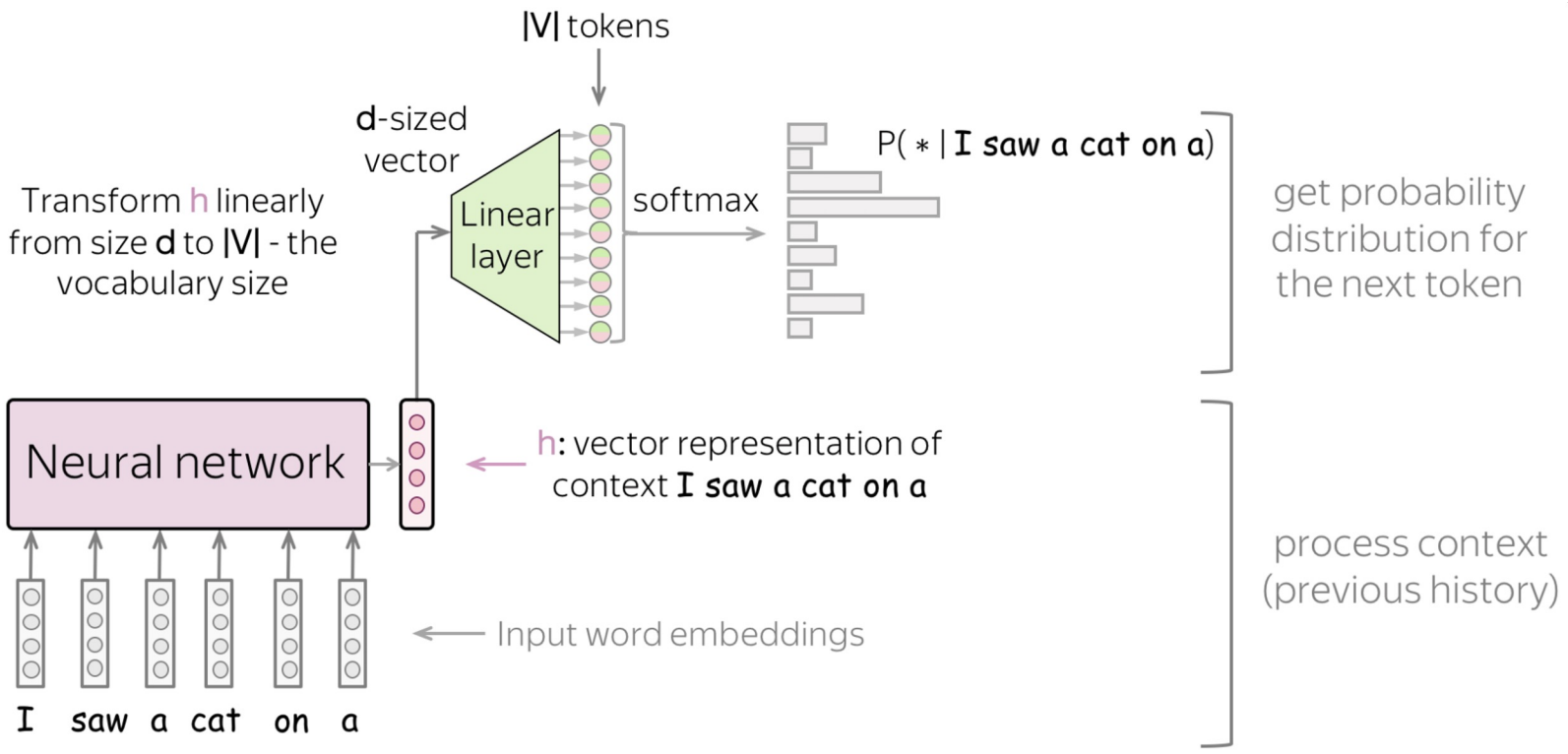
P(I) → P(I)
· P(saw | I) → · P(saw | I)
· P(a | I saw) → · P(a | I saw)
· P(cat | ~~I~~ saw a) → · P(cat | saw a)
· P(on | ~~I saw~~ a cat) → · P(on | a cat)
· P(a | ~~I saw a~~ cat on) → · P(a | cat on)
· P(mat | ~~I saw a cat~~ on a) → · P(mat | on a)

ignore use

Source of the image: https://lena-voita.github.io/nlp_course/language_modeling.html#intro

Neural language models

Neural LMs encode *the context* into a vector representation and then predict probability distribution of the *next token.*

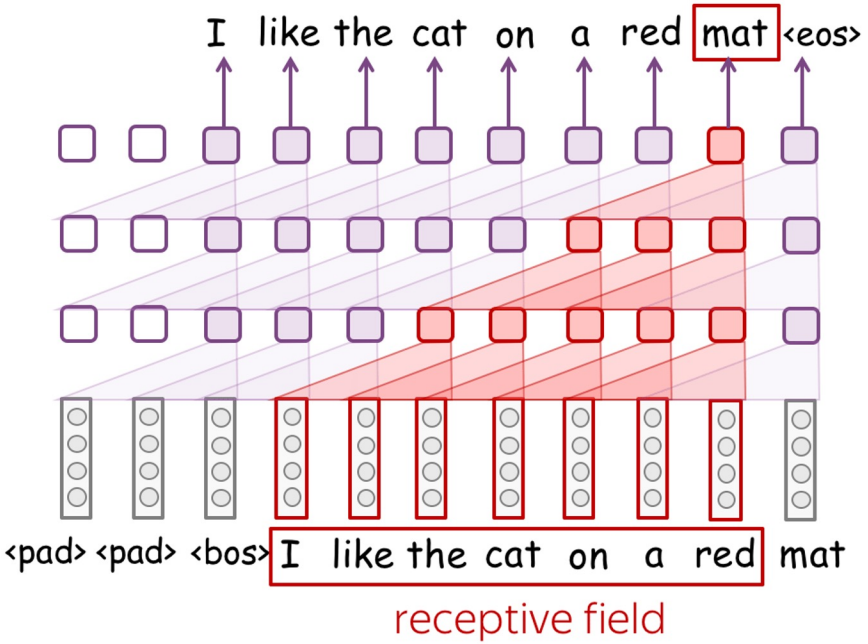


Source: https://lena-voita.github.io/nlp_course/language_modeling.html#generation

CNN language models (Fixed Window Neural Language Models)

- What are the **improvements** in comparison with N-gram LM:
  - No sparsity problem

CNN: a strictly limited receptive field; training is very fast because it can run in parallel on all time steps



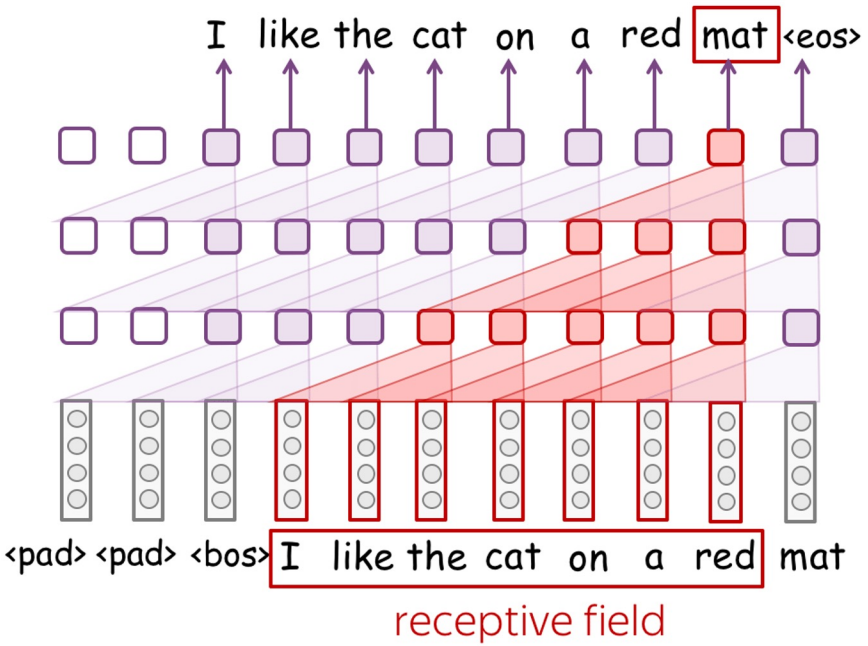
*In practice, when training RNNs, gradient flows only for *n* time steps backward, so the models do not learn to account for very distant context

Images from https://lena-voita.github.io/nlp_course/language_modeling.html#neural_lms

CNN language models (Fixed Window Neural Language Models)

- What are the **improvements** in comparison with N-gram LM:
  - No sparsity problem
- What **problems** remained:
  - Fixed size windows - very small, cannot be changed
  - Fixed word order
  - Weights can only be used inside the window

CNN: a strictly limited receptive field; training is very fast because it can run in parallel on all time steps



Images from https://lena-voita.github.io/nlp_course/language_modeling.html#neural_lms

RNN language models

- Final distribution:

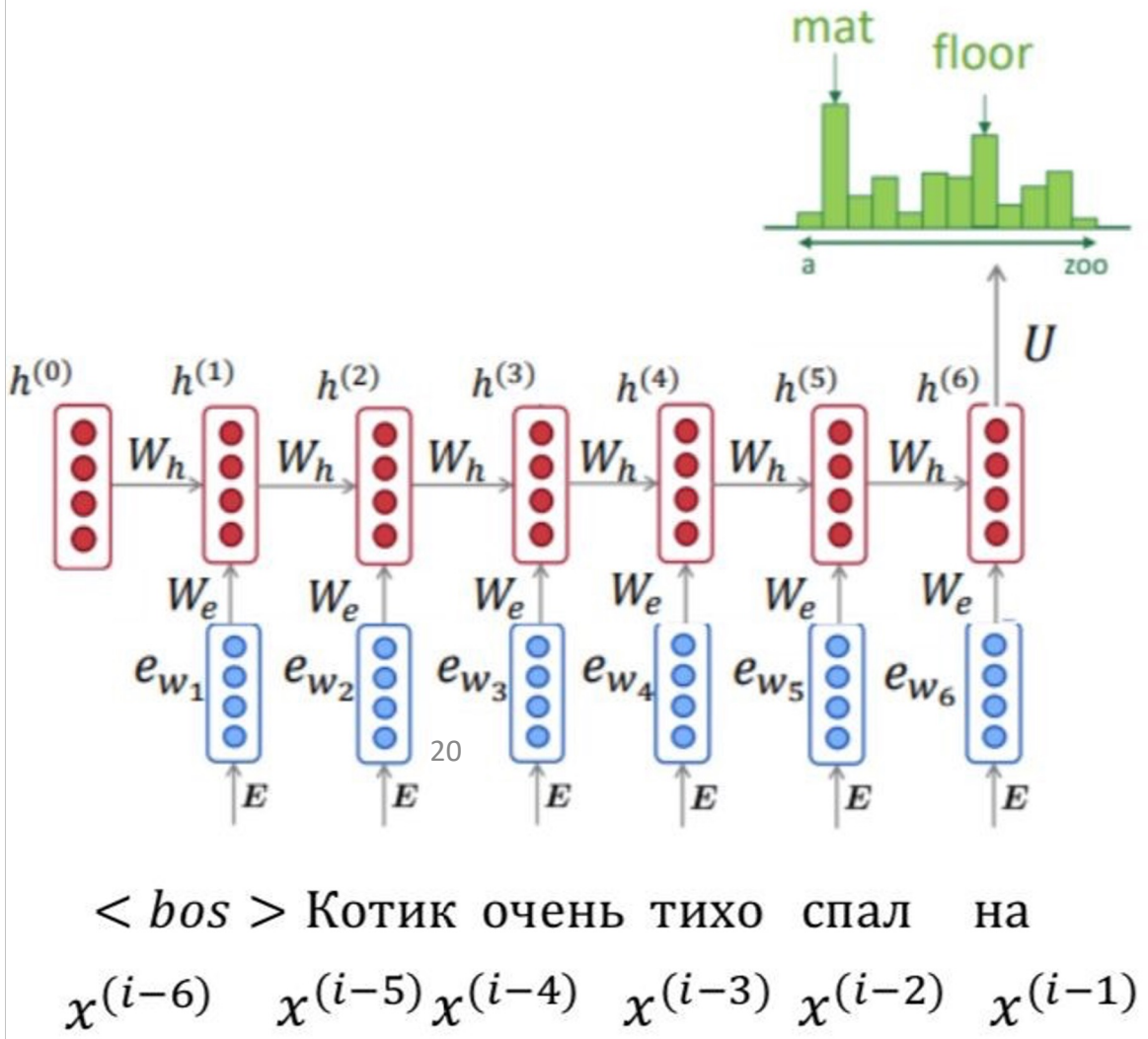
$$\hat{y} = softmax(Uh + b_2) \in R^{|V|}$$

- Hidden states

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_x e^{(t-1)} + b_1\right)$$

- Word embeddings

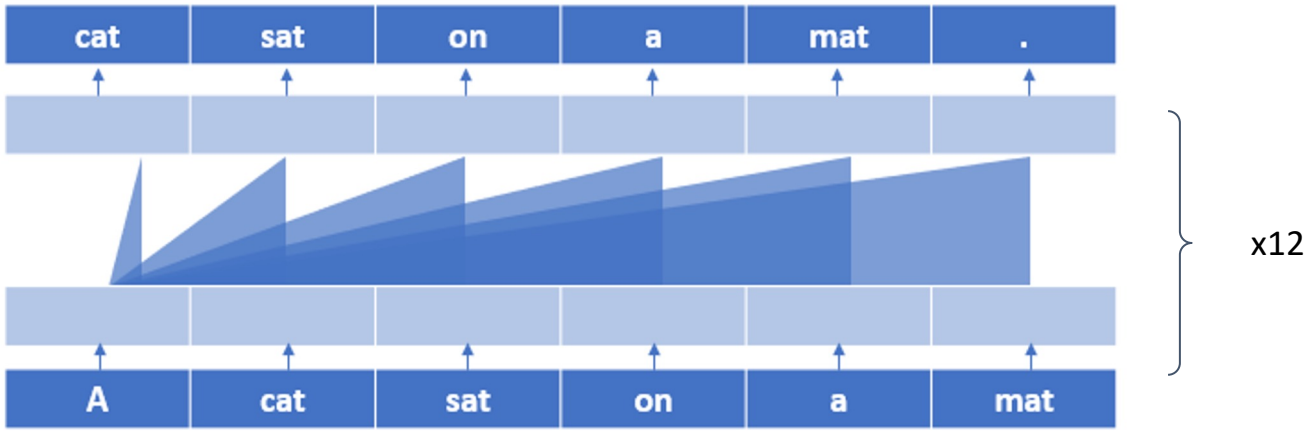
- Words



*In practice, when training RNNs, gradient flows only for *n* time steps backward, so the models do not learn to account for very distant context

Transformer language models

- **Attention** => they have direct access to previous tokens
- Support longer contexts than CNNs
- Train faster than RNNs, because do not require recurrence



Perplexity and likelihood

How good a language model is?

Perplexity and likelihood

How good a language model is?

When we train a LM, we *(equivalently)*

- *maximize log-likelihood:* $LL = \log p(text) = \sum_{i=1}^{n} \log p(y_i|y_{<i})$
- *minimize cross-entropy:* $CE = -\frac{1}{n}\sum_{i=1}^{n} \log p(y_i|y_{<i}) = -\frac{LL}{n}$
- *minimize perplexity:* $ppl = \frac{1}{p(text)} = e^{CE} = e^{-\frac{LL}{n}} = e^{-\frac{1}{n}\sum_{i=1}^{n} \log p(y_i|y_{<i})}$

Perplexity and likelihood

How good a language model is?

When we train a LM, we *(equivalently)*

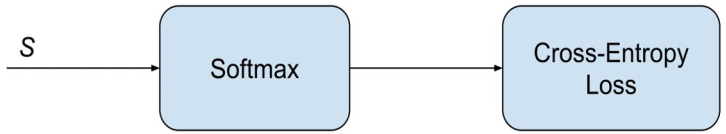
- *maximize log-likelihood:* $LL = \log p(text) = \sum_{i=1}^{n} \log p(y_i|y_{<i})$
- *minimize cross-entropy:* $CE = -\frac{1}{n}\sum_{i=1}^{n} \log p(y_i|y_{<i}) = -\frac{LL}{n}$
- *minimize perplexity:* $ppl = \frac{1}{p(text)} = e^{CE} = e^{-\frac{LL}{n}} = e^{-\frac{1}{n}\sum_{i=1}^{n} \log p(y_i|y_{<i})}$

To evaluate a model, we *calculate the same scores on test texts*

Categorical Cross-Entropy loss (Softmax loss)

In its output, autoregressive language models predict probability distribution over all tokens in a given vocabulary. It is a Categorical problem:

- We need Softmax to normalize probability



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

- Cross Entropy over softmax now looks like

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$

Pytorch: CrossEntropyLoss.

TensorFlow: softmax_cross_entropy.

Language modeling as a pretraining objective

- It is **difficult** (even for humans)
  - *Autoregressive modeling is more difficult than masked language modeling*

*The autoregressive model is a feed-forward model, that predicts the future word from a set of words in a given context.

- It forces the model to **learn lots of aspects of the language and the world**
- It requires **no inputs except the texts**
- **No such problems** with hyperparameters as in MLM

Everything as language modelling

Most NLP problems can be formulated as text continuation problems

Few Shot Prompt and Predicted Answer

The following are multiple choice questions about high school mathematics.

How many numbers are in the list 25, 26, ..., 100?
(A) 75 (B) 76 (C) 22 (D) 23
Answer: B

Compute $i + i^2 + i^3 + \cdots + i^{258} + i^{259}$.
(A) -1 (B) 1 (C) i (D) $-i$
Answer: A

If 4 daps = 7 yaps, and 5 yaps = 3 baps, how many daps equal 42 baps?
(A) 28 (B) 21 (C) 40 (D) 30
Answer: C

Translate each sentence into a string of emojis.

English: That cat ate the fish.

Emojis: 🐱😋🐟

English: What is this, a house for ants?

Emojis: 🏠🐜❓

English: The quick brown fox jumps over the lazy dog.

Emojis: 🦊🐕🐶

English: One small step for man, one giant leap for mankind.

Emojis: 🚀🌍🏃

Sources: https://arxiv.org/abs/2009.03300, https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

Everything as language modelling

Most NLP problems can be formulated as text continuation problems

A single good LM can learn to solve them all!

Few Shot Prompt and Predicted Answer

The following are multiple choice questions about high school mathematics.

How many numbers are in the list 25, 26, ..., 100?
(A) 75 (B) 76 (C) 22 (D) 23
Answer: B

Compute $i + i^2 + i^3 + \cdots + i^{258} + i^{259}$.
(A) -1 (B) 1 (C) i (D) $-i$
Answer: A

If 4 daps = 7 yaps, and 5 yaps = 3 baps, how many daps equal 42 baps?
(A) 28 (B) 21 (C) 40 (D) 30
Answer: C

Translate each sentence into a string of emojis.

English: That cat ate the fish.

Emojis: 🐱😋🐟

English: What is this, a house for ants?

Emojis: 🏠🐜❓

English: The quick brown fox jumps over the lazy dog.

Emojis: 🦊🟫🐶🐕

English: One small step for man, one giant leap for mankind.

Emojis: 🚀🌍🏃

Sources: https://arxiv.org/abs/2009.03300, https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

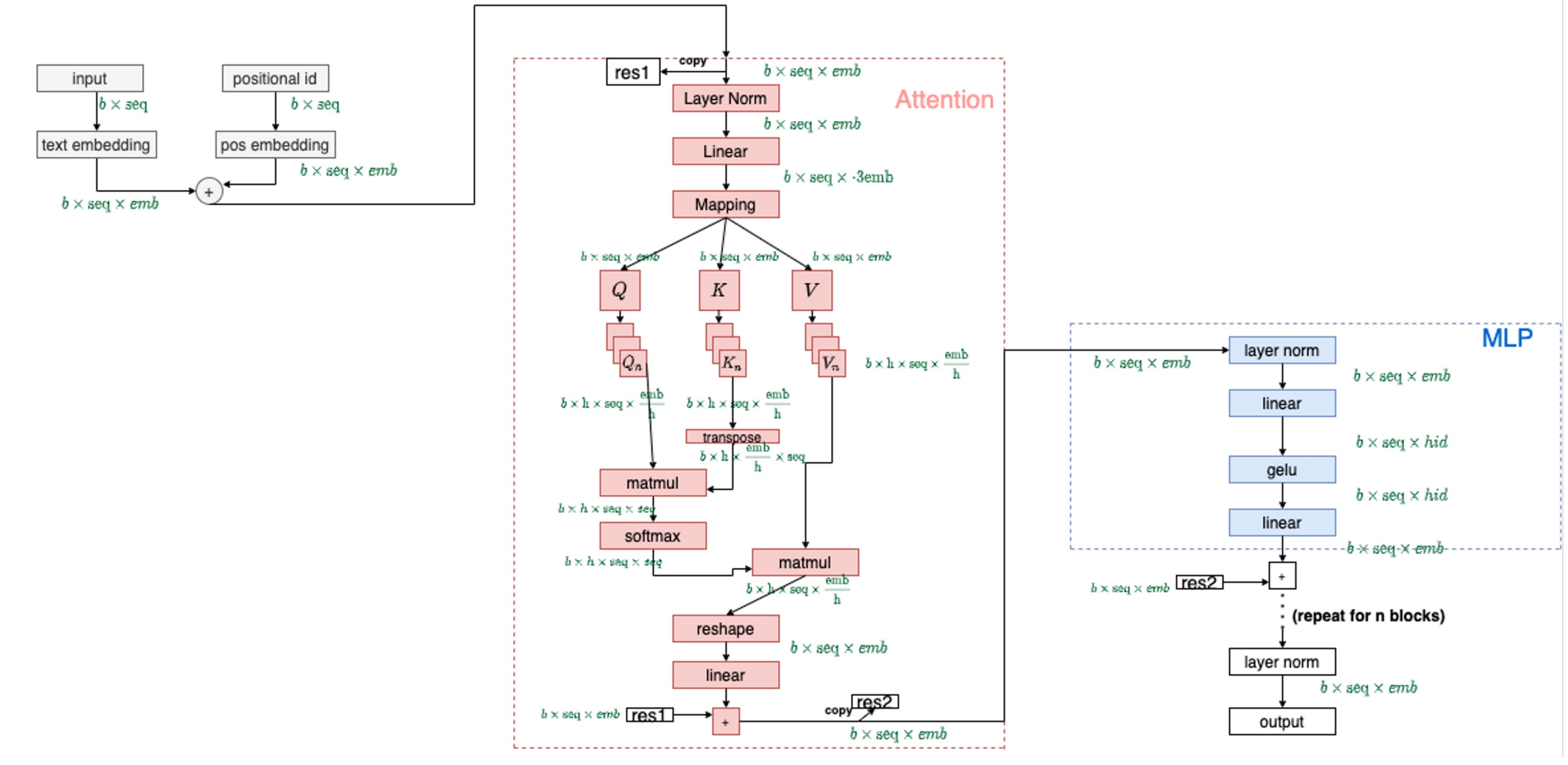
The GPT family

Generative Pretrained Transformers

The **main idea**:

- *Pretrain a large transformer* decoder on the language modelling task
- Formulate another NLP task as *text continuation*
- Use the LM to solve this task with little or no fine-tuning

GPT architecture

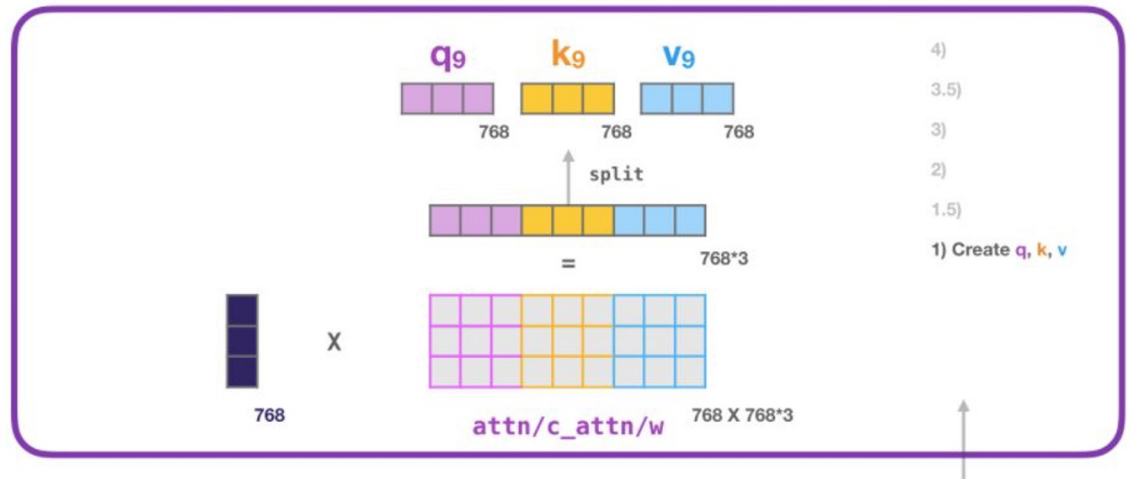


How GPT looks in the real world?

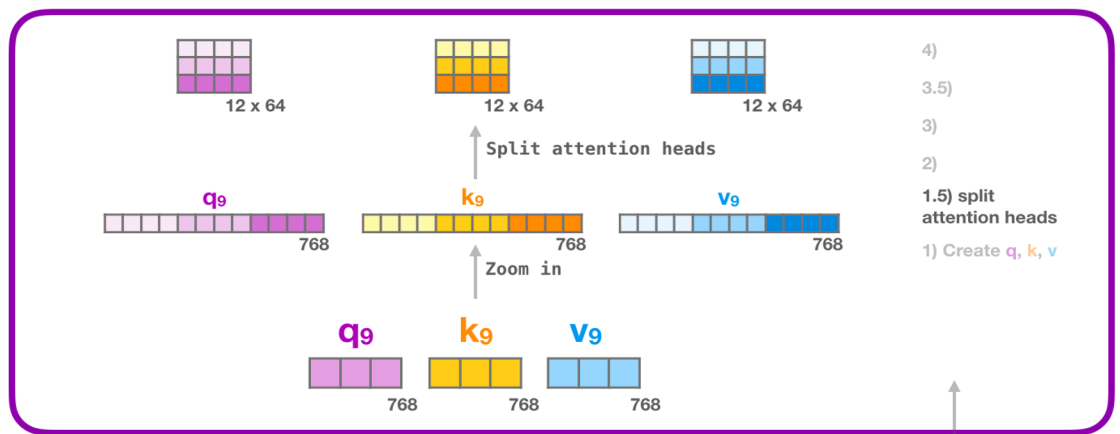
```
gpt2.transformer.h[1]
```

```
GPT2Block(
  (ln_1): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
  (attn): GPT2Attention(
    (c_attn): Conv1D() [768, 2304]
    (c_proj): Conv1D()  [768, 768]
    (attn_dropout): Dropout(p=0.1, inplace=False)
    (resid_dropout): Dropout(p=0.1, inplace=False)
  )
  (ln_2): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
  (mlp): GPT2MLP(
    (c_fc): Conv1D() [768, 3072]
    (c_proj): Conv1D()[3072, 768]
    (act): NewGELUActivation()
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
```

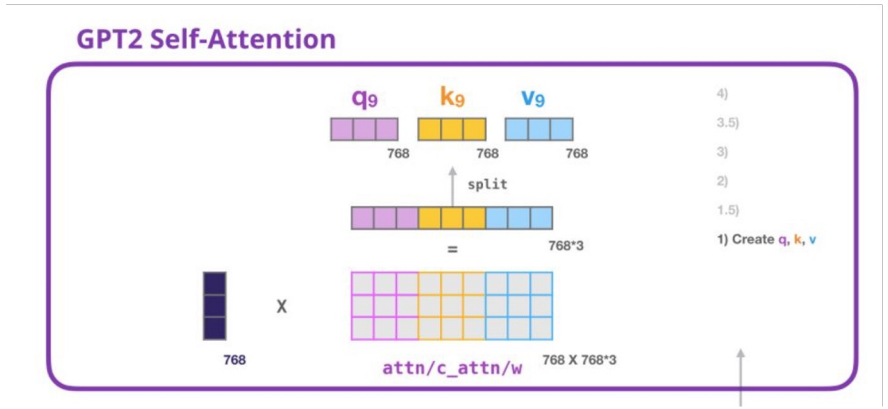
GPT2 Self-Attention

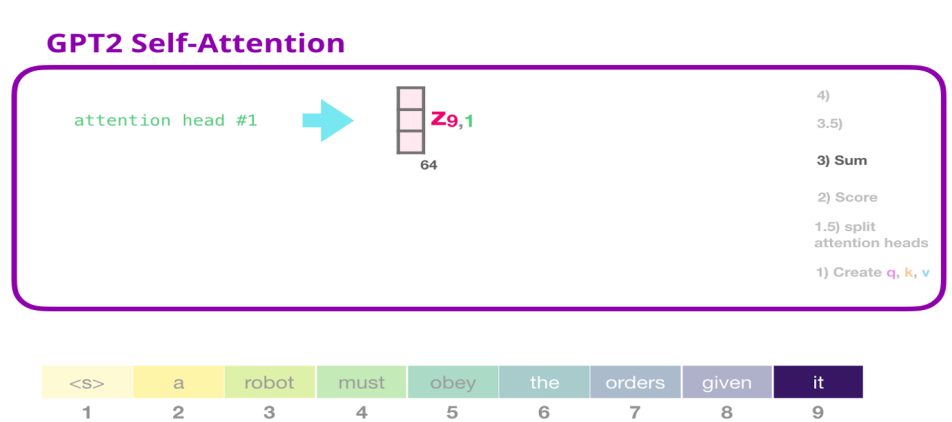


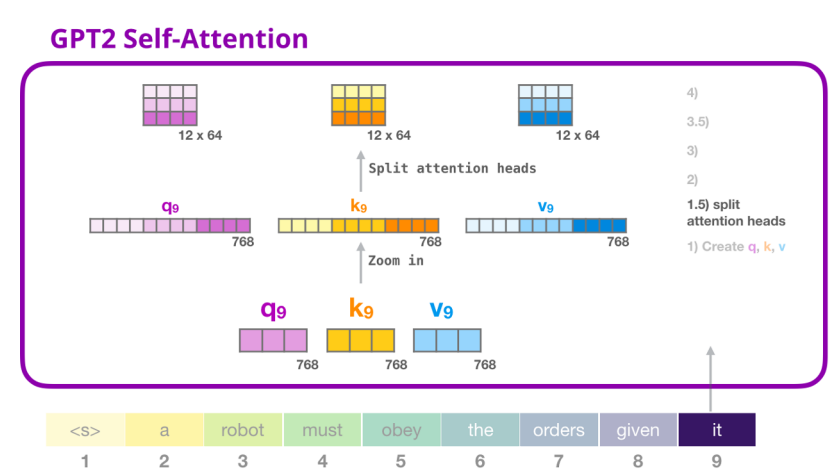
GPT2 Self-Attention

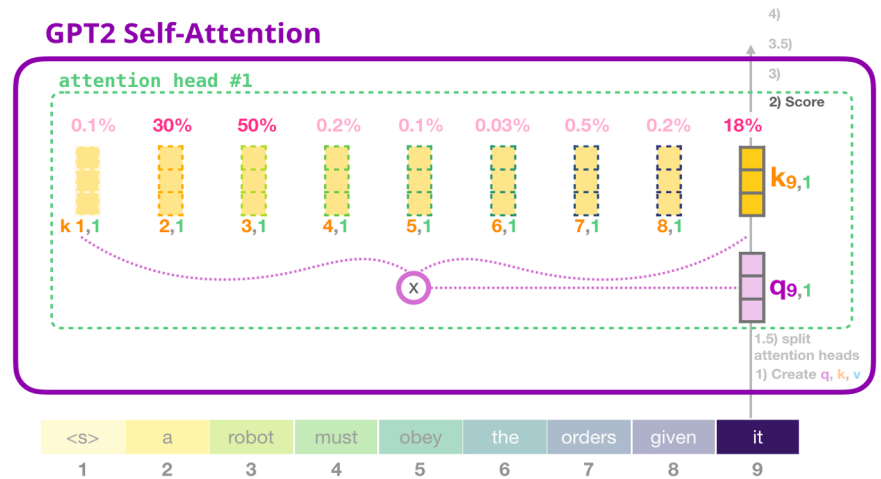


<s>	a	robot	must	obey	the	orders	given	it
1	2	3	4	5	6	7	8	9

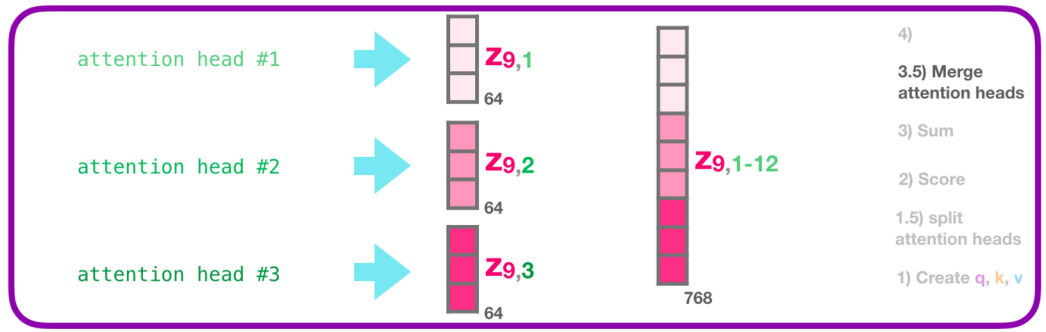

GPT2 Self-Attention
q9
k9
v9
768
768
768
split
768*3
=
X
768
attn/c_attn/w
768 X 768*3
4)
3.5)
3)
2)
1.5)
1) Create q, k, v

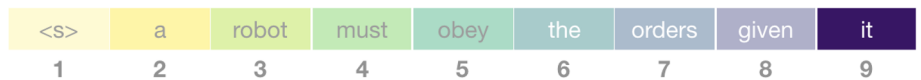

GPT2 Self-Attention
attention head #1
z9,1
64
4)
3.5)
3) Sum
2) Score
1.5) split attention heads
1) Create q, k, v
<s>
a
robot
must
obey
the
orders
given
it
1
2
3
4
5
6
7
8
9


GPT2 Self-Attention
12 x 64
12 x 64
12 x 64
Split attention heads
q9
k9
v9
768
768
768
Zoom in
q9
k9
v9
768
768
768
4)
3.5)
3)
2)
1.5) split attention heads
1) Create q, k, v
<s>
a
robot
must
obey
the
orders
given
it
1
2
3
4
5
6
7
8
9

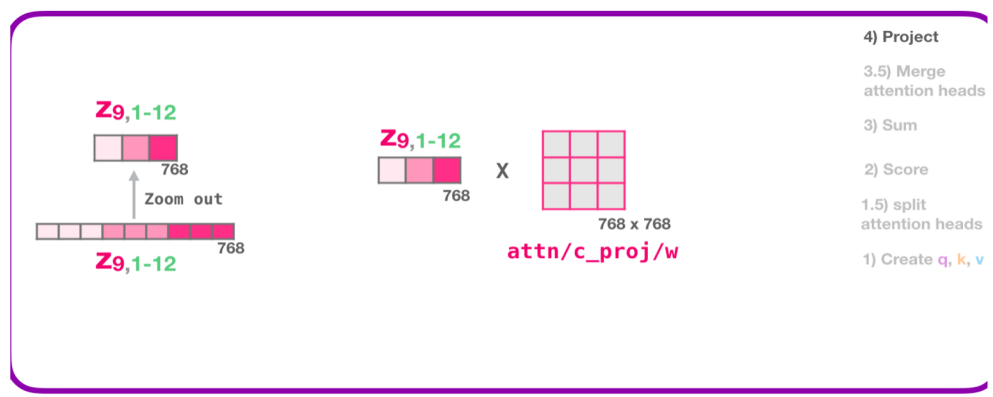

GPT2 Self-Attention
attention head #1
0.1%
30%
50%
0.2%
0.1%
0.03%
0.5%
0.2%
18%
k1,1
2,1
3,1
4,1
5,1
6,1
7,1
8,1
k9,1
q9,1
X
4)
3.5)
3)
2) Score
1.5) split attention heads
1) Create q, k, v
<s>
a
robot
must
obey
the
orders
given
it
1
2
3
4
5
6
7
8
9

GPT2 Self-Attention



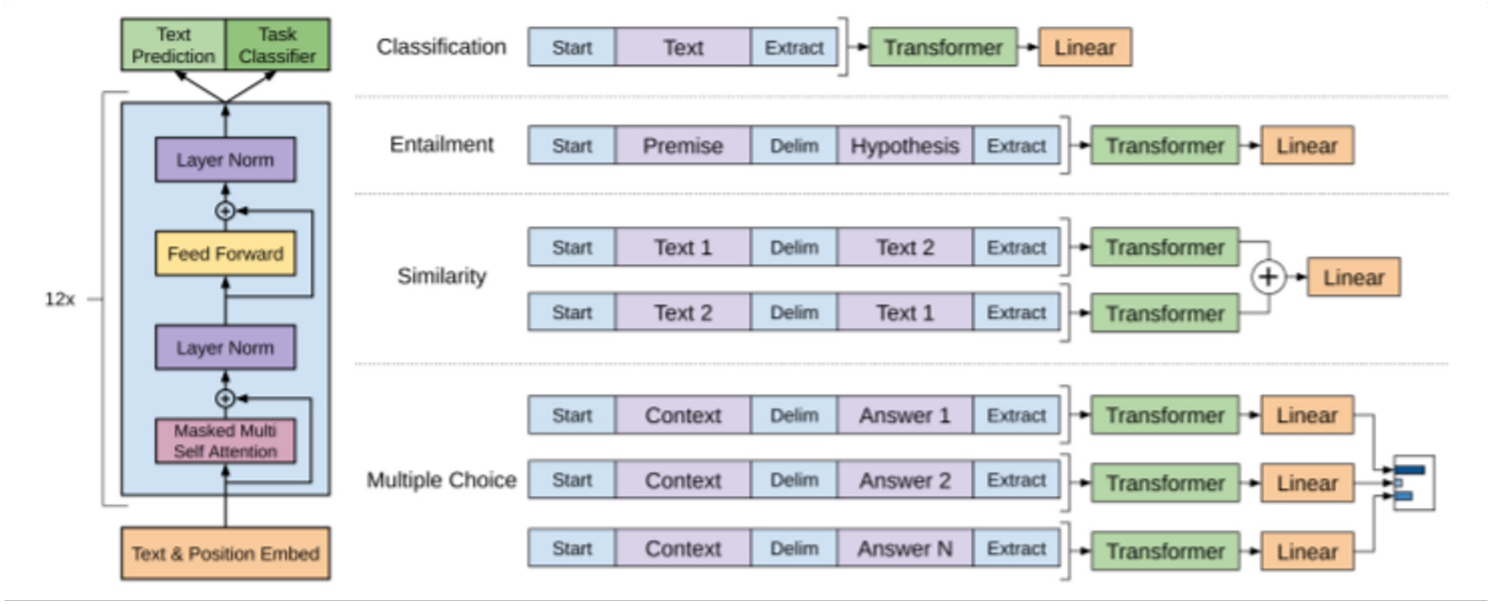


GPT2 Self-Attention



GPT-1

- *Improving Language Understanding by Generative Pre-Training, Radford et al, 2018*
  - The paper was published several months before BERT
- Idea: pre-train a transformer decoder on the LM problem, then fine-tune on NLU tasks
  - GPT-1 is trained on BookCorpus dataset bases on 7000 unpublished books
  - GPT-1 is fine-tuned on classification, semantic proximity detection, natural language generation, and question-answer tasks
  - GPT-1 is evaluated on 12 tasks and be SOTA for 8 of them.



https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

GPT-2

- *Language Models are Unsupervised Multitask Learners, Radford et al, 2019*
- **Idea:** if we increase a model size, we increase performance!
- Scale GPT up:
  - Train on **40GB** of WebText, up to **1.5B** params (vs 117M in GPT-1)
  - Perplexity on multiple test LM datasets improves with model size and does not saturate
- The models show good zero-shot performance on reading comprehension, question answering, even fr-en translation
- The largest model generates fluent and well-structured texts

https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

GPT-3

- *Language Models are Few-Shot Learners, Brown et al, 2020*
- Scale GPT-2 further: **570GB training data**, up to **175B parameters**

https://arxiv.org/abs/2005.14165

GPT-3

- *Language Models are Few-Shot Learners, Brown et al, 2020*
- Scale GPT-2 further: **570GB training data**, up to **175B parameters**
- **Few-shot approach**

  - apply to NLP tasks without fine-tuning: the only “learning” takes place with few training examples being part of a prompt

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:      ← task description
2  cheese =>                         ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:      ← task description
2  sea otter => loutre de mer        ← examples
3  peppermint => menthe poivrée      ←
4  plush girafe => girafe peluche    ←
5  cheese =>                         ← prompt
```

https://arxiv.org/abs/2005.14165

GPT-3 application examples: NLU

Prompt

```
The following is a list of companies and the categories they fall into:

Apple, Facebook, Fedex

Apple
Category:
```

Sample response

```
Technology

Facebook
Category: Social Media

Fedex
Category: Delivery
```

A simple prompt for extracting airport codes from text.

Prompt

```
Extract the airport codes from this text:

Text: "I want to fly from Los Angeles to Miami."
Airport codes: LAX, MIA

Text: "I want to fly from Orlando to Boston"
Airport codes:
```

Sample response

```
MCO, BOS
```

From https://beta.openai.com/examples

GPT-3 application examples: NLG

Explain a piece of Python code in human understandable language.

Prompt

```
# Python 3
def remove_common_prefix(x, prefix, ws_prefix):
    x["completion"] = x["completion"].str[len(prefix) :]
    if ws_prefix:
        # keep the single whitespace as prefix
        x["completion"] = " " + x["completion"]
return x

# Explanation of what the code does

#
```

Sample response

The code above is a function that takes a dataframe and a prefix as input and returns a dataframe with the prefix removed from the completion column.

Turn a product description into ad copy.

Prompt

Write a creative ad for the following product to run on Facebook aimed at parents:

Product: Learning Room is a virtual environment to help students from kindergarten to high school excel in school.

Sample response

Are you looking for a way to help your child excel in school? Look no further than Learning Room! Our virtual environment is designed to help students from kindergarten to high school learn and grow. With our help, your child will be able to get ahead in school and achieve their academic goals.

From https://beta.openai.com/examples

GPT-3 application examples: code generation

Translate text into programmatic commands.

Prompt

```
Convert this text to a programmatic command:

Example: Ask Constance if we need some bread
Output: send-msg `find constance` Do we need some bread?

Contact the ski store and figure out if I can get my skis fixed before I leave on Thursday
```

Sample response

```
send-msg `find ski-store` Can I get my skis fixed before I leave on Thursday?
```

Translate natural language to SQL queries.

Prompt

```
### Postgres SQL tables, with their properties:
#
# Employee(id, name, department_id)
# Department(id, name, address)
# Salary_Payments(id, employee_id, amount, date)
#
### A query to list the names of the departments which employed more than 10 employees in the last 3 months
SELECT
```

Sample response

```
 DISTINCT department.name
FROM department
JOIN employee ON department.id = employee.department_id
JOIN salary_payments ON employee.id = salary_payments.employee_id
WHERE salary_payments.date > (CURRENT_DATE - INTERVAL '3 months')
GROUP BY department.name
HAVING COUNT(employee.id) > 10;
```

From https://beta.openai.com/examples

Details of GPT implementation

- All GPTs are almost vanilla transformer decoders
  - GPT-3 also uses sparse attention alongside classical one
    - details are unknown
- Byte-level BPE vocabulary, ~50K tokens
- Context window of 2048 tokens in GPT-3

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Source: https://arxiv.org/abs/2005.14165

GPT-4

- *Based on GPT*
- Accepts image and text inputs, emitting text outputs
- Model context 8,192 tokens
- There is a version with 32,768 tokens (50 text pages)
- **NO PAPER**
- **NO INFORMATION about its architecture**

Архитектура:

Любопытно, что GPT-4 закрыла собой долгие споры о том, куда все-таки нужно ставить LayerNorm в Трансформере!

From https://beta.openai.com/examples

GPT-like models

- GPT-Neo by EleutherAI (English, public reproduction of GPT-3)
- LaMDA by Google (English, dialogue-oriented, private)
- Megatron-Turing NLG by Microsoft and Nvidia (English, private)
- PanGu-α by Huawei (Chineze, public)
- Wu Dao by BAAI (Chineze, private)
- HyperCLOVA by Naver (Korean, private)
- RuGPT by Sber (Russian, public)
- YaLM by Yandex (Russian, public)
- XGLM by Meta (multilingual, public)
- mGPT by Sber (multilingual, public)
- ChatGPT
- GPT-4

https://gpt3demo.com/category/alternative-language-models , https://lifearchitect.ai/models/

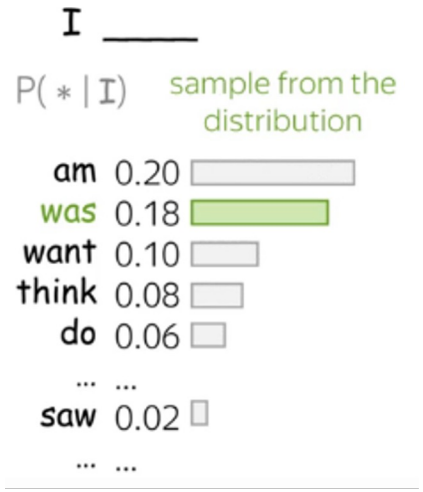
A possible future of GPTs

- Similar models for other languages
- GPT-like models for other modalities (e.g. DALL-E for images)
- More technologies for controllable generation
- Various model sizes
  - Even larger models for better quality?
  - Smaller models for better efficiency?

Methods of text generation

Basic text generation

- Decoding strategy (Exhaustive, Greedy, Beam searches)
  - This does not necessary produce the most probable text
- Sampling method: modify the distribution on each step in proper way
  - This will accurately reproduce the distribution of all possible texts

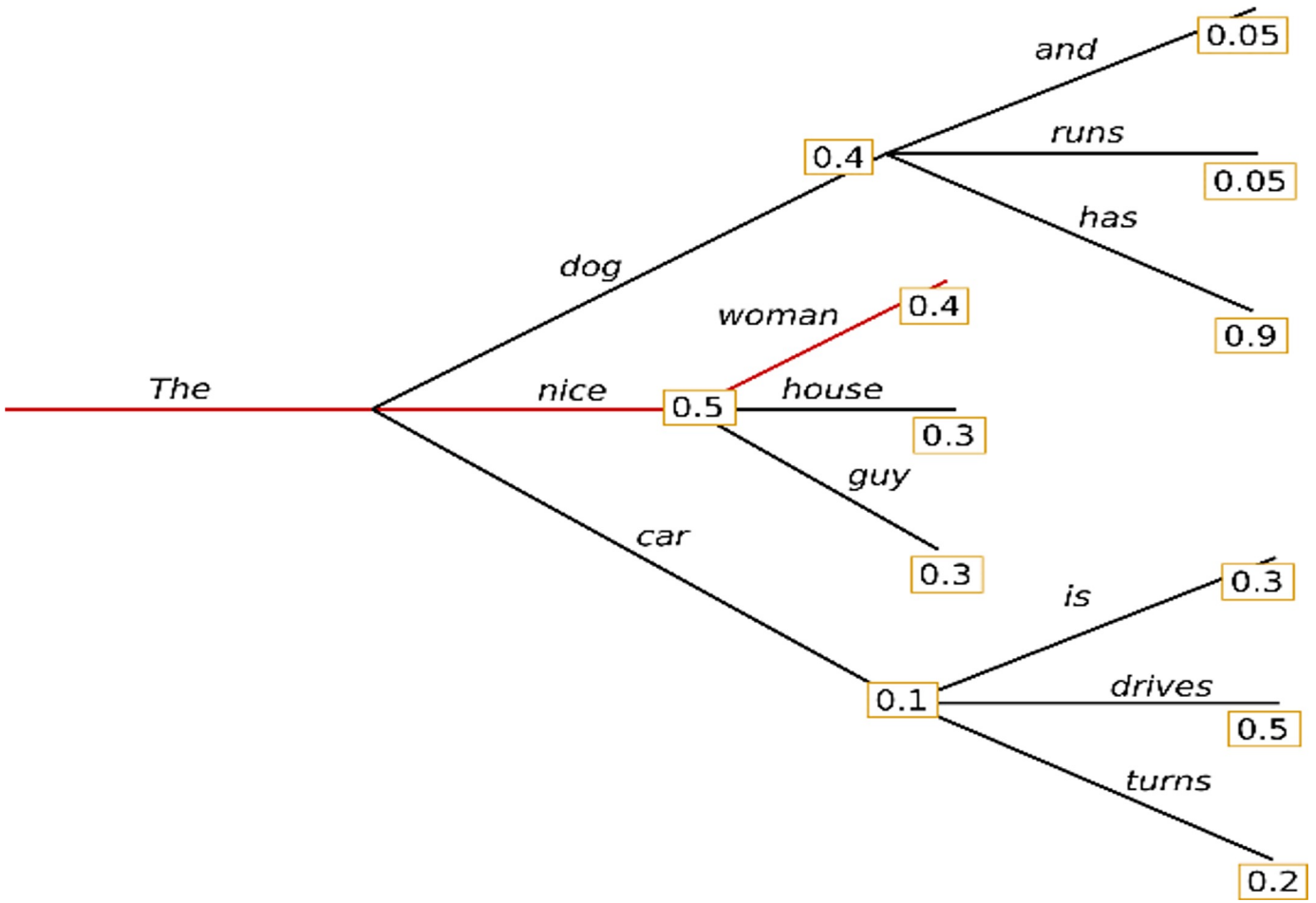


Images from: https://lena-voita.github.io/nlp_course/language_modeling.html#generation

Decoding strategies:

- Exhaustive search
  - Intractable, O(V^T), but guarantees the most probable text
- Greedy decoding (the most probable word on each step)
  - This does not necessary produce the most probable text
- Beam search
  - On each step, keep only the K best hypotheses
  - A compromise between complexity and quality, O(K*T)
  - The default approach in machine translation

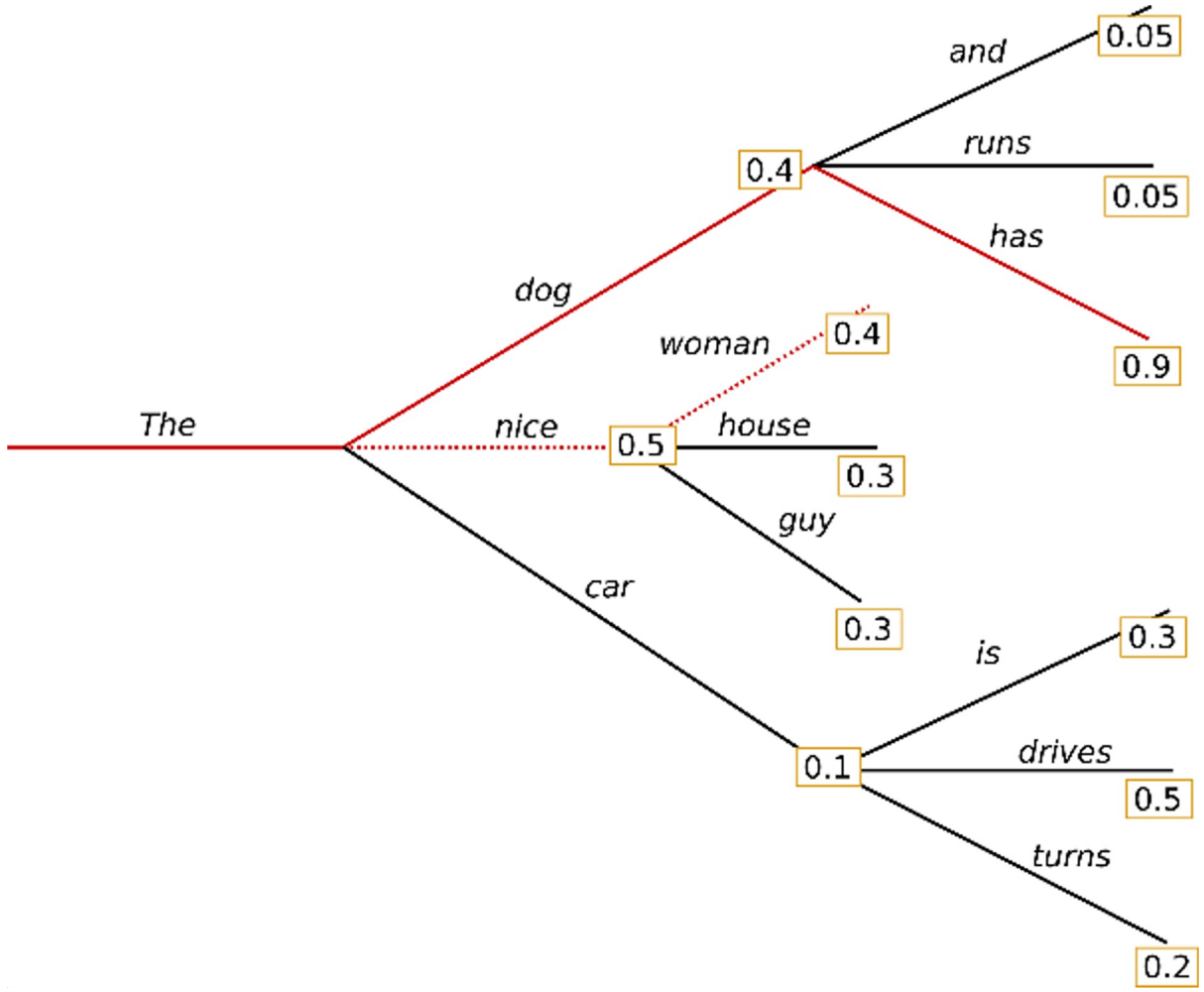
Gready search example



Socher, Manning. Cs224n, 2017

Beam search example

Beam size = 2



Socher, Manning. Cs224n, 2017

Sampling methods:

- Simple sampling
  - Produce scores S of all tokens
  - Sample the next token from softmax: P ~ exp(S)
  - This can generate any text with non-zero probability

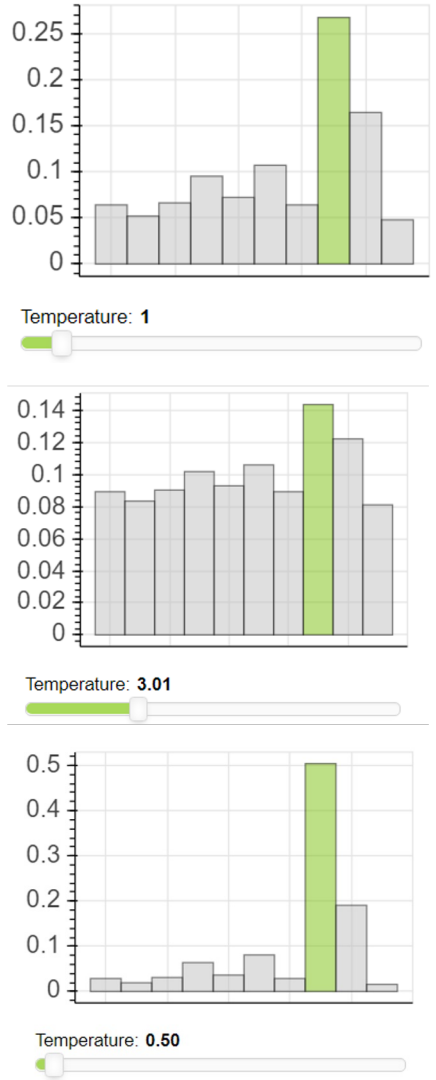
- Problem: sometimes words with not high probability is sampled and text looks incoherent
- Solution: make distribution sharper by multiplying exponent by temperature

$$q = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

https://lena-voita.github.io/resources/lectures/lang_models/sampling/softmax_temperature.html

Sampling

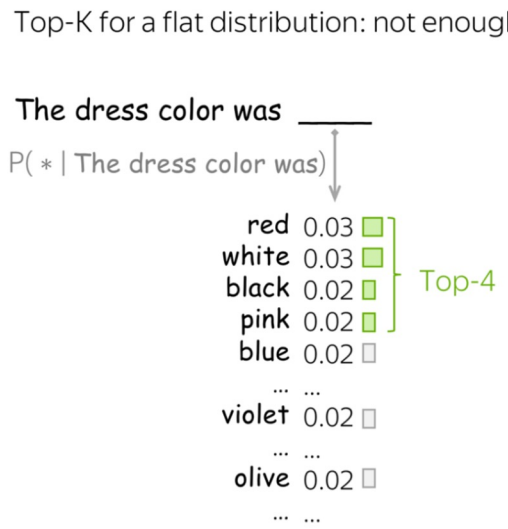
- Temperature sampling
  - Sample from $P \sim \exp(S/\tau)$, where $\tau>0$ is a parameter
    - higher $\tau \leftrightarrow$ more diversity
    - $\tau \to \infty$ – uniform sampling
    - $\tau \to 0$ – greedy decoding (always choose the most probable token)

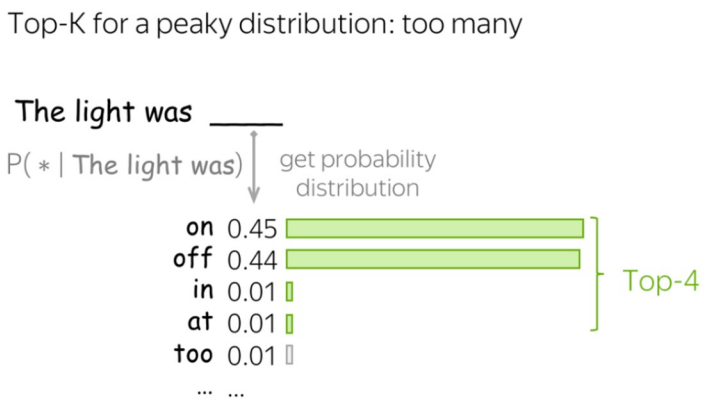


https://lena-voita.github.io/resources/lectures/lang_models/sampling/softmax_temperature.html

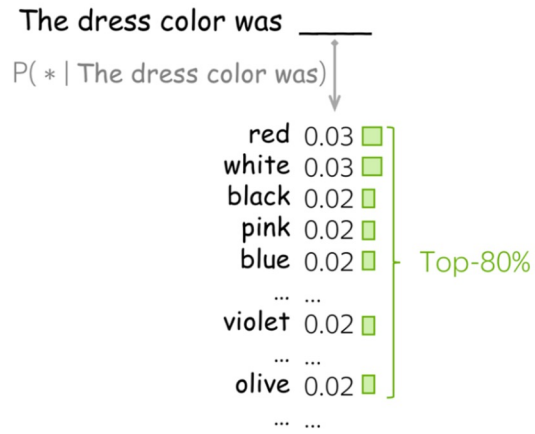
Restricted sampling

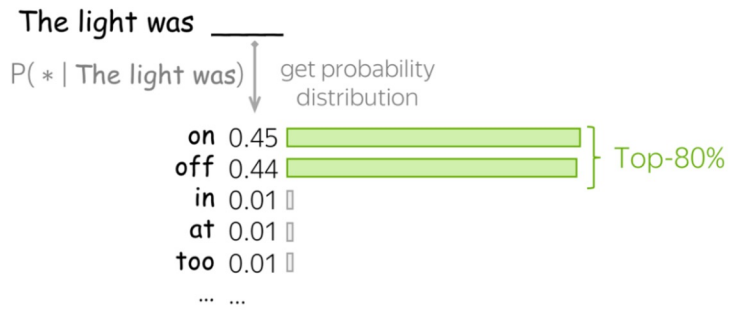
- Top-k sampling
  - Sample only from the K most probable tokens





- Top-p (nucleus) sampling
  - Sample from the M most probable tokens with P% of total probability





https://lena-voita.github.io/nlp_course/language_modeling.html#generation_strategies_top_k

Some extra generation parameters

- Repetition penalty
  - Lower the probability of tokens that were already used
- Length penalty
  - Prefer longer or shorter hypotheses
- Prohibiting certain n-grams:
  - Blacklist of words and phrases ("bad words")
  - Repeating n-grams
- Group beam search (with diversity penalty)
  - Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models, Vijayakumar et al, 2018
  - Split the beam in groups; add to objective measuring the dissimilarity of a sequence y[t] against other groups (i.e. by counting unintersected n-grams or KL)
  - Sequence is learning to be most probable and diversed
  - This increases diversity and even BLEU for translation and image captioning.

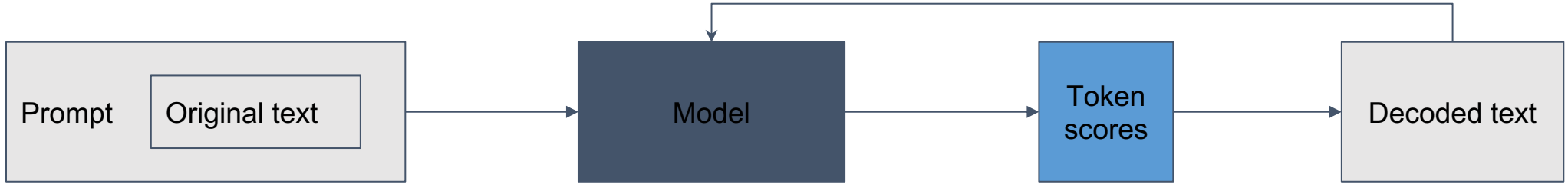
https://arxiv.org/abs/1610.02424, https://huggingface.co/docs/transformers/internal/generation_utils

Controllable text generation

Controllable text generation: why

- Making language models bigger does not inherently make them better at following a user's intent.
- Language models generate "whatever they want"
  - There is an infinite number of likely continuations of any text
- A user may want to control attributes of the generated text:
  - vocabulary
  - topic
  - sentiment
  - style
    - e.g. formality, toxicity, simplicity
  - structure
- Some attributes may be implicit
  - e.g. what makes a good summary?

How to control text generation?



Design the prompt

- Heuristic search of prompts
- Gradient-based search
  - Prompt Tuning (Lester et al. 2021)

Tune the model

- Conditional training (e.g. control codes)
  - CTRL (Keskar et al., 2019)
- Steerable layers
  - PPLM (Dathathri et al., 2019)
- Unlikelihood training (Welleck & Kulikov et al. 2019)
- RL fine-tuning
  - e.g. Stiennon et al., 2020

Adjust the decoding process

- Decoding guided by extra features/models
- Constrained decoding
  - e.g. Li et al, 2020
- Trainable decoding
- Combining models
  - GeDi (Krause et al., 2020)

A large review of controlled text generation methods: https://lilianweng.github.io/posts/2021-01-02-controllable-text-generation/

Training with control codes

CTRL: A Conditional Transformer Language Model for Controllable Generation, Keskar et al, 2019

- Prepend all LM training documents with control codes, identifying the source, url, rating, or other attributes of the document
- Control codes learned by the model can be combined e.g. to control language and topic

Horror *A knife* handle pulled through the open hole in the front. I jumped when the knife hit.\n\nEyes widened in horror. Her scream was the only sound I heard besides her sobs.\n\nThe spider touched her feet as it started to dig into the top of her arch. The creature's tears began to flow.\n\nThe spider looked up to her and looked back at me with eyes filled with burning tears. My heart started to race. . .

Reviews *A knife* is a tool and this one does the job well.\n\nRating: 4.0\n\nI bought these for my husband who has been using them to cut up his own meat since he got them. He says they are very sharp so be careful when you use them, but that doesn't seem like much of an issue because he's used it on everything from chicken breasts to beef tenderloin. . .

Diet English : *I lost 10 kgs!* ; German : Ich habe 10 Kilogramm verloren!\n\nEnglish : It is a great achievement for me and my family. ; German : Das ist eine groe Leistung fr mich und meine Familie.\n\nEnglish : Thank you very much to all the people who have helped me. ; German : Vielen Dank an alle, die mir geholfen haben.. . .

Control Code	Description
Wikipedia	English Wikipedia
Books	Books from Project Gutenberg
Reviews	Amazon Reviews data (McAuley et al., 2015)
Links	OpenWebText (See Sec. 3.2)
Translation	WMT translation date (Barrault et al., 2019)
News	News articles from CNN/DailyMail Nallapati et al. (2016), New York Times and Newsroom (Grusky et al., 2018)
multilingual	Wikipedias in German, Spanish and French
Questions	(Questions and answers only) MRQA shared task (See Section 3.1)
Explain	(Only main post) (Fan et al., 2019)
Sub-reddit data (Title, Text and Score/Karma) collected from `pushshift.io`.	
Alone	`r/childfree`
Atheism	`r/atheism`
Christianity	`r/christianity`
Computing	`r/computing`
Thoughts	`r/showerthoughts`
Tip	`r/lifeprotips`
Weight	`r/loseit`
Writing	`r/writingprompts`

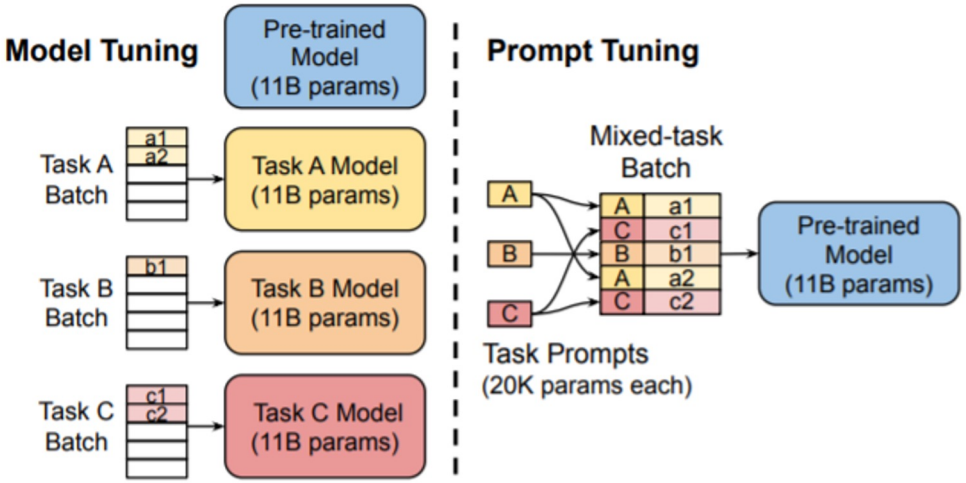
Table 7: Data and control codes. Wikipedia, Books, News and multilingual have no secondary code. `Reviews` can be followed by `Rating:` and a value of {`1.0, 2.0, 3.0, 4.0, 5.0`}. For Links, a full or partial URL can be provided (See Table 3). For all the Reddit data, the secondary code can be `Title:` or `Text:`, which is the title and text of the article, respectively.

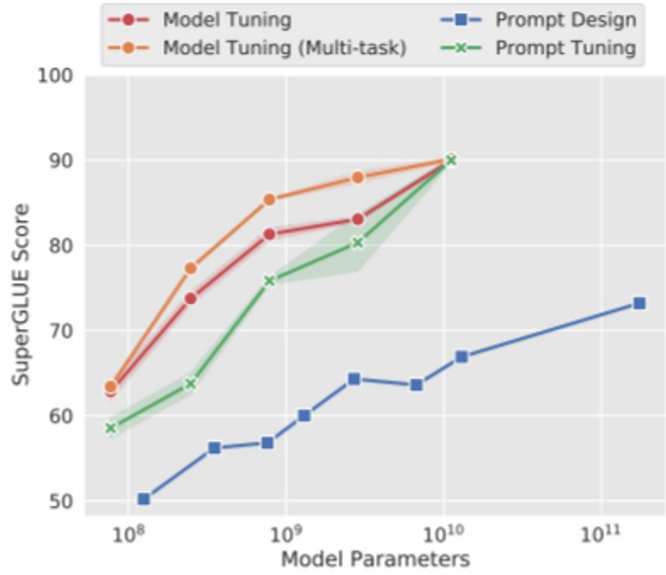
https://arxiv.org/abs/1909.05858

Prompt tuning

The Power of Scale for Parameter-Efficient Prompt Tuning, Lester et al, 2021

- Fine-tuning a small set of parameters inside the model is computationally cheaper
- Prepend 100 random vectors to the zero-layer embeddings of the input texts
- Fine-tune the vectors using back-propagation through the (frozen) model
- Results:
  - With large models, this works as good as full model fine-tuning
  - Performance is more stable to domain shift than with full model fine-tuning



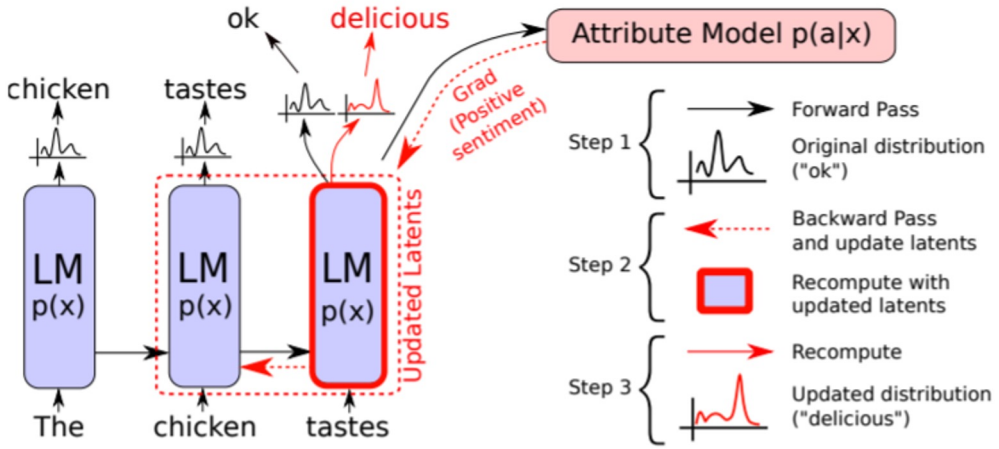


https://arxiv.org/abs/2104.08691

Steering language models: PPLM

- *Plug and Play Language Models: A Simple Approach to Controlled Text Generation, Dathathri, 2019*
- Fine-tuning a small extra set of parameters while the base model stays fixed is computationally cheaper.
- For each output token:
  - Use an external classifier to score the generated text
  - Update the language model hidden states, using the gradient w.r.t. the classifier score
  - Re-generate the last token using the new hidden states

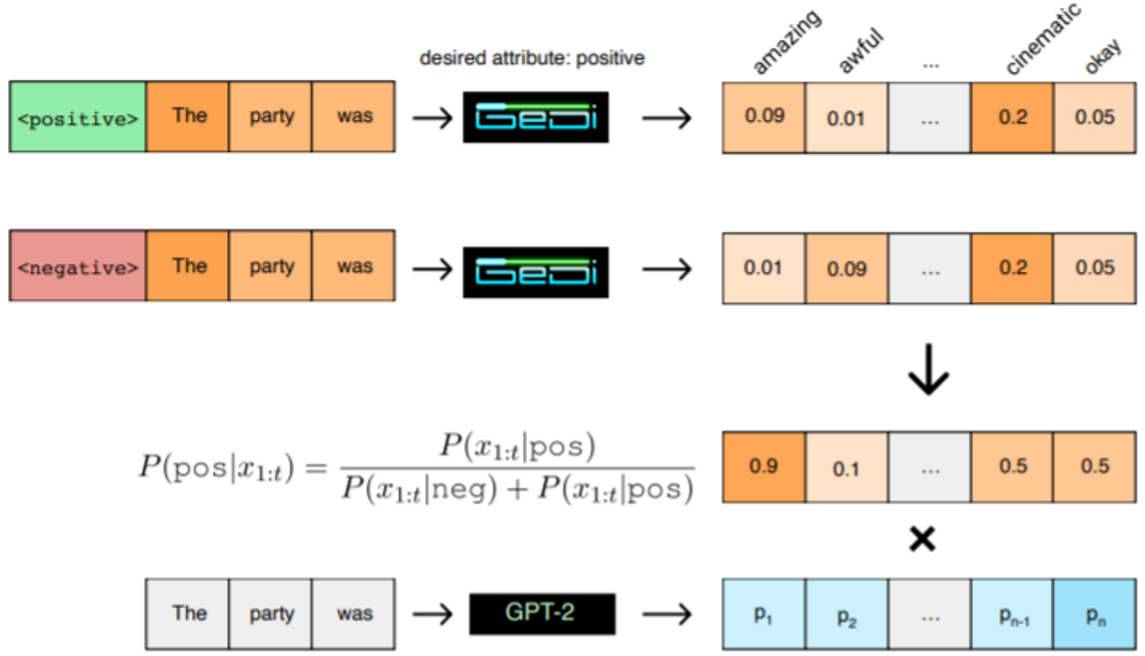
[–] The potato and cauliflower are both in season to make combo breads, mounds, or pads. For an added challenge, try some garlic mashed potatoes.
[Negative] The potato is a pretty bad idea. It can make you fat, it can cause you to have a terrible immune system, and it can even kill you...
[Positive] The potato chip recipe you asked for! We love making these, and I've been doing so for years. I've always had a hard time keeping a recipe secret. I think it's the way our kids love to eat them – so many little ones.
[Science] The potato was once thought to have no health problems and has been promoted as a nutritious food source since the mid-1800s, but recent reports indicate that it has many harmful health issues. In fact, researchers from Johns Hopkins University...
[Politics] [Positive] To conclude this series of articles, I will present three of the most popular and influential works on this topic. The first article deals with the role of women's political participation in building a political system that is representative of the will of the people.



https://arxiv.org/abs/1912.02164

Combining language models: GeDi

- *GeDi: Generative Discriminator Guided Sequence Generation, Krause et al, 2020*
- What: guide the main language model towards the desired ("positive") class
- How: with big LM (generator) use smaller LMs as generative discriminators to guide generation.
- Result: more flexible control than PPLM, and generation is x30 faster



https://arxiv.org/abs/2009.06367

Unlikelihood training

- *Neural Text Generation with Unlikelihood Training, Welleck et al, 2019*
- Problem: generated texts are often dull and repetitive
- Solution: train th

$$\mathcal{L}^t_{\text{UL-token}}(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = -\alpha \cdot \underbrace{\sum_{c \in \mathcal{C}^t} \log(1 - p_\theta(c|x_{<t}))}_{\text{unlikelihood}} - \underbrace{\log p_\theta(x_t|x_{<t})}_{\text{likelihood}}$$

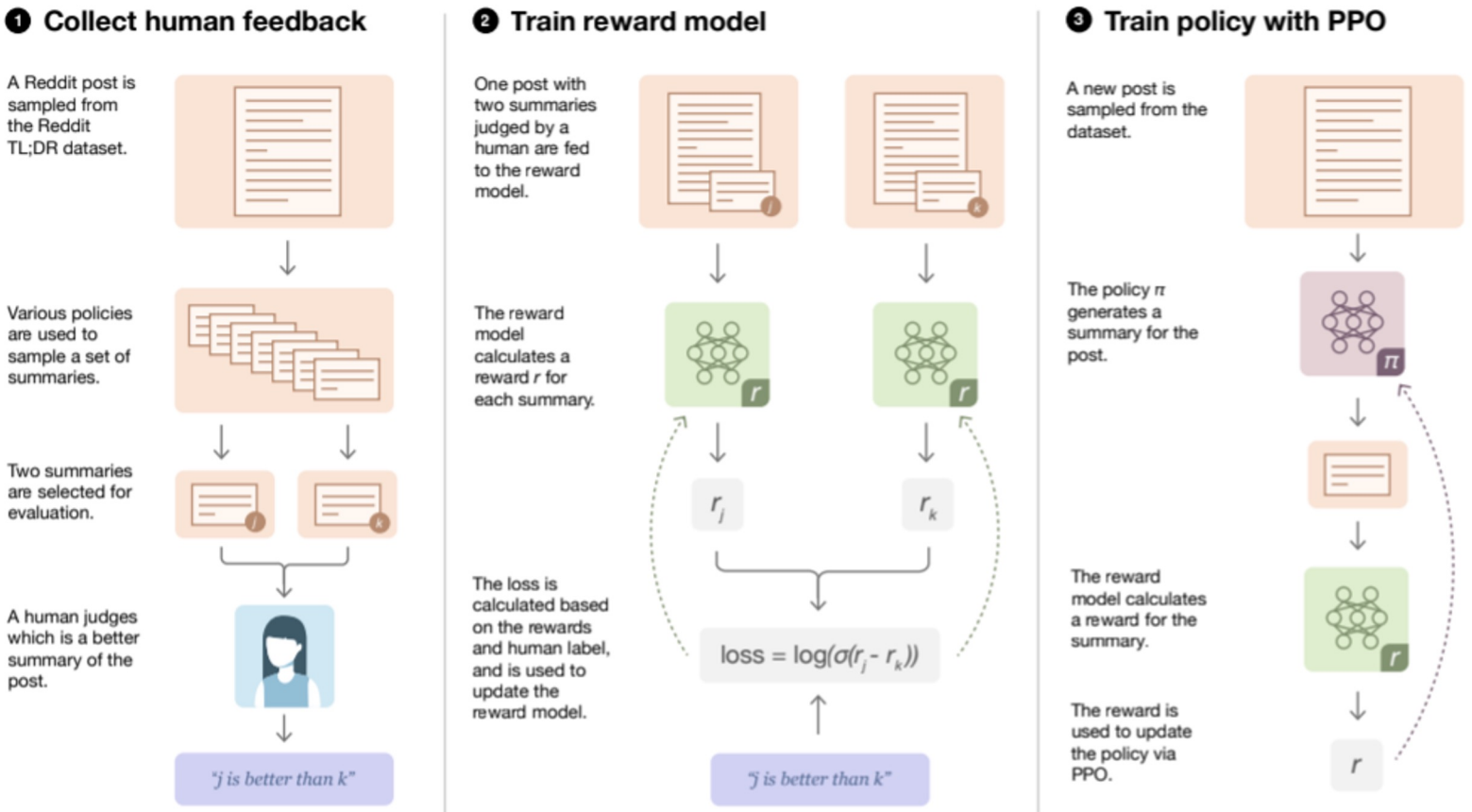
  of "bad" tokens

  or "bad" sequences (decoded negative samples)
  - in this work, "bad" is defined as repeated tokens or token n-grams
- Result: more fluent generated texts, with no harm to LM perplexity

Example of RL fine-tuning

Learning to summarize from human feedback, Stiennon et al, 2020

- Train the reward model to score summaries like humans
- Fine-tune the generator model to increase the probability of summaries with higher scores
- Result: people prefer the generated summaries even to the reference summaries



https://arxiv.org/abs/2009.01325

RL fine-tuning (ChatGPT, InstructGPT)

- The **ChatGPT** is a fine tune of the **GPT-3.5** (text-davinci-003) transform architecture, which belongs to the InstructGPT family of models.
- There is no research paper on the architecture of the **ChatGPT**
- To train the InstructGPT, the Reinforcement Learning with Human Feedback (RLHF) approach is used towards understanding more complex user requests / instructions
- Example of instruction of **InstructGPT**:

Table 1: Distribution of use case categories from our API prompt dataset.

Use-case	**(%)**
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

Table 2: Illustrative prompts from our API prompt dataset. These are fictional examples inspired by real usage—see more examples in Appendix A.2.1.

Use-case	**Prompt**
Brainstorming	List five ideas for how to regain enthusiasm for my career
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.
Rewrite	This is the summary of a Broadway play: """ {summary} """ This is the outline of the commercial for that play: """

InstructGPT (a sibling of ChatGPT)

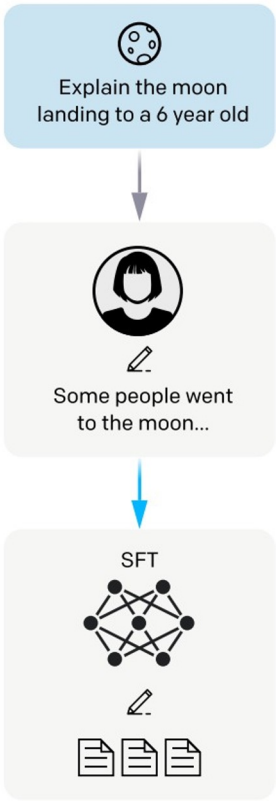
Step 1

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.



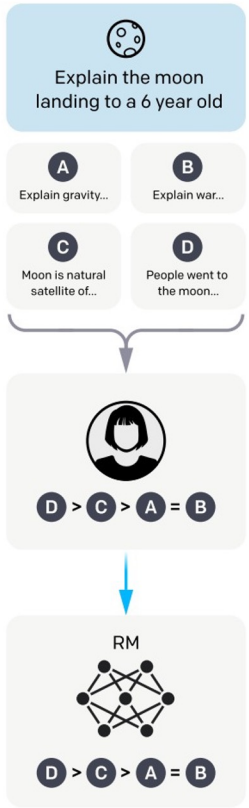
Step 2

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.



Step 3

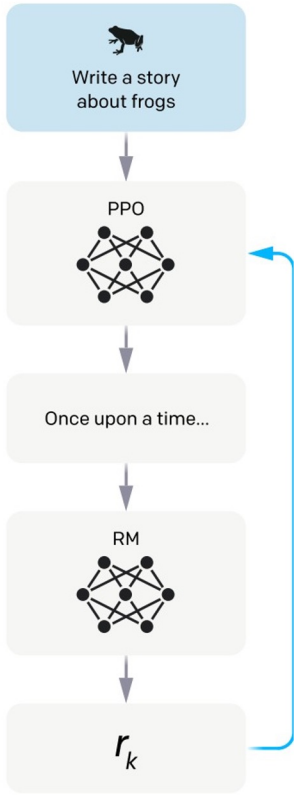
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Training language models to follow instructions with human feedback. Ouyang et al., 2022

Based on https://youtu.be/2Jxcly7AgFQ

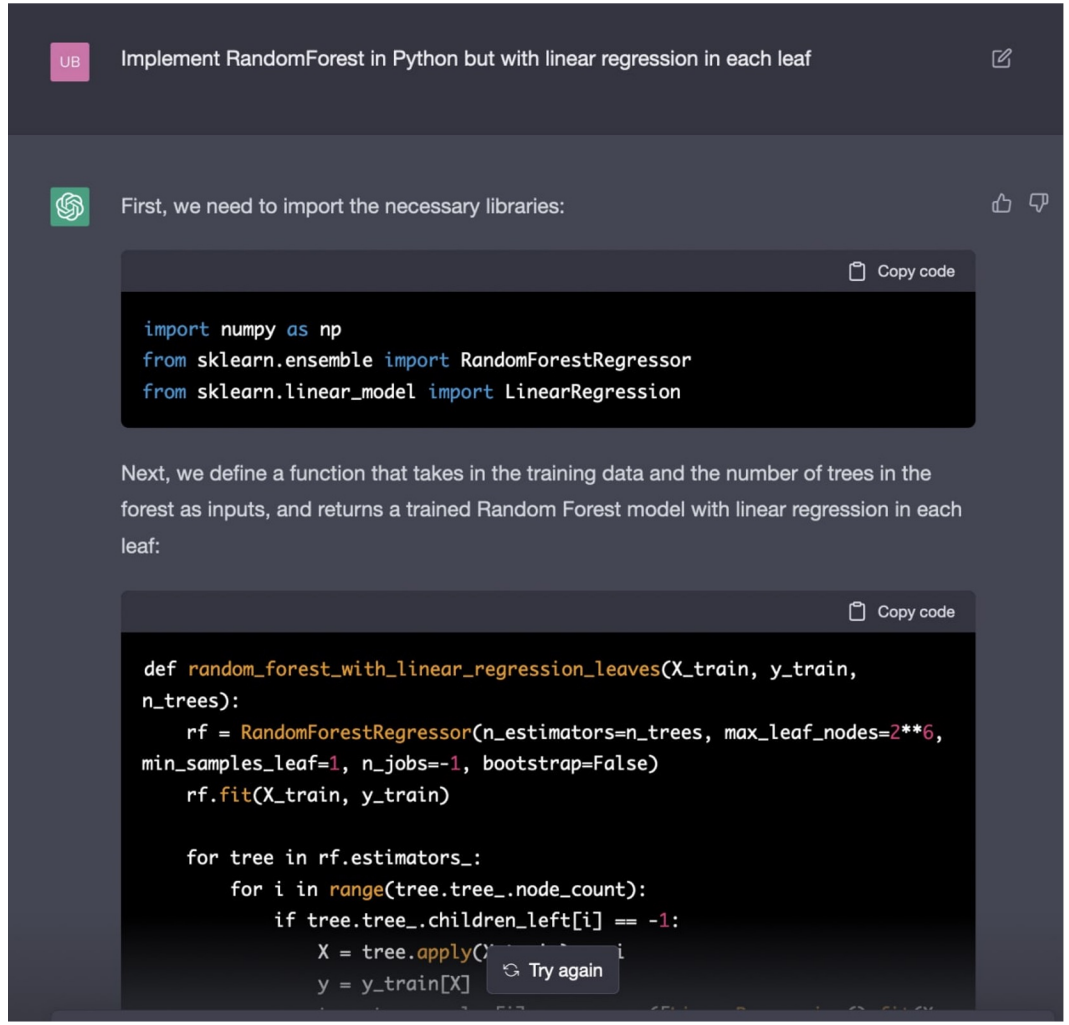
ChatGPT vs InstructGPT

- Similar setup to InstructGPT, but “slight differences in data collection”
- Trade in-context learning for dialog history modeling (tracking context)

Ability	OpenAI Model	Training Method	OpenAI API	OpenAI Paper	Open Source Approximate
+ Follow human value + More detailed generation + in-context learning + zero-shot generation	Instruct-GPT RLHF **More aligned than 002, less performance loss	Instruction tuning w. RLHF	Text-Davinci-003	Instruct-GPT paper, RLHF part Summarization .w human feedback	DeepMind Sparrow paper AI2 RL4LMs
++ Follow human value ++ More detailed generation ++ Reject questions beyond its knowledge (why?) ++ Model dialog context -- In-context learning	ChatGPT ** Trade in-context learning for dialog history modeling	Tuning on dialog w. RLHF	-	-	DeepMind Sparrow paper AI2 RL4LMs

yaofu.notion.site/How-does-GPT-Obtain-its-Ability-Tracing-Emergent-Abilities-of-Language-Models-to-their-Sources-b9a57ac0fcf74f30a1ab9e3e36fa1dc1

Based on https://youtu.be/2Jxcly7AgFQ



Перефразируй текст ниже с использованием научных терминов, как будто это докторская

В этом посте мы хотим разобрать (и разобраться сами), как можно использовать и объединять генеративные (и не только) модели для решения одной большой задачи — генерации контента. Мы подобрали большое количество материалов про каждую конкретную модель, чтобы ими можно было пользоваться сразу.

В настоящем исследовании мы рассматриваем возможность использования и комбинирования генеративных (и других) моделей для решения конкретной практической задачи - генерации контента. Для углубленного анализа были собраны многочисленные материалы, которые позволят использовать разнообразные модели в практических целях.

Summary

- Autoregressive language modeling is a useful pretraining task
- GPT and other large transformer language models generalize well for various NLP problems
- Standard methods of text decoding provide control of quality or diversity of generated texts
- New and more precise methods of controllable text generation are being developed