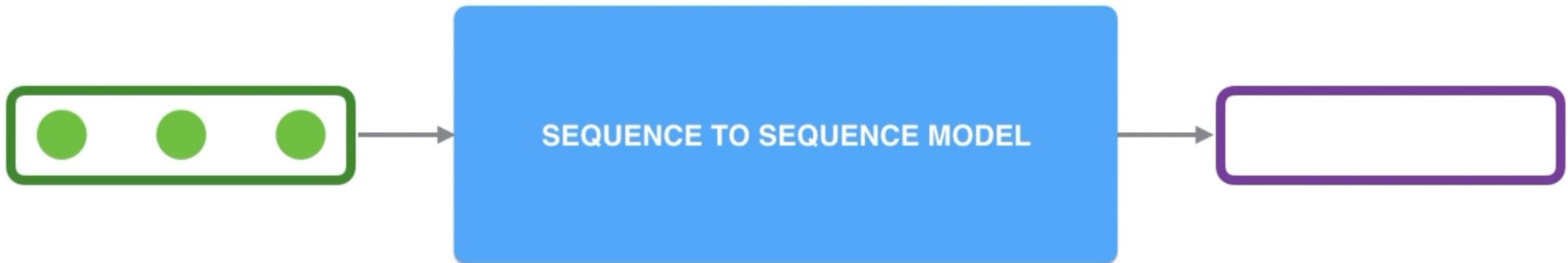
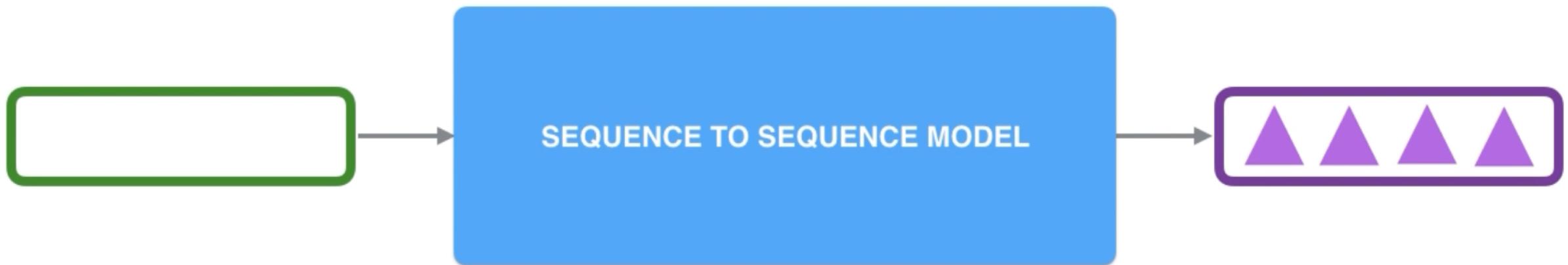


Трансформеры (I): GPT-n, BERT, BART, T5...

Recap

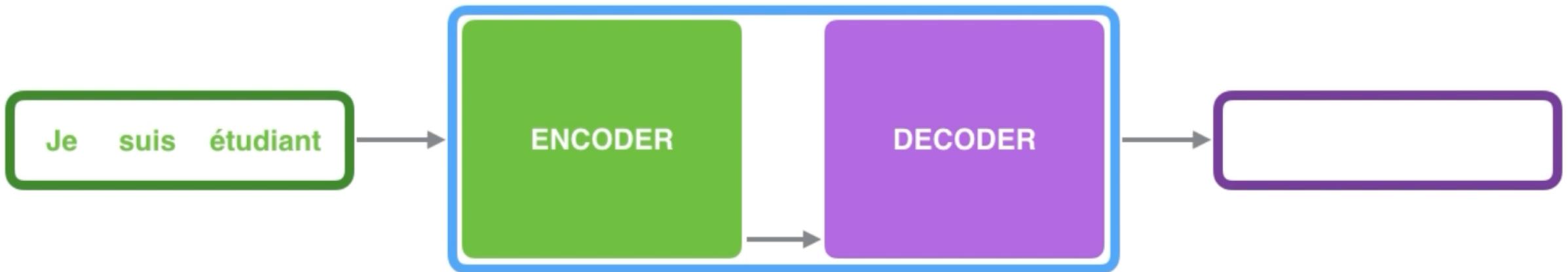


Recap

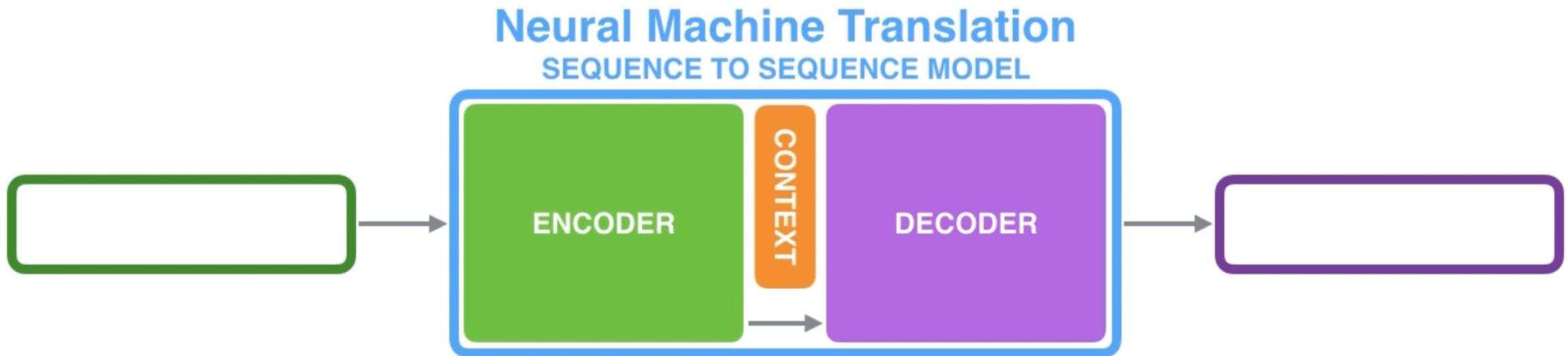


Recap

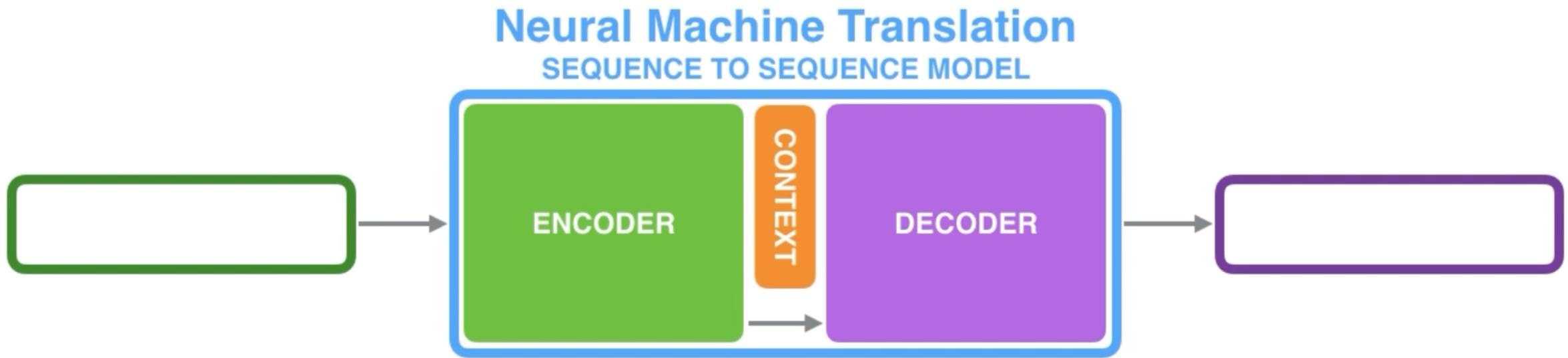
Neural Machine Translation SEQUENCE TO SEQUENCE MODEL



Recap



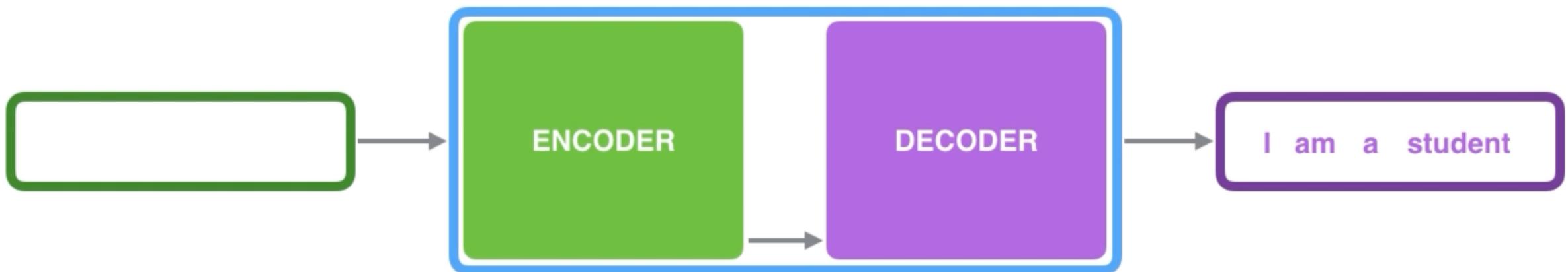
Recap



Применительно к машинному переводу контекст представляет собой вектор (массив чисел), а энкодер и декодер, в свою очередь, чаще всего являются рекуррентными нейронными сетями

Recap

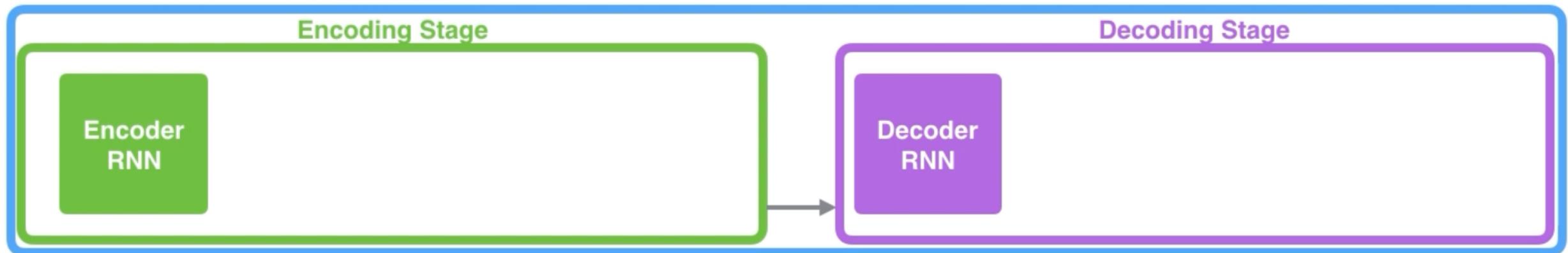
Neural Machine Translation SEQUENCE TO SEQUENCE MODEL



Recap

Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL

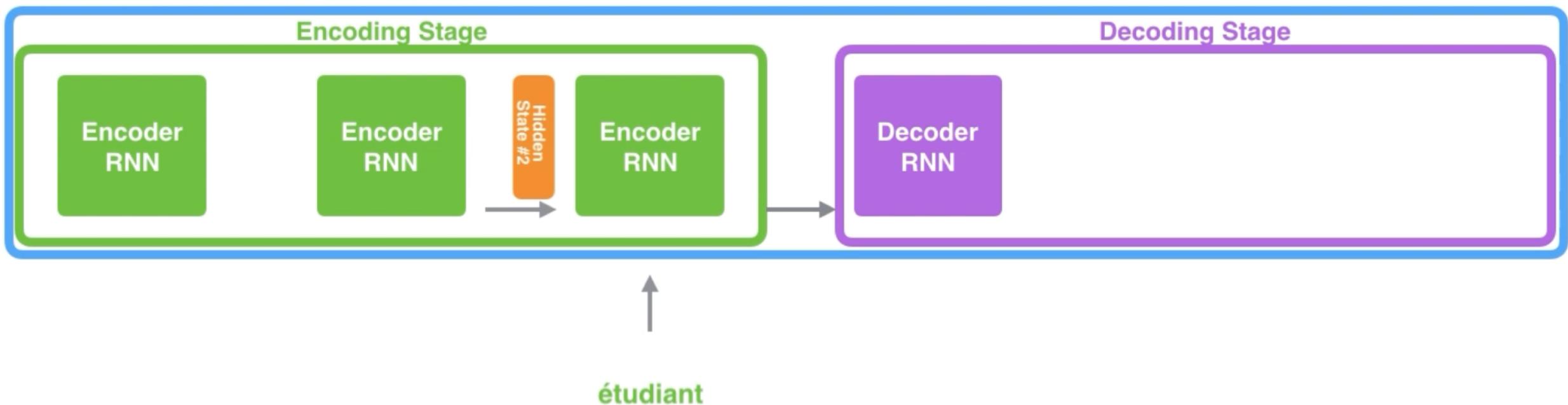


Je suis étudiant

Unrolled представление, где вместо того, чтобы показывать один декодер, мы показываем его копию для каждого временного отрезка

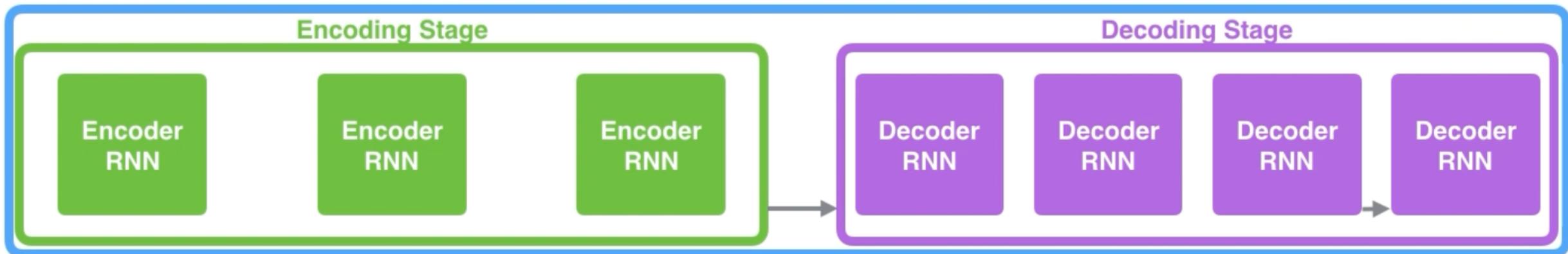
Recap

Neural Machine Translation SEQUENCE TO SEQUENCE MODEL



Recap

Neural Machine Translation SEQUENCE TO SEQUENCE MODEL



Контекстный вектор является бутылочным горлышком для такого типа моделей, благодаря чему, им сложно иметь дело с длинными предложениями.

Решение было предложено в статьях [Bahdanau et al., 2014](#) и [Luong et al., 2015](#), где была представлена техника, получившая название «механизм внимания» (attention). Данный механизм значительно улучшает качество систем машинного перевода, позволяя моделям концентрироваться на релевантных частях входных последовательностей.

Recap

Энкодер передает значительно больше данных декодеру: вместо передачи лишь последнего скрытого состояния после этапа кодирования, энкодер отправляет ему все свои скрытые состояния



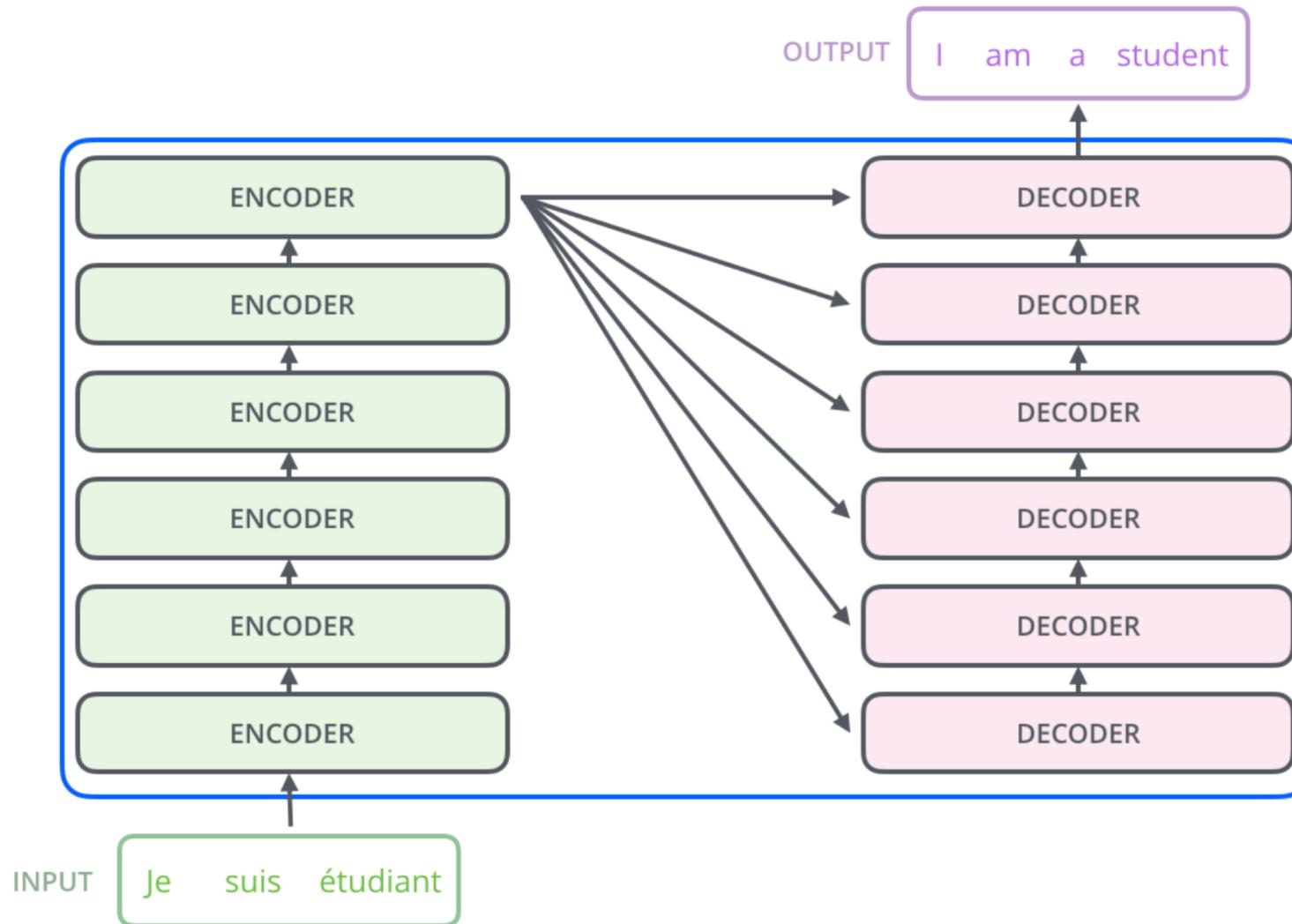
Декодер проходит через дополнительный этап, прежде чем сгенерировать выход. Для того, чтобы сфокусироваться на тех частях входной последовательности, которые релевантны для соответствующего временного отрезка, декодер делает следующее:

1. Сматривает на набор скрытых состояний, полученных от энкодера – каждое из скрытых состояний соотносится наилучшим образом с одним из слов в входной последовательности;
2. Назначает каждому скрытому состоянию некую оценку;
3. Умножает каждое скрытое состояние на преобразованную softmax функцией оценку, выделяя, таким образом, скрытые состояния с большой оценкой и отводя на второй план скрытые состояния с маленькой.

Recap



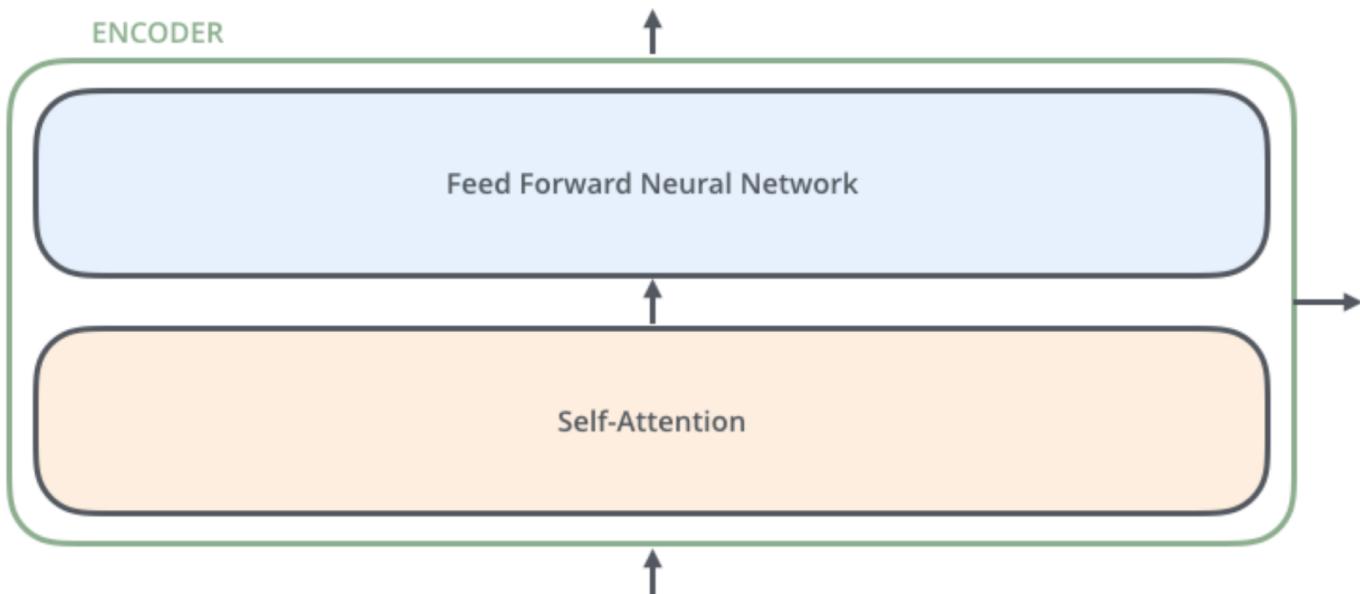
Recap



Кодирующий компонент – это стек энкодеров; в иллюстрации ниже мы изобразили 6 энкодеров, расположенных друг над другом (можно любое кол-во). Декодирующий компонент – это стек декодеров, представленных в том же количестве.

Recap

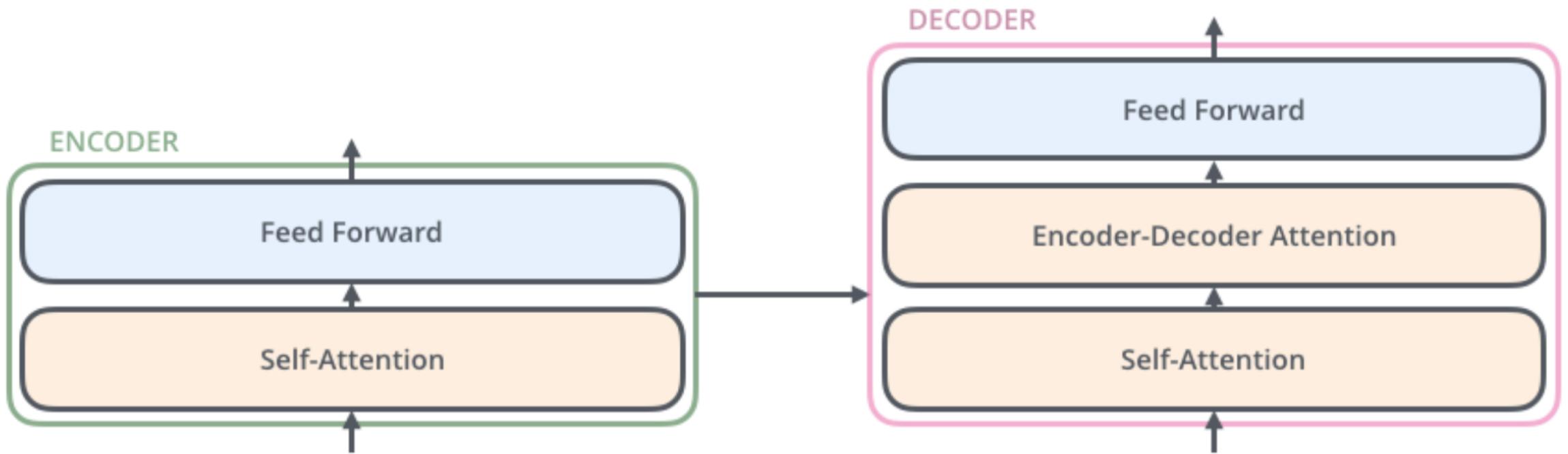
Все энкодеры идентичны по структуре, хотя и имеют разные веса. Каждый можно разделить на два подслоя:



Входная последовательность, поступающая в энкодер, сначала проходит через слой внутреннего внимания (self-attention), помогающий энкодеру посмотреть на другие слова во входящем предложении во время кодирования конкретного слова. Мы рассмотрим этот механизм далее в статье.

Выход слоя внутреннего внимания отправляется в нейронную сеть прямого распространения (feed-forward neural network). Точно такая же сеть независимо применяется для каждого слова в предложении.

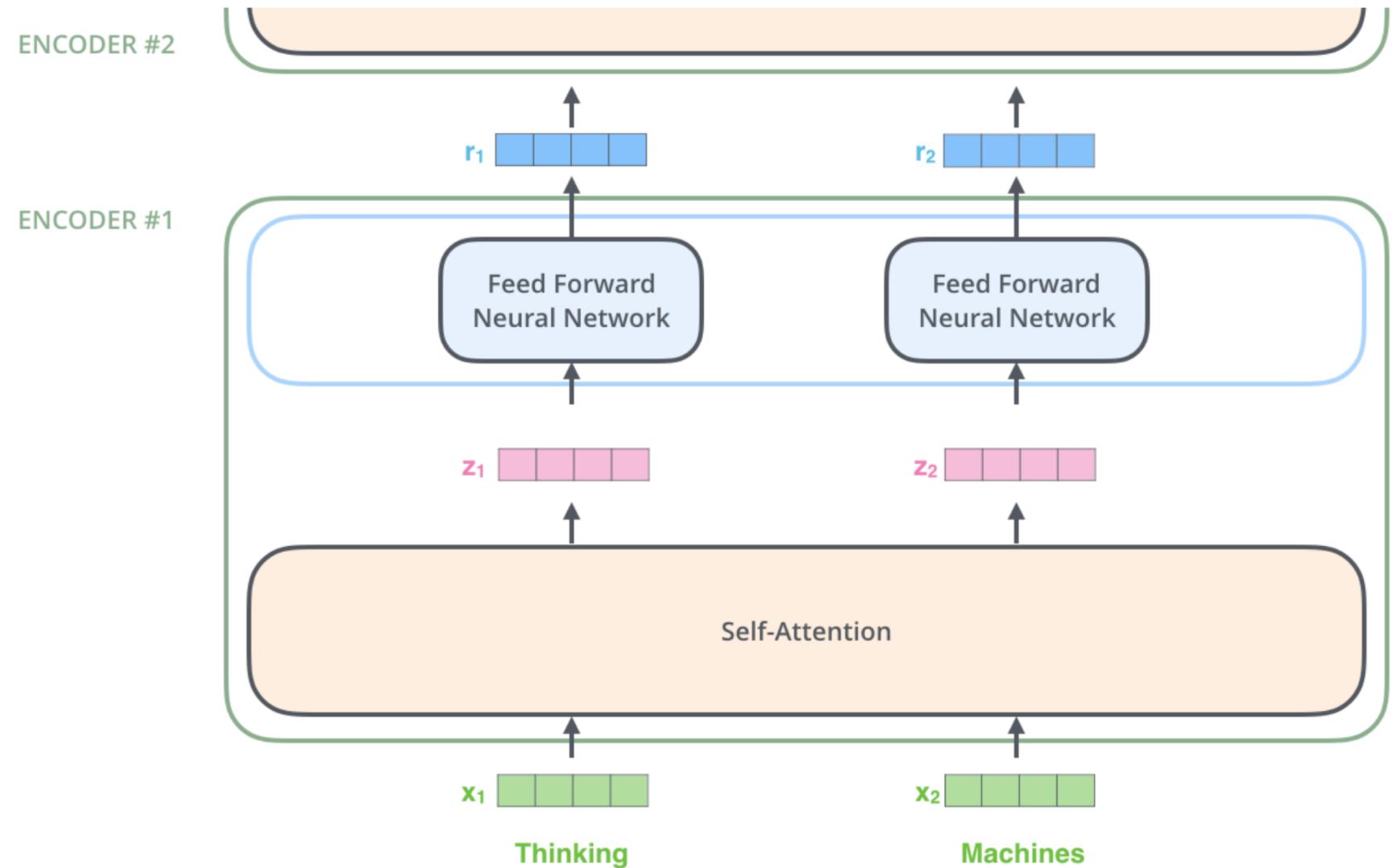
Recap



Декодер также содержит два этих слоя, но между ними есть слой внимания, который помогает декодеру фокусироваться на релевантных частях входящего предложения

Recap

Энкодер получает на вход и обрабатывает набор векторов, проводя их через слой внимания и далее – через нейронную сеть прямого распространения, пока, наконец, не передает свой выход следующему энкодеру



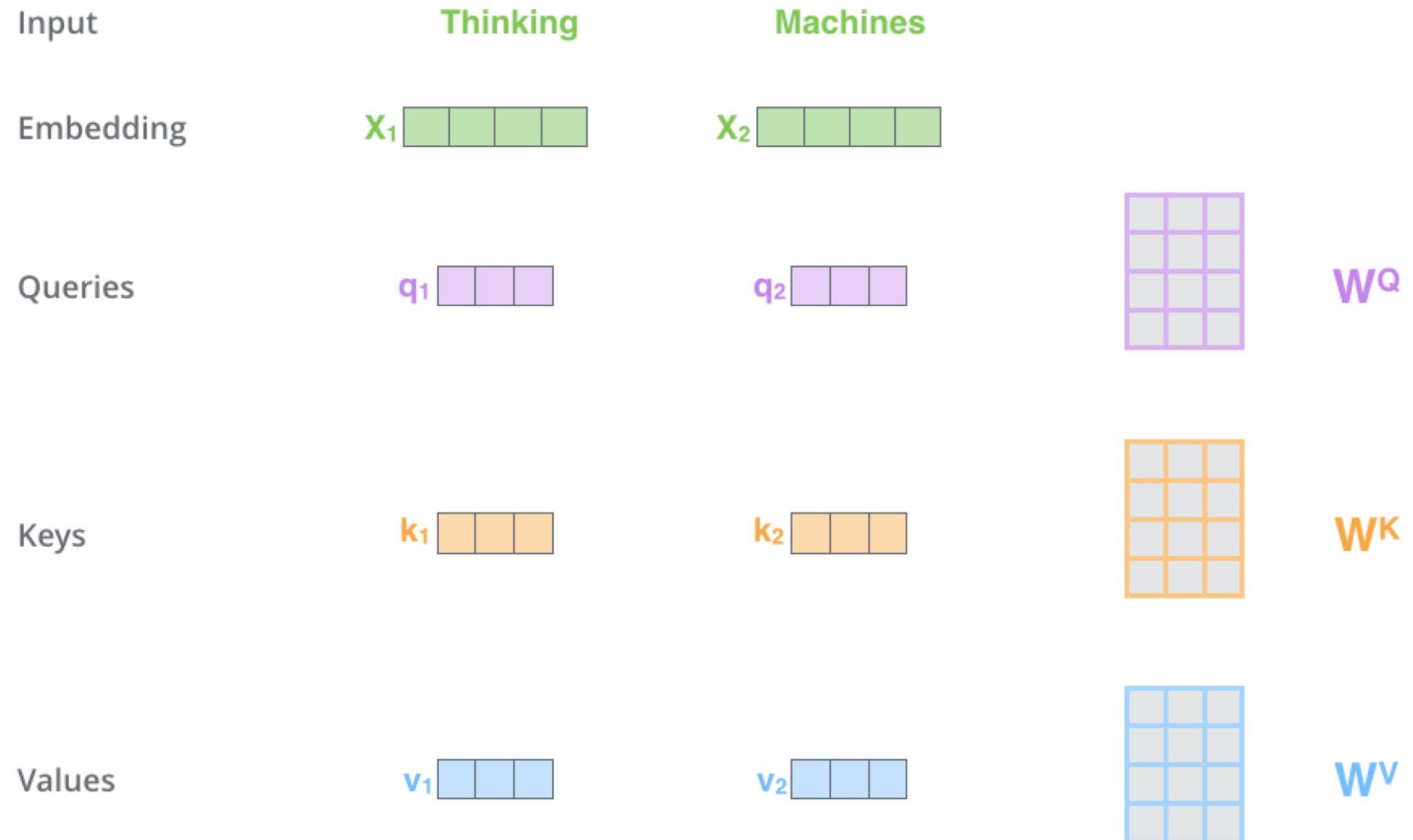
Recap

"The animal didn't cross the street because **it** was too tired"

К чему относится «it» в этом предложении? К улице (street) или к животному (animal)? Простой вопрос для человека становится целой проблемой для алгоритма.

Recap – Calculate Attention – Step I

Сначала создаем 3 вектора из каждого входящего вектора (в нашем случае – эмбеддинга каждого слова): вектор запроса (Query vector), вектор ключа (Key vector) и вектор значения (Value vector). Эти векторы создаются с помощью перемножения эмбеддинга на три матрицы, которые мы обучили во время процесса обучения.



Умножение x_1 на матрицу весов WQ производит q_1 , вектор «запроса», относящийся к этому слову. В итоге мы создаем проекции «запроса», «ключа» и «значения» для каждого слова во входящем предложении.

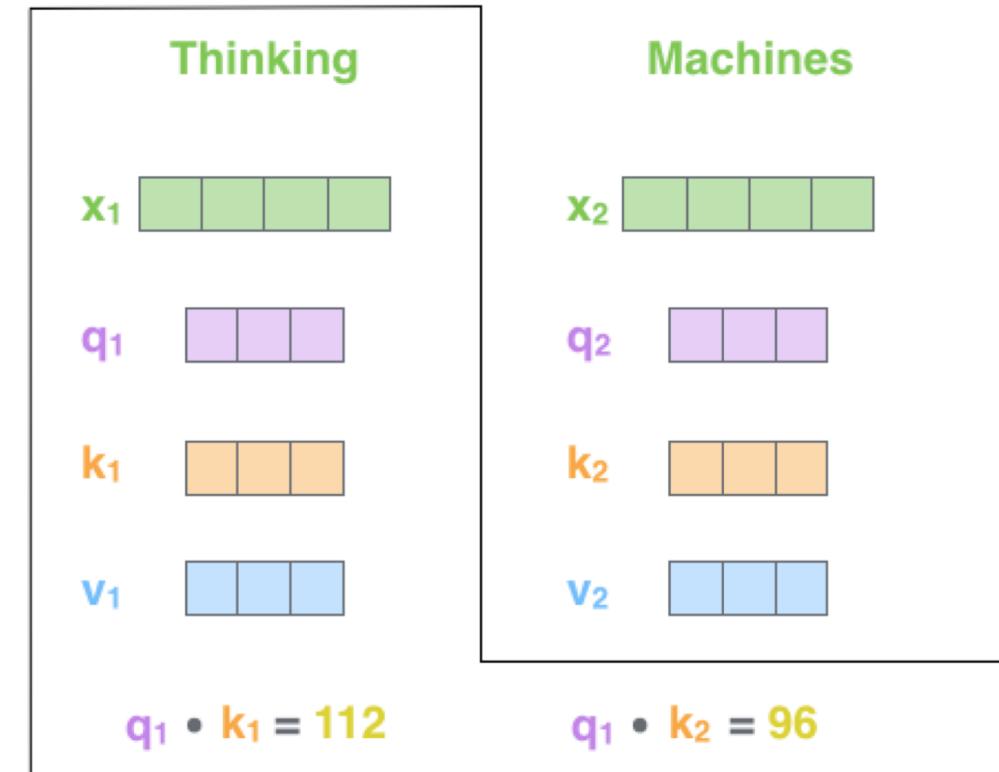
Recap – Calculate Attention – Step II

Подсчитываем внутреннее внимание для первого слова в нашем примере – «Thinking». Нам нужно оценить каждое слово во входящем предложении по отношению к данному слову.

Коэффициент определяет, насколько нужно сфокусироваться на других частях входящего предложения во время кодирования слова в конкретной позиции.

Коэффициент подсчитывается с помощью скалярного произведения вектора запроса и вектора ключа соответствующего слова. Таким образом, если мы вычисляем внутреннее внимание для слова в позиции #1, первый коэффициент будет скалярным произведением q_1 и k_1 , второй — скалярным произведением q_1 и k_2 .

Input
Embedding
Queries
Keys
Values
Score



Recap – Calculate Attention – Step III

Input			
Embedding	Thinking	Machines	
Queries	x_1	x_2	
Keys	q_1	q_2	
Values	k_1	k_2	
Score	$q_1 \cdot k_1 = 112$	$q_1 \cdot k_2 = 96$	
Divide by 8 ($\sqrt{d_k}$)	14	12	
Softmax	0.88	0.12	

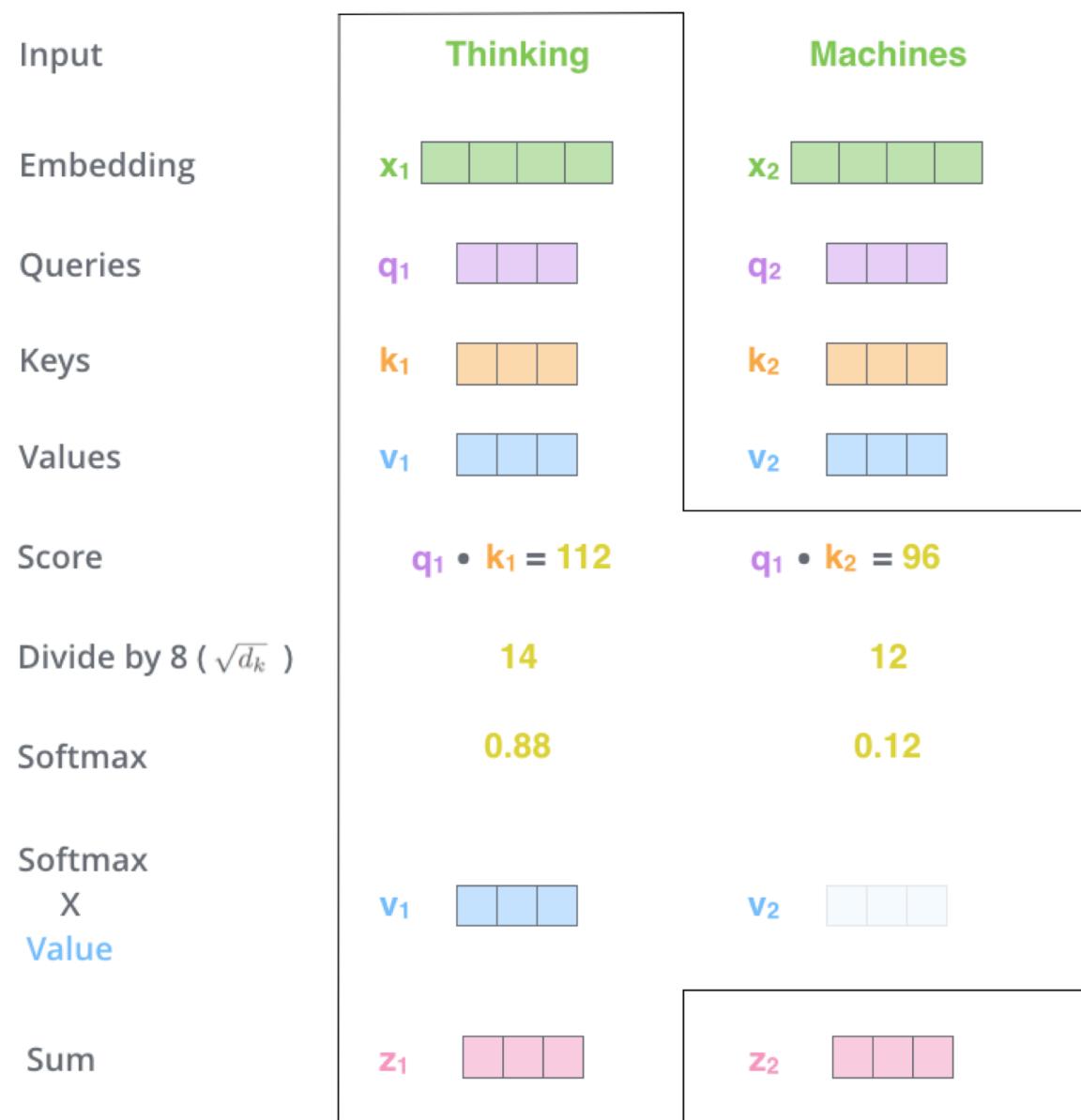
Разделим эти коэффициенты на 8 (квадратный корень размерности векторов ключа, используемой в статье – 64; данное значение обеспечивает более стабильные градиенты и используется по умолчанию, но возможны также и другие значения), а затем пропустить результат через функцию softmax. Данная функция нормализует коэффициенты так, чтобы они были положительными и в сумме давали 1. Имеем softmax score

Recap – Calculate Attention – Step IV

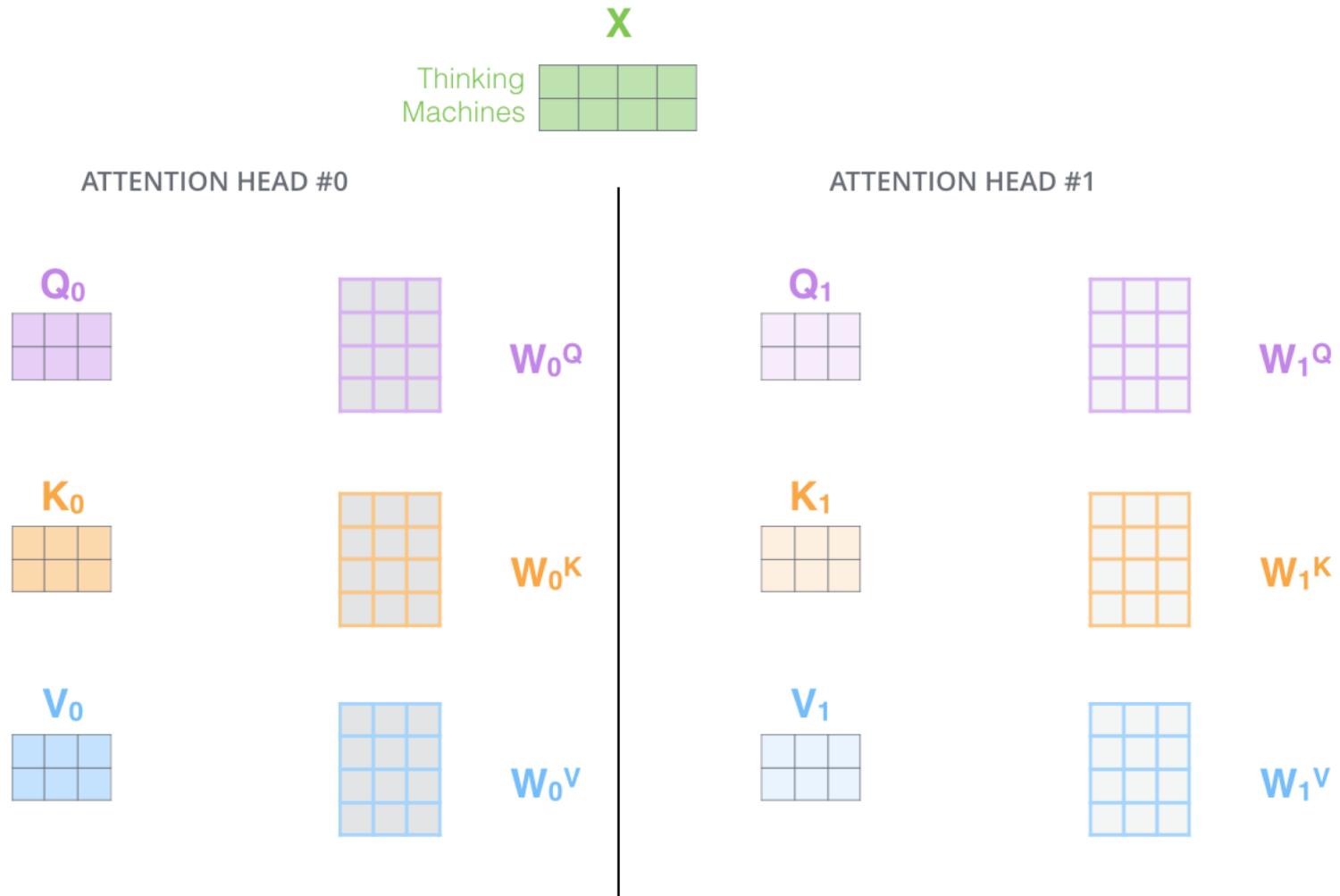
Умножаем каждый вектор значения на softmax-коэффициент (перед их сложением). Интуиция здесь следующая: нужно держать без изменений значения слов, на которых мы фокусируемся, и отвести на второй план нерелевантные слова (умножив их на небольшие значения, например, 0.001).

Складываем взвешенные векторы значения. Это и будет представлять собой выход слова внутреннего внимания в данной позиции (для первого слова).

В результате мы получаем вектор, который можем передавать дальше в нейронную сеть прямого распространения. В настоящих реализациях, однако, эти вычисления делаются в матричной форме для более быстрой обработки.



Recap

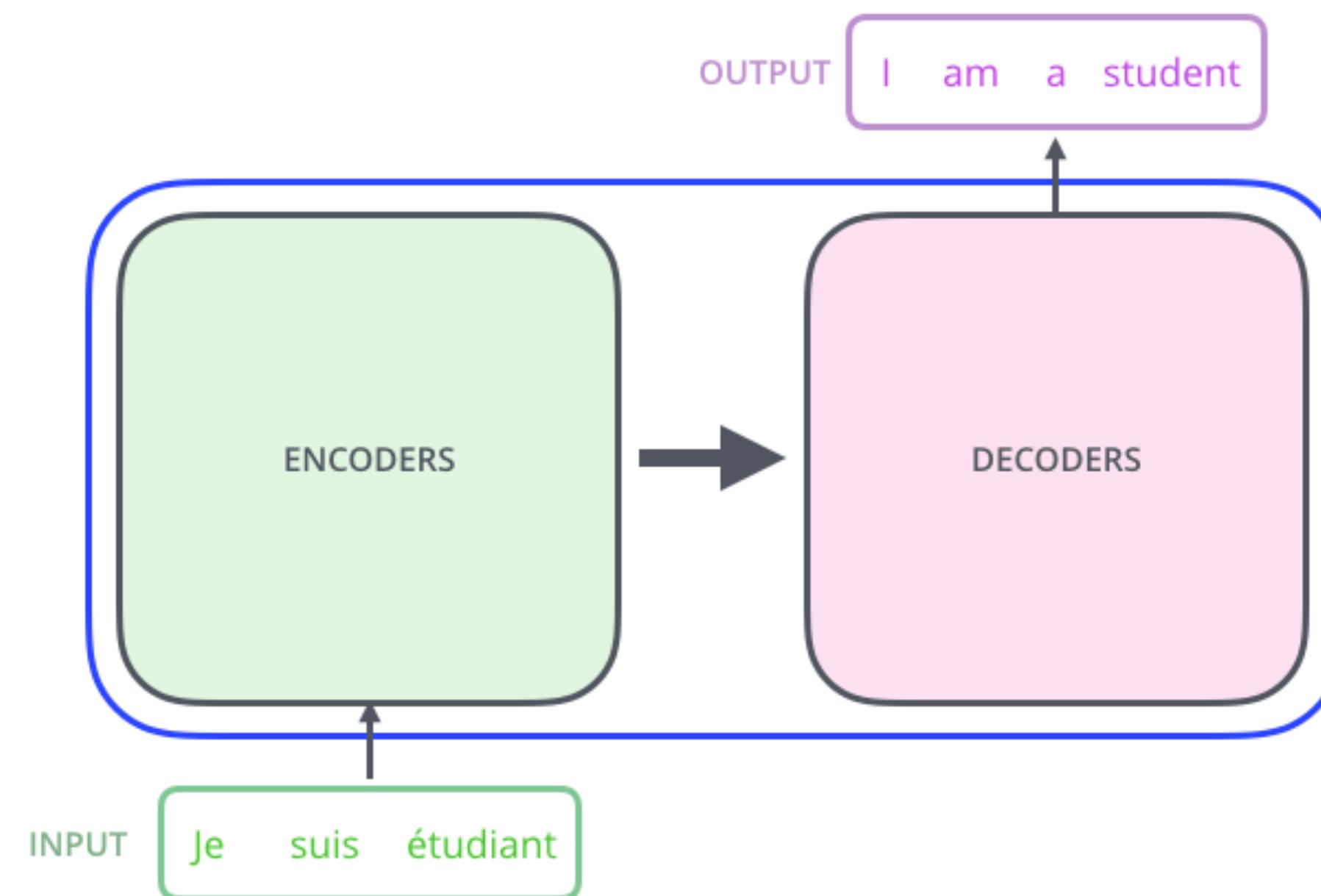


В случае множественного внимания, мы располагаем отдельными $WQ/WK/WV$ матрицами весов для каждой «головы», что в результате дает разные $Q/K/V$ матрицы. Как мы делали ранее, умножаем X на $WQ/WK/WV$ матрицы для получения $Q/K/V$ матриц.

Pre-trained Transformer

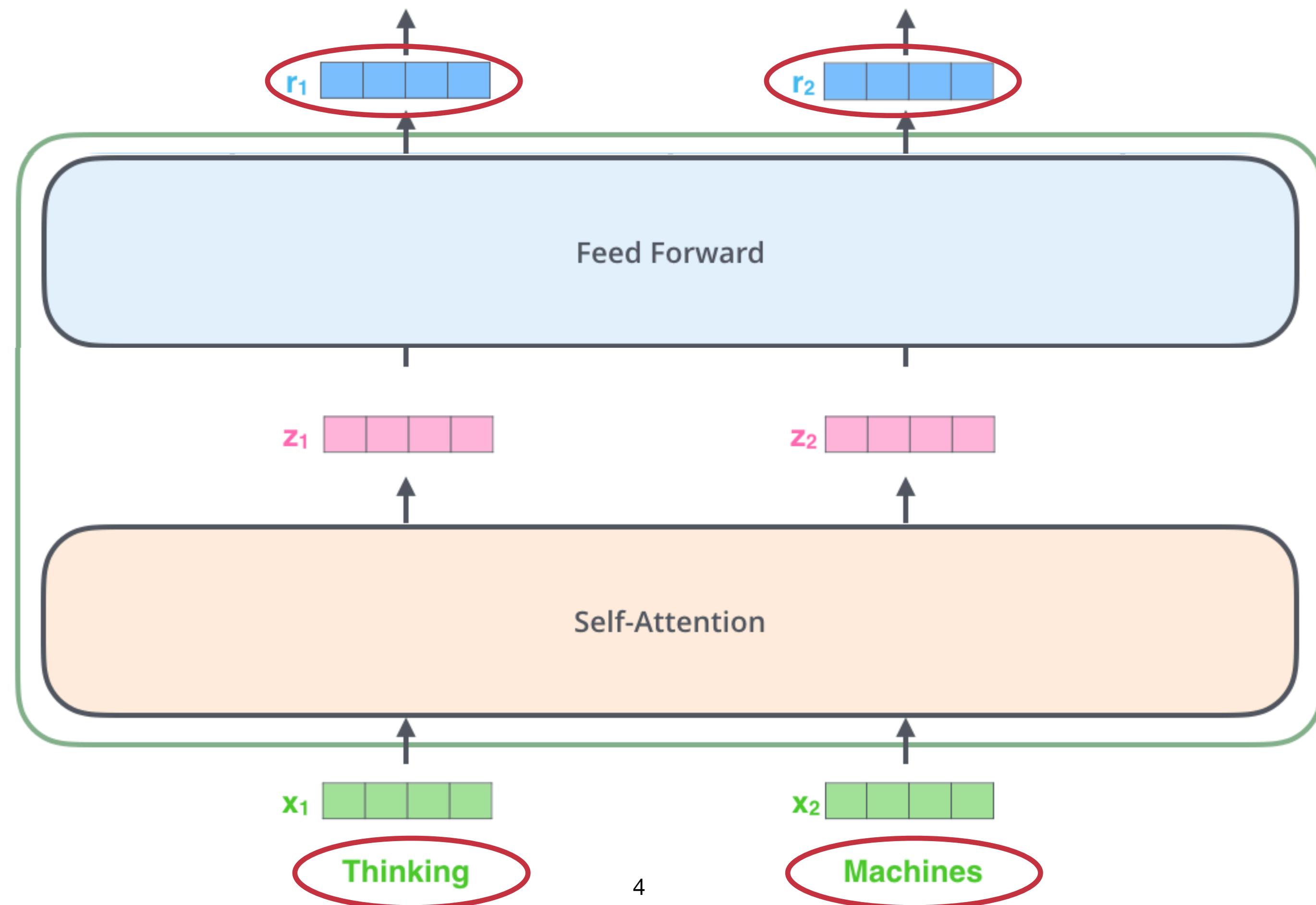
Если есть обученный для перевода Transformer, то:

- **Encoder:** выучил хорошие признаки для слов на языке входа
- **Decoder:** выучил хорошие признаки для слов на языке выхода



Pre-trained Transformer

Для каждого слова Transformer block выдает эмбеддинг, зависящий от всего контекста



Pre-trained Transformer

Inference:

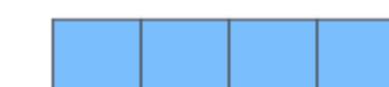
*A **plane** crash*



Transformer



*“plane” embedding
(in context 1)*



*A **plane** surface*



Transformer



*“plane” embedding
(in context 2)*



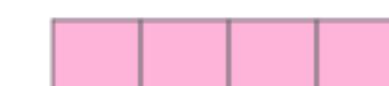
*A **plane** crash /
A **plane** surface*



Word2Vec



*“plane” embedding
(equal for both contexts)*

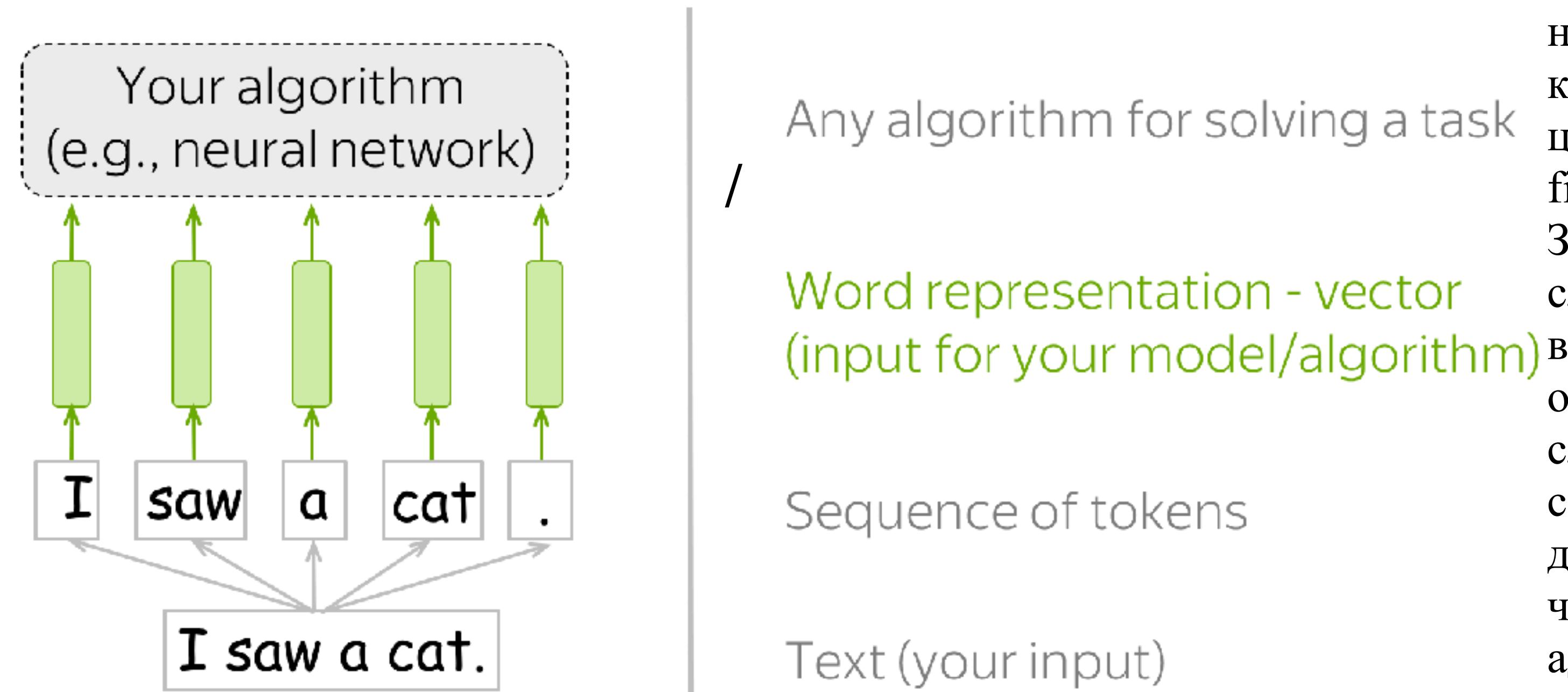


Pre-trained Transformer

Основная идея: взять предобученные эмбеддинги и дообучить на нужную задачу

Pre-train + Fine-tune (Transfer Learning)

Transfer learning (перенос обучения) — это подход в машинном обучении, при котором модель, обученная на одной задаче, используется для решения другой, связанной задачи.



Fine-tuning (дообучение) — это процесс адаптации предварительно обученной модели к новой, часто более специфичной задаче или набору данных, разновидность transfer learning. Как работает: получаем предобученную модель, адаптируем (fine-tune) с помощью небольшого набора данных, который имеет отношение к целевой задаче. В процессе fine-tuning можно: Замораживать некоторые слои модели, оставляя их веса фиксированными, и обучать только верхние слои, которые отвечают за специфику новой задачи, дообучать все слои модели, чтобы она могла адаптировать свои представления под специфическую задачу. Это требует больше данных и вычислительных ресурсов.

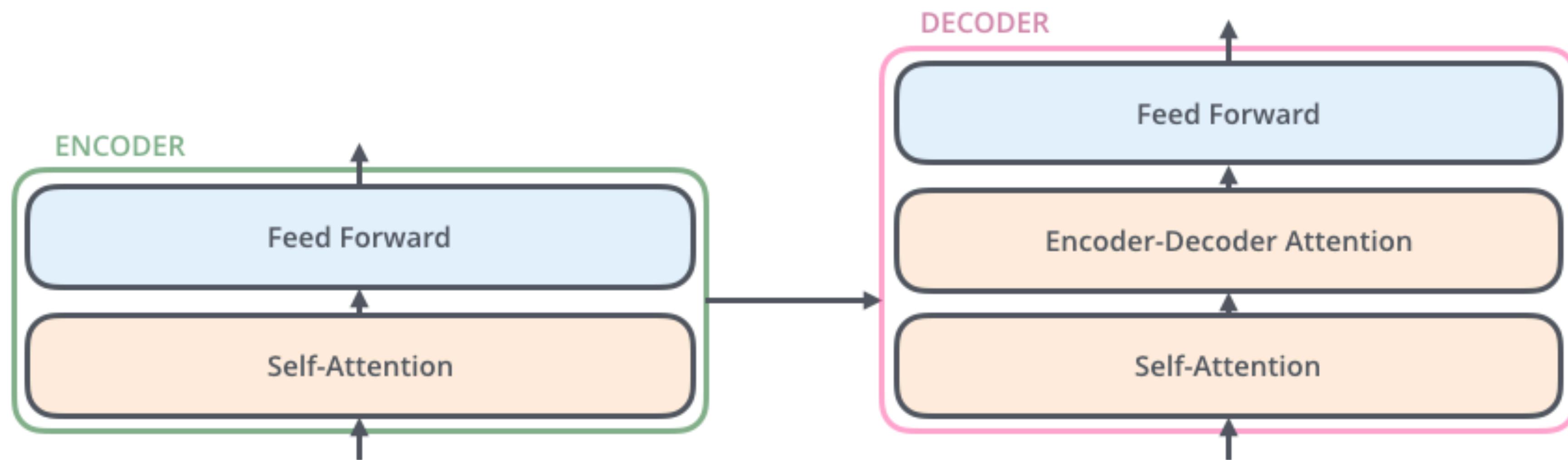
[Image credit](#)

GPT

GPT

Generative Pre-Training

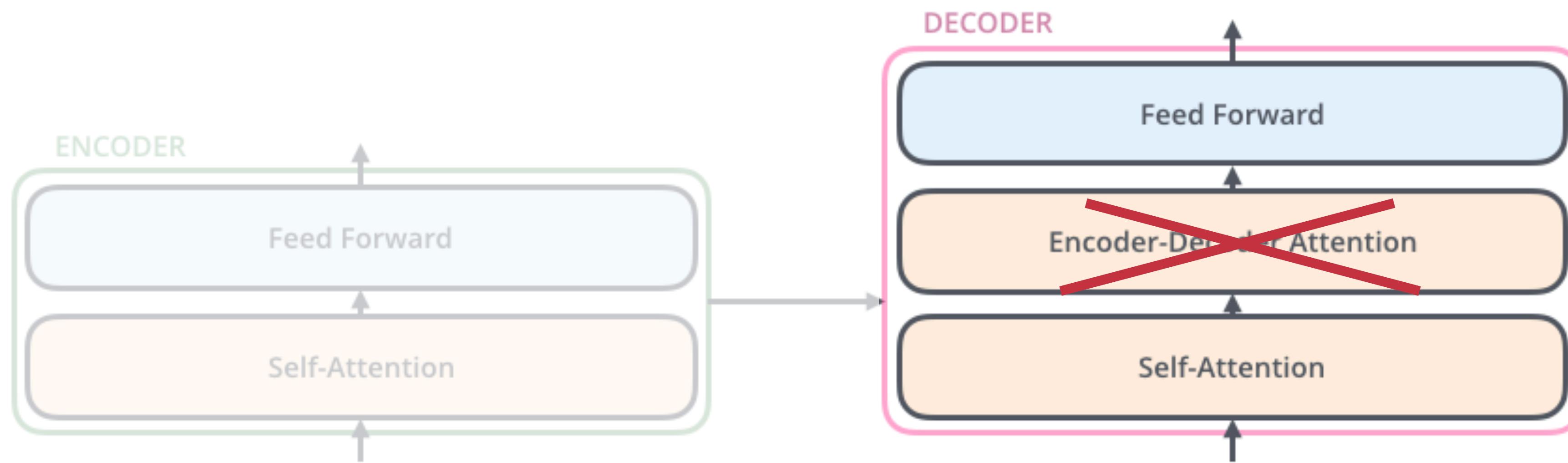
Архитектура: Transformer Decoder



GPT

Generative Pre-Training

Архитектура: Transformer Decoder

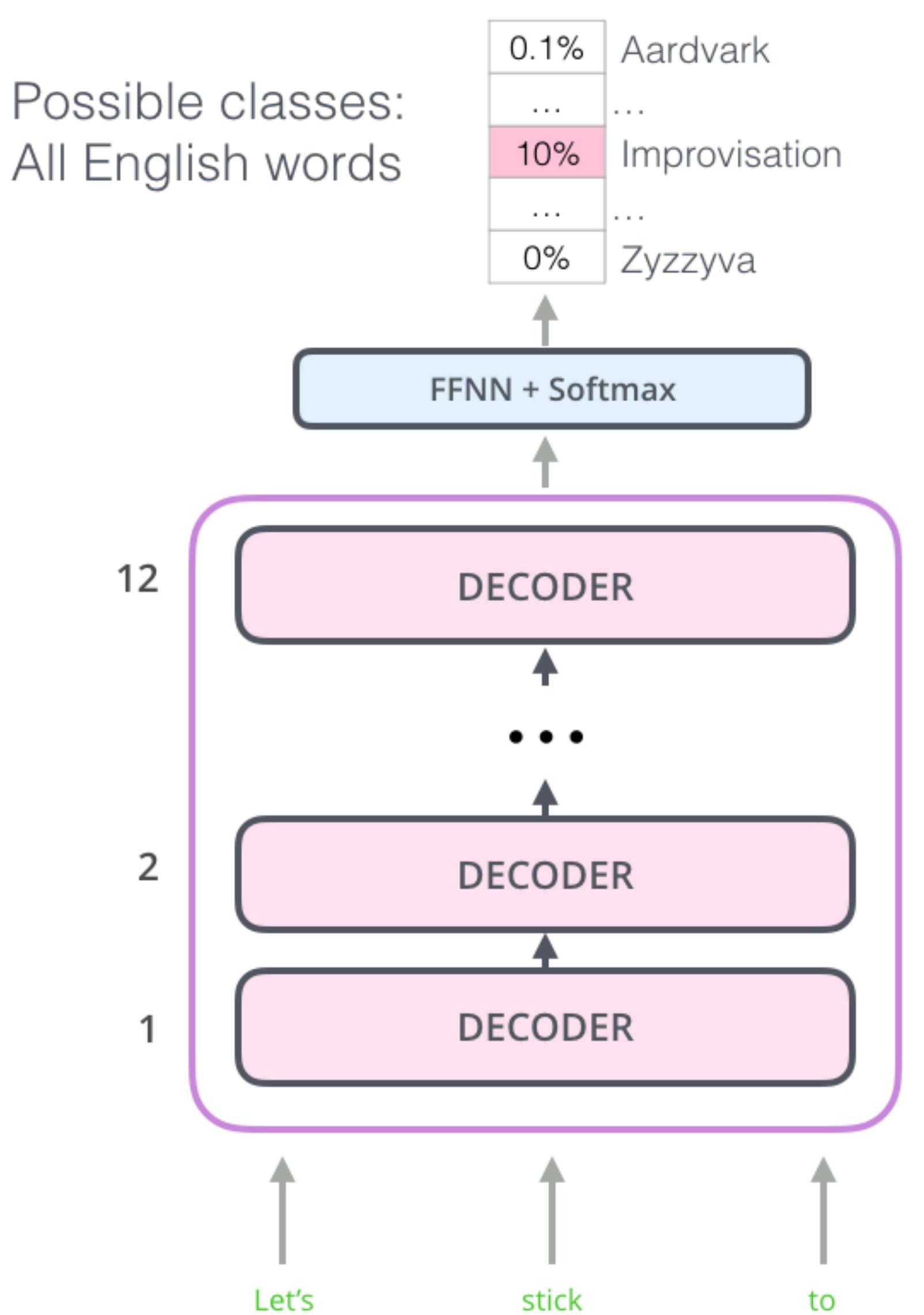


[Image credit](#)

GPT

Generative Pre-Training

Архитектура: Transformer Decoder



GPT

Generative Pre-Training

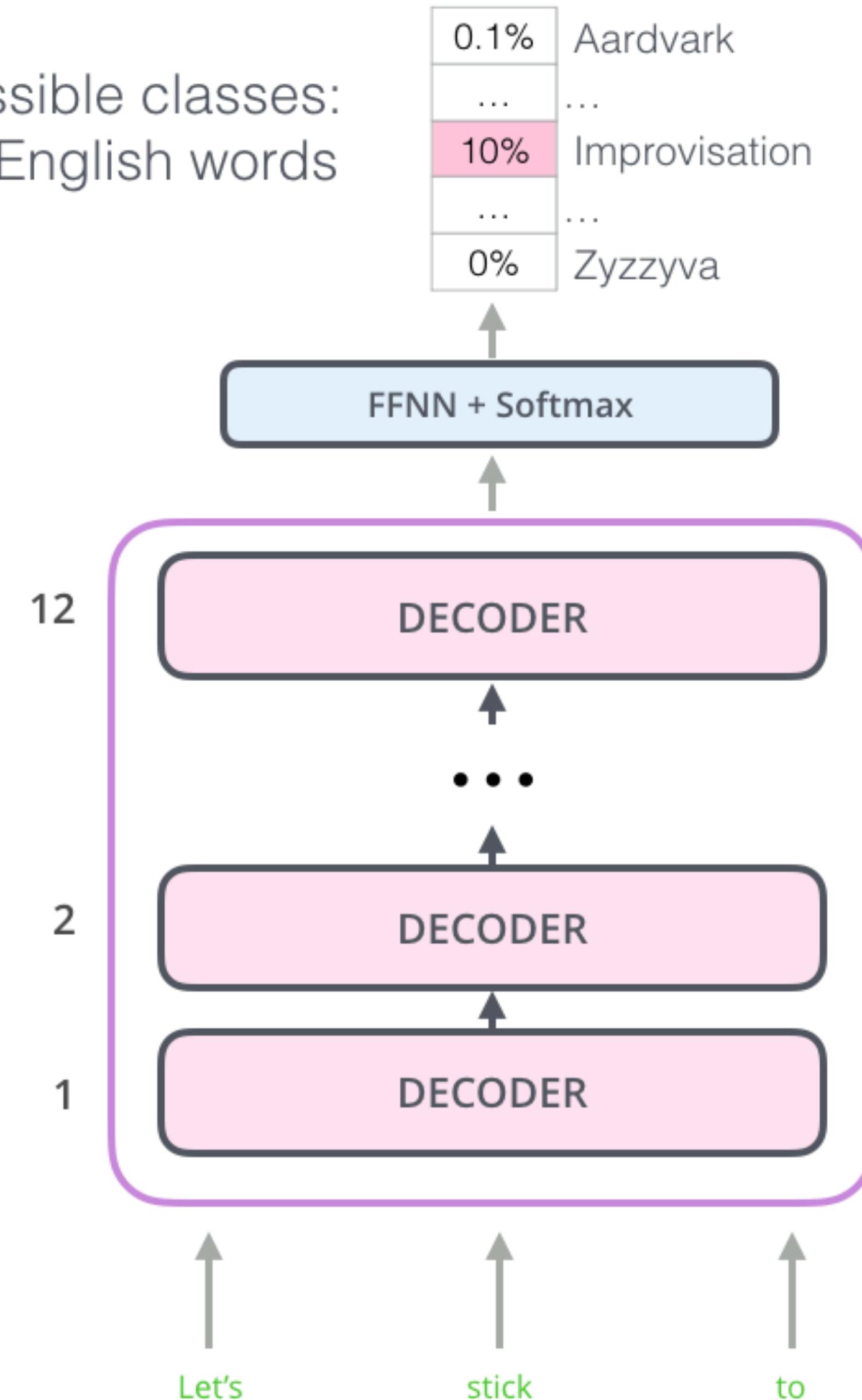
Архитектура: Transformer Decoder

Данные: тексты книг (BookCorpus)

- + Разнообразие
- + Большой объем

Possible classes:
All English words

0.1%	Aardvark
...	...
10%	Improvisation
...	...
0%	Zyzyva



GPT

Generative Pre-Training

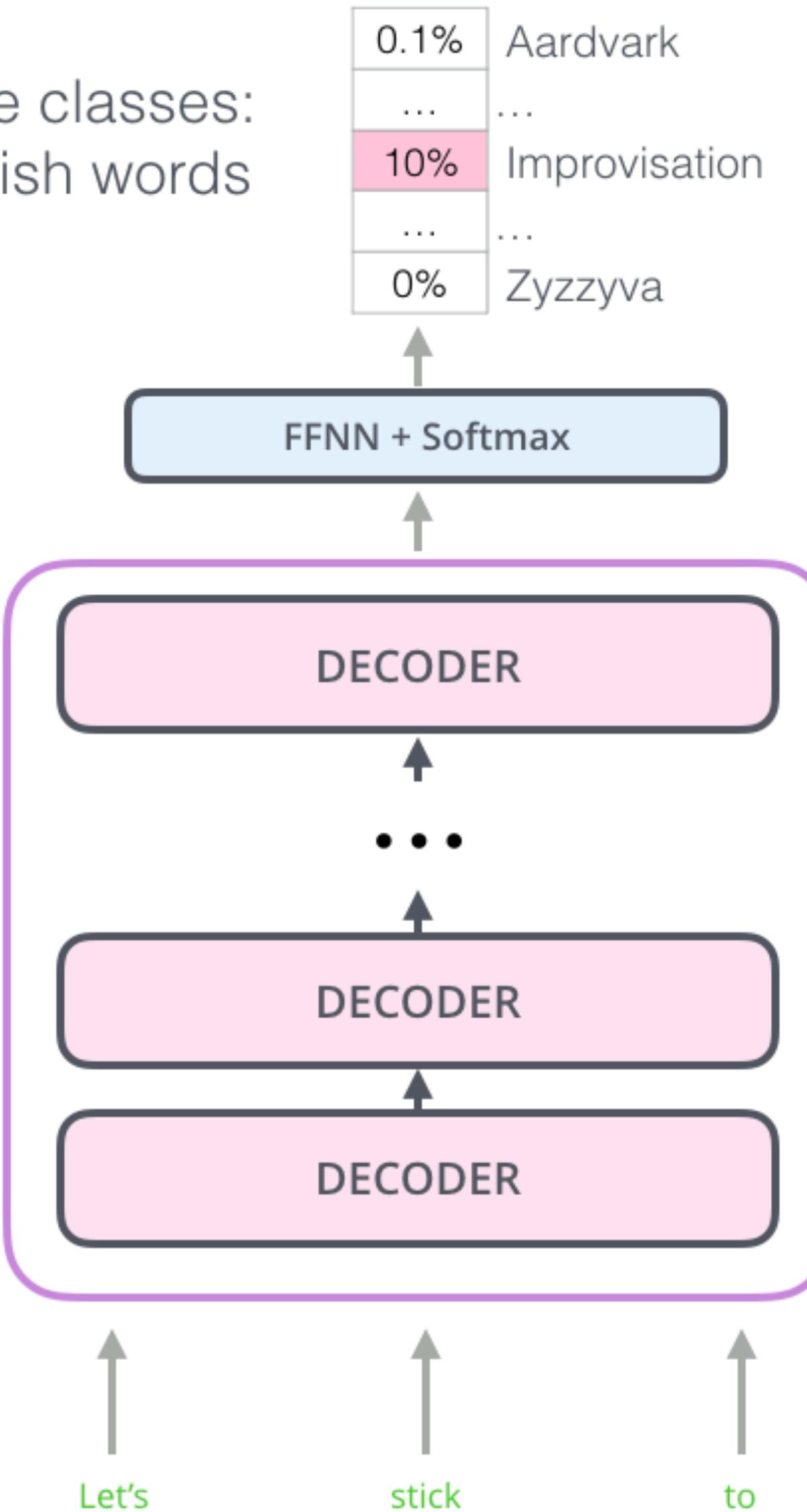
Архитектура: Transformer Decoder

Данные: тексты книг (BookCorpus)

Задача для обучения: Language Modeling

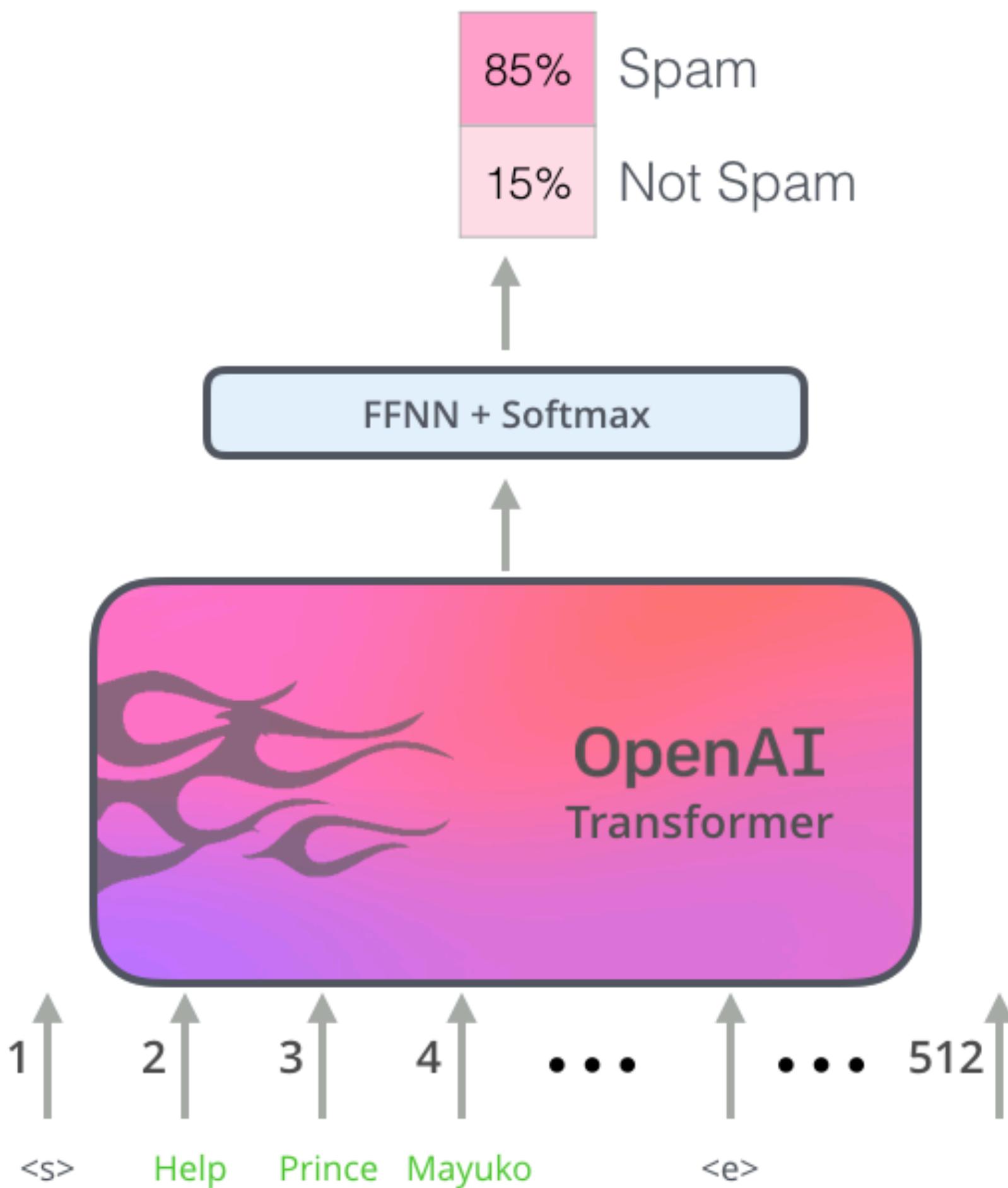
Possible classes:
All English words

0.1%	Aardvark
...	...
10%	Improvisation
...	...
0%	Zyzyva



GPT

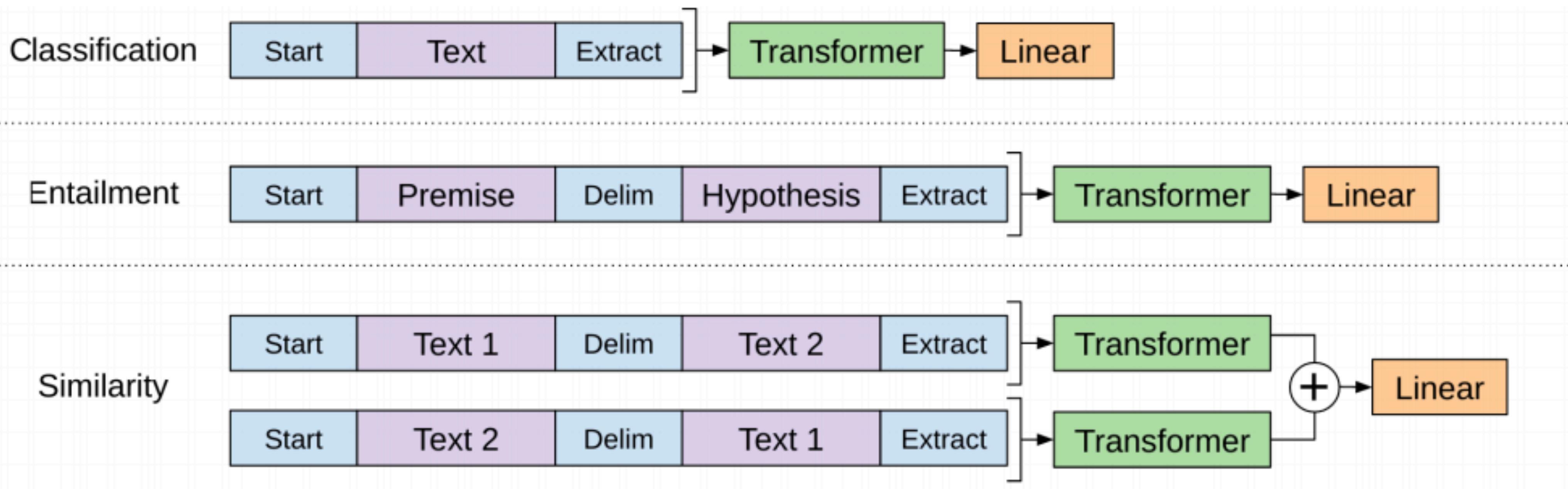
Как использовать?



GPT

Как использовать?

Для разных задач - разный формат входа



GPT-2

х10 параметров

х10 данных

GPT-2

Как использовать? **zero-shot task transfer**

Нет дообучения на новую задачу (fine-tuning)

Форматируем input, чтобы была понятна задача

Zero-shot task transfer — это подход в машинном обучении, при котором модель обучена выполнять задачу без явного обучения на примерах этой задачи

В рамках zero-shot подхода вы можете просто предоставить модели описание задачи, например: "Классифицируй следующий текст в одну из категорий: политика, спорт, наука, искусство." И, например, передать сам текст: "Сегодня прошел важный матч в Лиге чемпионов, где победу одержал клуб из Барселоны."

GPT-2

Форматируем input, чтобы была понятна задача

Пример: задача суммаризации

Article: Amina Ali Qassim is sitting with her youngest grandchild on her lap, wiping away tears with her headscarf. Only a few months old, this is the baby girl whose ears she desperately tried to cover the night the aerial bombardment started. She lay awake, she says, in a village mosque on the Yemeni island of Birim, counting explosions as the baby cried.

It could have been worse though. They could have still been in their house when the first missile landed.

"Our neighbor shouted to my husband 'you have to leave, they're coming.' And we just ran. As soon as we left the house, the first missile fell right by it and then a second on it. It burned everything to the ground," Qassim tells us
...

TL;DR:

Input

GPT-2

Форматируем input, чтобы была понятна задача

Пример: задача суммаризации

Article: Amina Ali Qassim is sitting with her youngest grandchild on her lap, wiping away tears with her headscarf. Only a few months old, this is the baby girl whose ears she desperately tried to cover the night the aerial bombardment started. She lay awake, she says, in a village mosque on the Yemeni island of Birim, counting explosions as the baby cried.

It could have been worse though. They could have still been in their house when the first missile landed.
"Our neighbor shouted to my husband 'you have to leave, they're coming.' And we just ran. As soon as we left the house, the first missile fell right by it and then a second on it. It burned everything to the ground," Qassim tells us
...

Input

TL;DR: Yemen is in the middle of a civil war. Saudi Arabia is leading the coalition bombing campaign. It's been bombing Yemen for more than two months now.

GPT-2 prediction

GPT-2

Language Modeling (генерация текста)

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.



GPT-3

x100 параметров (по сравнению с GPT-2)

x5 данных (по сравнению с GPT-2)

* нет в свободном доступе

GPT-3

х100 параметров (по сравнению с GPT-2)

х5 данных (по сравнению с GPT-2)

Как использовать? **zero-shot/one-shot/few-shot settings**

- показываем 0/1/несколько примеров

One-shot learning (обучение на одном примере) — это подход, при котором модель учится выполнять задачу, используя только один пример. Задача: Классификация текста. Пример 1: "Этот текст о футболе." Задача: "Определите, к какой категории относится следующий текст." Текст: "Сегодня прошел важный матч Лиги чемпионов." Ответ модели: "спорт".

В few-shot learning модель получает несколько примеров и на основе них обучается выполнять задачу. Этот подход использует методы обобщения и передачу знаний (transfer learning), чтобы эффективно работать с ограниченным количеством данных. Пример: Задача: Классификация текста. Примеры: "Текст о футболе." → категория: "спорт" "Текст о политике." → категория: "политика" Задача: "Классифицируйте следующий текст." Текст: "Сегодня прошел важный матч Лиги чемпионов." Ответ модели: "спорт"

GPT-3

x100 параметров (по сравнению с GPT-2)

x5 данных (по сравнению с GPT-2)

Как использовать? **zero-shot/one-shot/few-shot settings**

- показываем 0/1/несколько примеров

English sentence 1 = France sentence 1
English sentence 2 = France sentence 2
English sentence 3 =

GPT-3 prediction

Input

GPT-3

Prompt:

The following are ideas for startups, apps, marketplaces, platforms:

###

A startup that helps content creators, such as photographers and videographers, monetize their work.

###

An online forum that helps people find roommates

###

GPT-3:

A community hub for entrepreneurs to share ideas and ask questions of experienced entrepreneurs

###

A website that helps people discover new interests and hobbies.

###

A company that helps brands advertise their products to potential buyers on Instagram.

###

Image credit

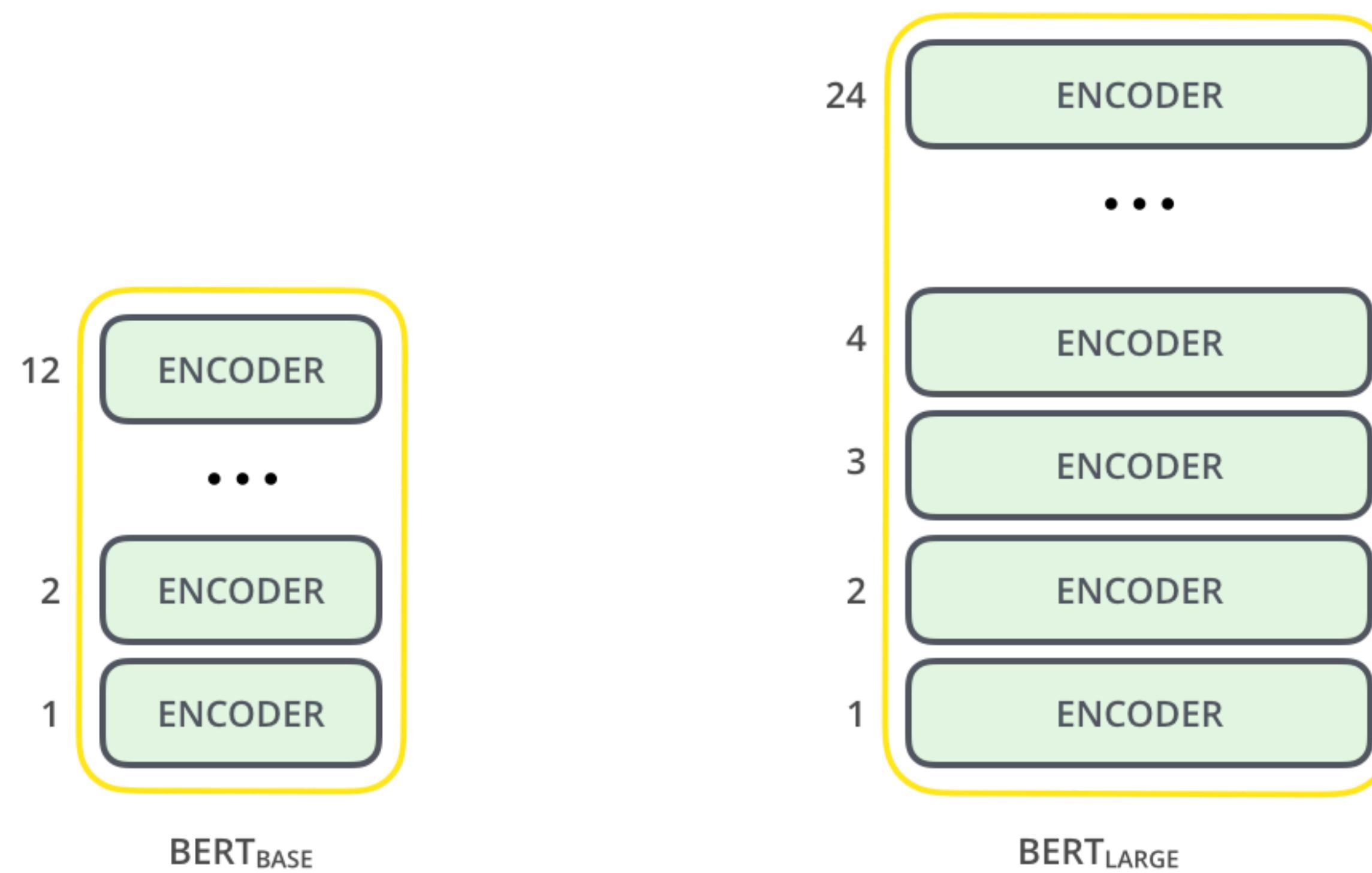
BERT



BERT

Bidirectional Encoder Representations from Transformers

Архитектура: Transformer Encoder



BERT

Bidirectional Encoder Representations from Transformers

Архитектура: Transformer Encoder

Данные: Wikipedia и тексты книг (BookCorpus)

BERT

Bidirectional Encoder Representations from Transformers

Архитектура: Transformer Encoder

Данные: Wikipedia и тексты книг (BookCorpus)

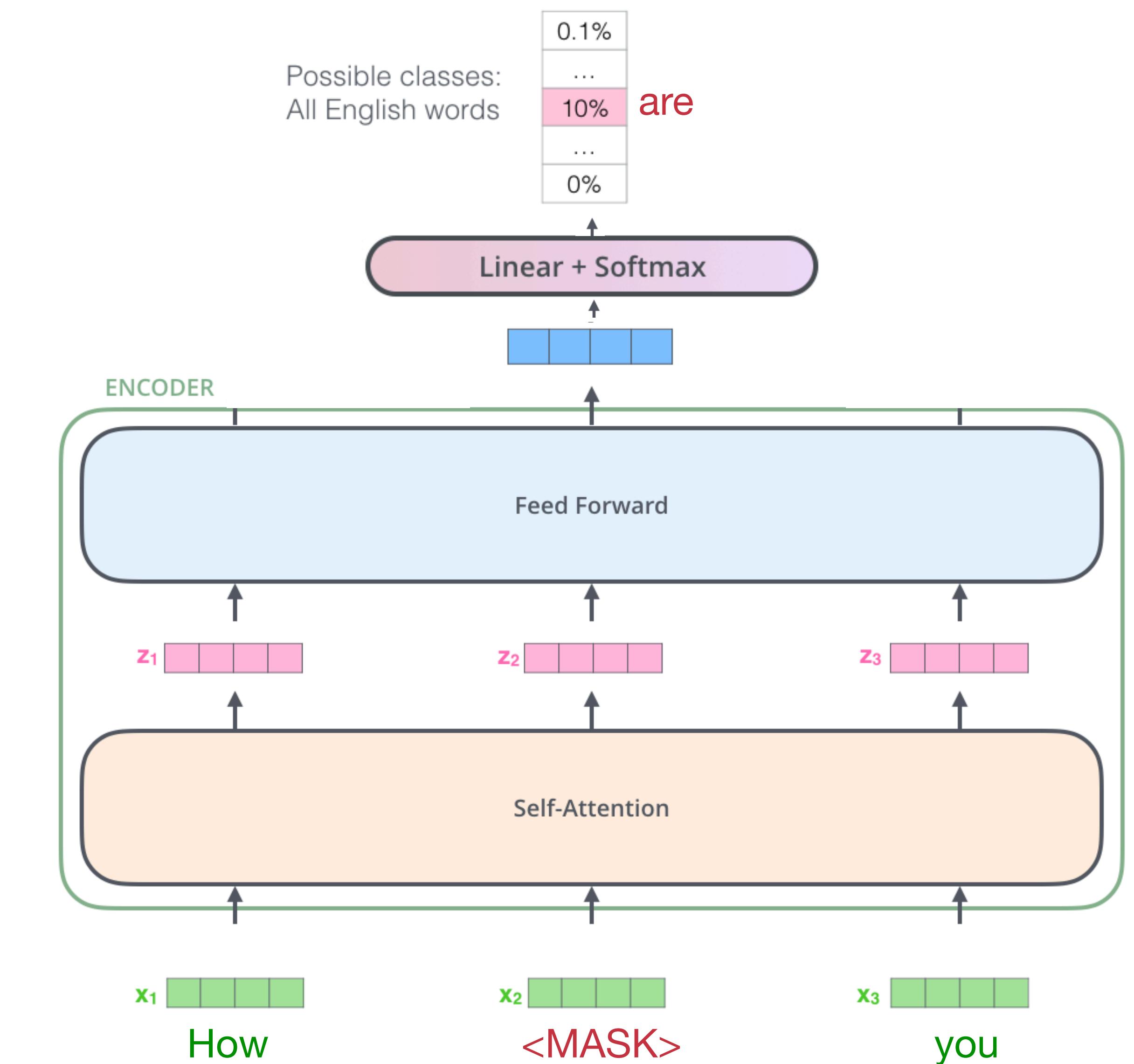
Задачи для обучения: Masked LM и Next Sentence Prediction

BERT

Masked Language Model

Случайно выбираем 15% позиций и заменяем на <MASK>

Задача: предсказать исходный токен



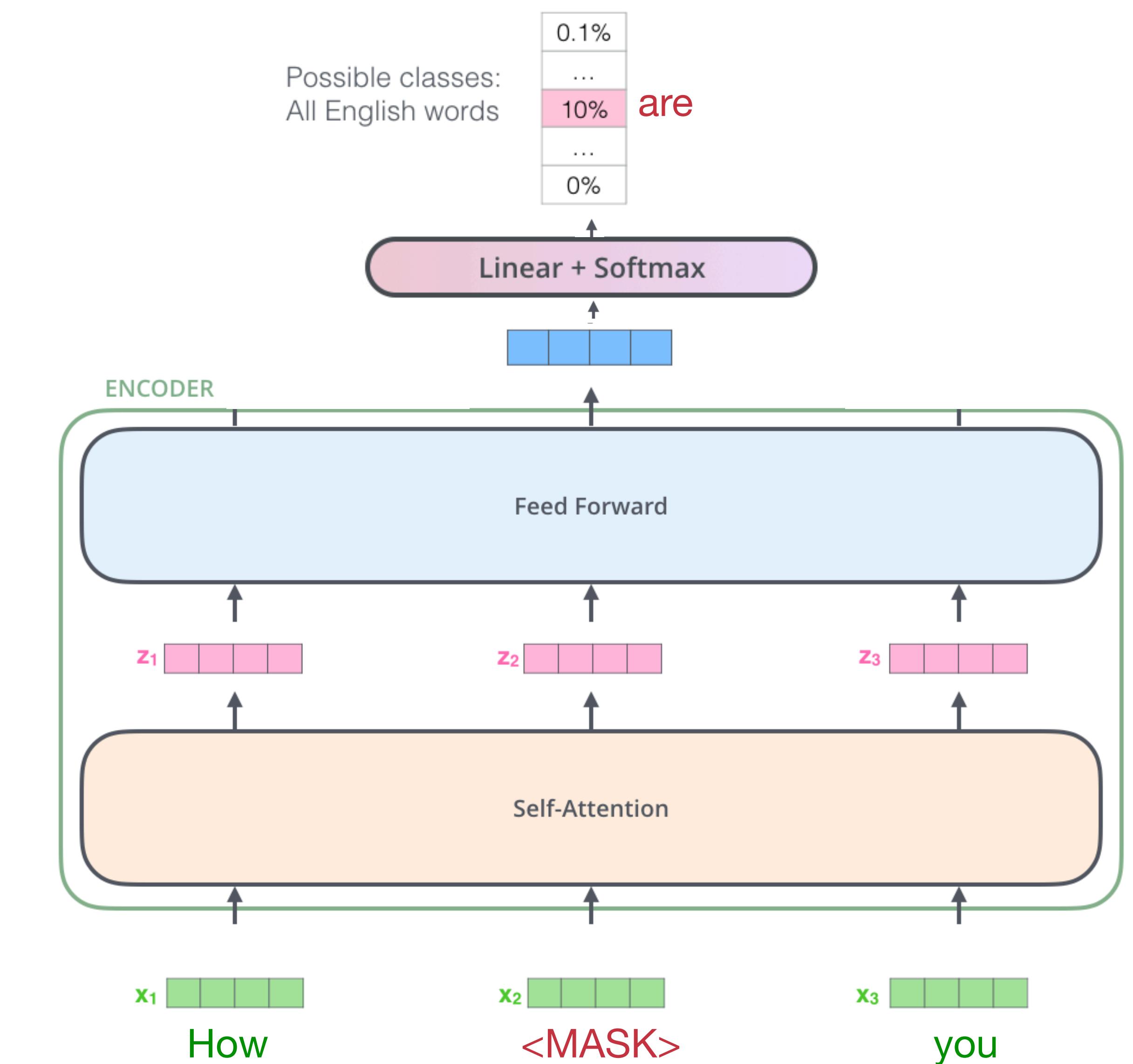
BERT

Masked Language Model

Случайно выбираем 15% позиций:

- 80% заменяем на <MASK>
- 10% заменяем на случайный токен
- 10% оставляем

Задача: предсказать исходный токен



BERT

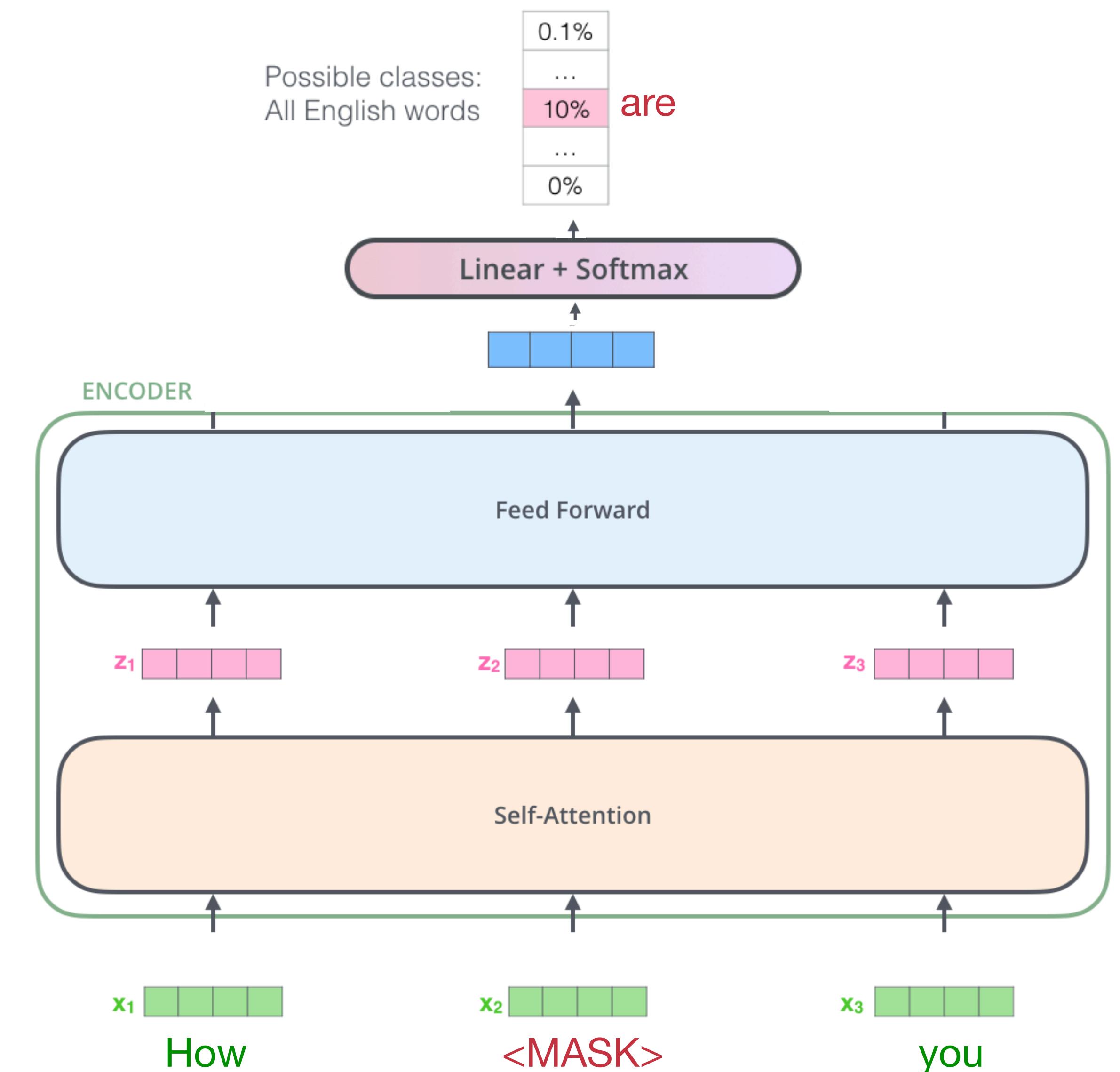
Masked Language Model

Случайно выбираем 15% позиций:

- 80% заменяем на <MASK>
- 10% заменяем на случайный токен
- 10% оставляем

Задача: предсказать исходный токен

Обученные эмбеддинги учитывают контекст слева и справа



BERT

Next Sentence Prediction

Для некоторых задач нужно понимать взаимоотношения между двумя предложениями:

- Similarity
- Entailment
- Question Answering
- ...

BERT

Next Sentence Prediction

Вход: 2 предложения (A и B)

50% - В следует за А в тексте

50% - В выбрано случайно

Формат входа:



BERT

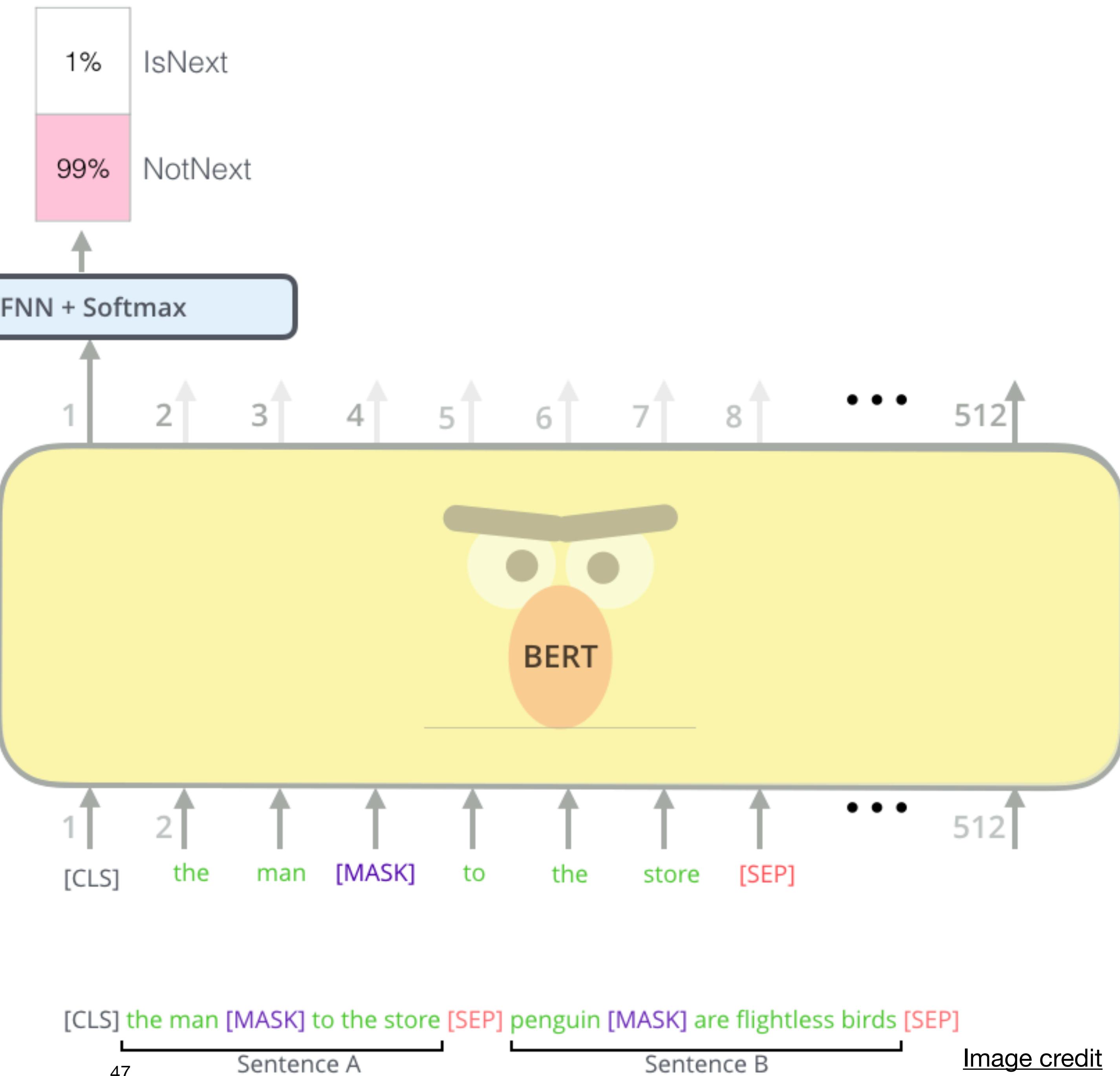
Next Sentence Prediction

Вход: 2 предложения (A и B)

50% - В следует за A в тексте

50% - В выбрано случайно

Задача: предсказать, следует ли
B за A



BERT

Next Sentence Prediction

Вход: 2 предложения (A и B)

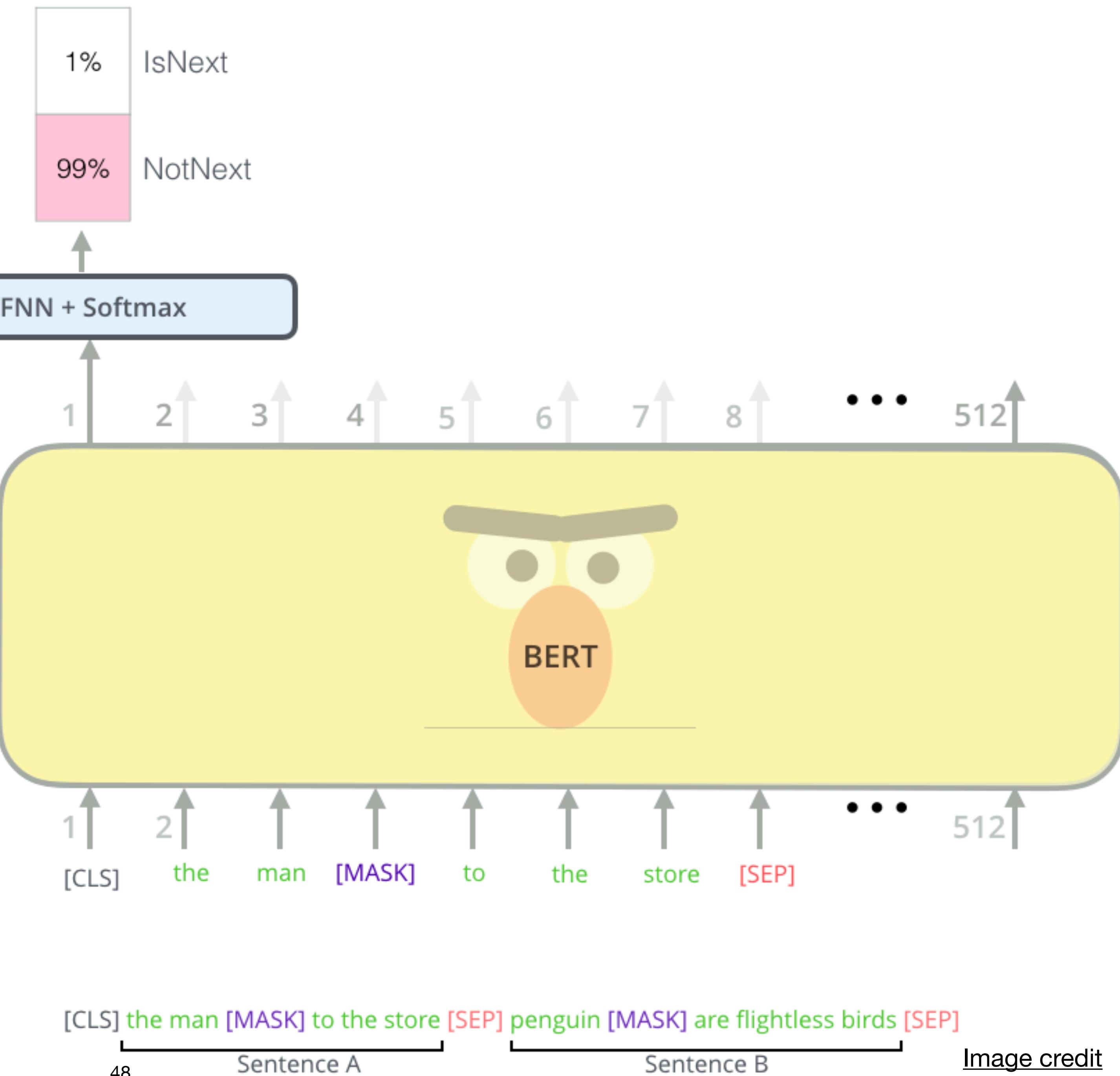
50% - В следует за A в тексте

50% - В выбрано случайно

Задача: предсказать, следует ли
B за A

<CLS> - выучивает
агрегированную информацию

<SEP> - разделитель



BERT

Как использовать?

- Linear+Softmax поверх <CLS> - для задач классификации предложения (или двух)
- Linear+Softmax поверх всех выходов - для задач классификации токенов
- Выходы BERT - как вход в другие модели (task-specific)
- ...

RoBERTa

Robustly Optimized BERT Pretraining Approach

Улучшенная версия BERT:

- x10 данных
- x10 вычислительных ресурсов (дольше обучение, больше batch size)
- Не использовали задачу Next Sentence Prediction

Улучшение в 2-20% (в зависимости от задачи)

BART and T5

BART

Bidirectional and Auto-Regressive Transformers

Идея: соединить преимущества BERT и GPT

Архитектура: Transformer Encoder-Decoder

Данные: как в RoBERTa

Гибридная архитектура: BART сочетает двухкомпонентную модель трансформера (энкодер и декодер), что позволяет использовать преимущества как двунаправленного внимания (подобного BERT), так и авторегрессивной генерации (подобной GPT).

В отличие от T5, который использует текст как вход и выход, BART обучается на задачах восстановления текста, где входной текст и его зашумленная версия используются для обучения:

Пример: Входной текст может быть: "Этот фильм был восхитительным", а зашумленная версия: "Этот фильм был __". Модель должна восстанавливать пропущенное слово.

BART

Bidirectional and Auto-Regressive Transformers

Идея: соединить преимущества BERT и GPT

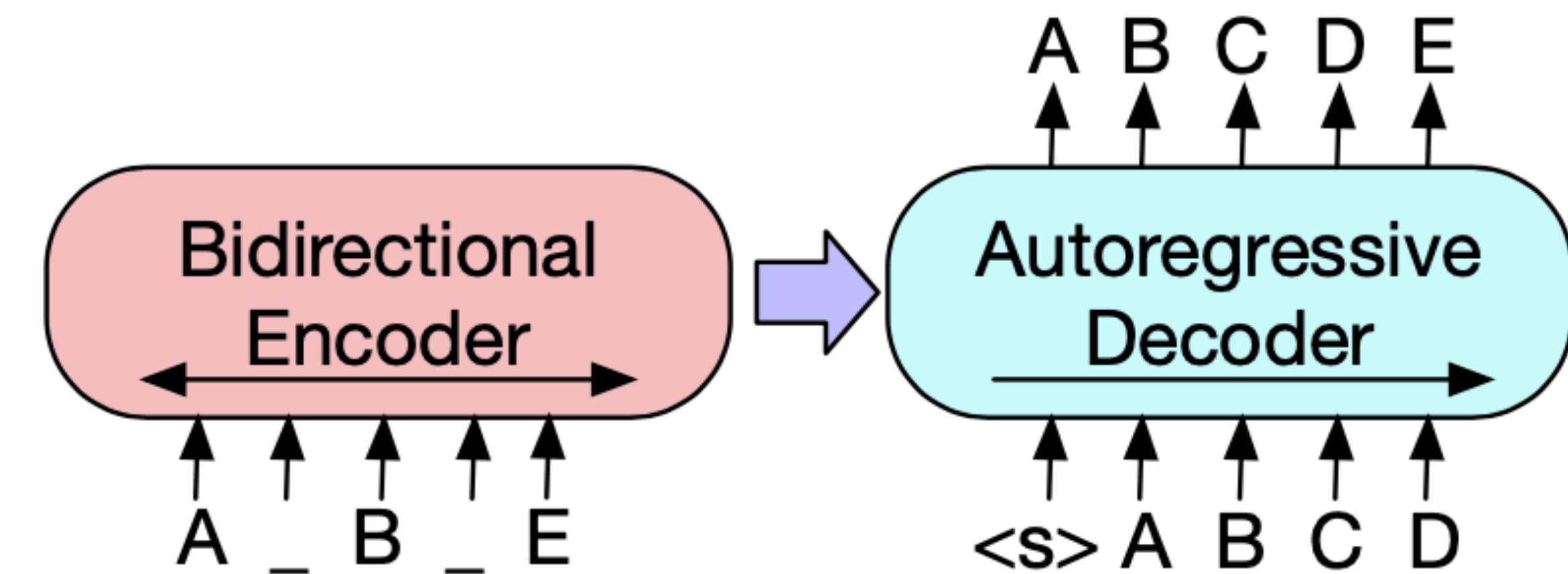
Архитектура: Transformer Encoder-Decoder

Данные: как в RoBERTa

Задача для обучения: восстановить последовательность

Последовательность: ABCDE

- Маскируем C, D
- Шум: лишняя маска перед B



T5

Text-to-Text Transfer Transformer

Идея: соединить преимущества BERT и GPT

Архитектура: Transformer Encoder-Decoder

Данные: x5 от данных RoBERTa

Задача для обучения: восстановить последовательность

T5 — это модель от Google Research, которая использует архитектуру трансформера, адаптированную для работы с текстом в формате "от текста к тексту". Все задачи, такие как перевод, суммирование, классификация, извлечение информации, преобразуются в задачу текстового преобразования. Таким образом, вход и выход всегда представлены в виде текста, что делает модель очень универсальной.

Основные особенности T5:

Текст как вход и выход: T5 преобразует все задачи NLP в задачу, где и вход, и выход — это текст. Например:

Перевод: Вход может быть "Переведи этот текст на французский: 'I love programming.'" и выход "J'adore programmer."

Классификация: Вход может быть "Классифицируй текст: 'Этот фильм замечательный.'", а выход — "Позитивный".

Суммирование: Вход — длинный текст, выход — краткое изложение.

Подход "Text-to-Text": В отличие от моделей, обученных специально для одной задачи (например, классификации или перевода), T5 решает задачи в универсальной текстовой форме⁵⁴, что делает его подход гибким и универсальным.

T5

Text-to-Text Transfer Transformer

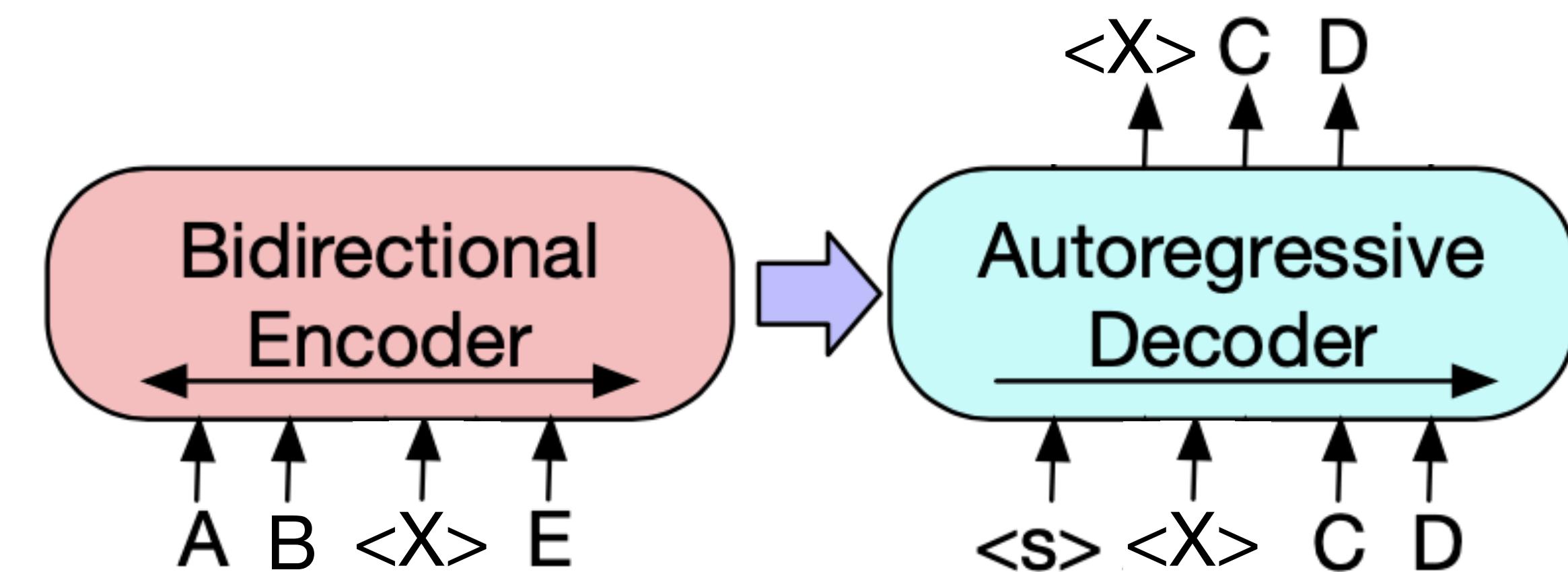
Идея: соединить преимущества BERT и GPT

Архитектура: Transformer Encoder-Decoder

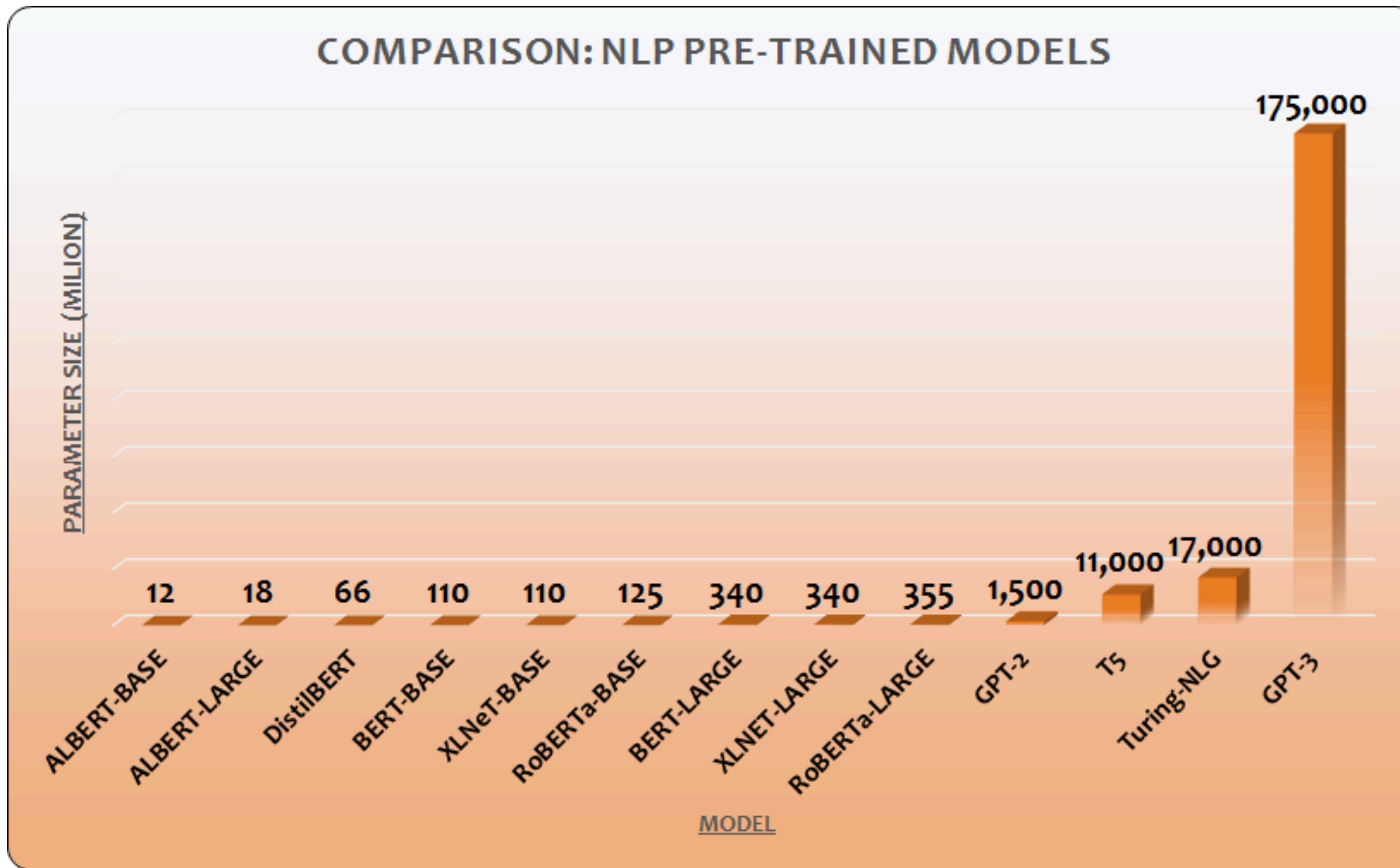
Данные: x5 от данных RoBERTa

Задача для обучения: восстановить последовательность

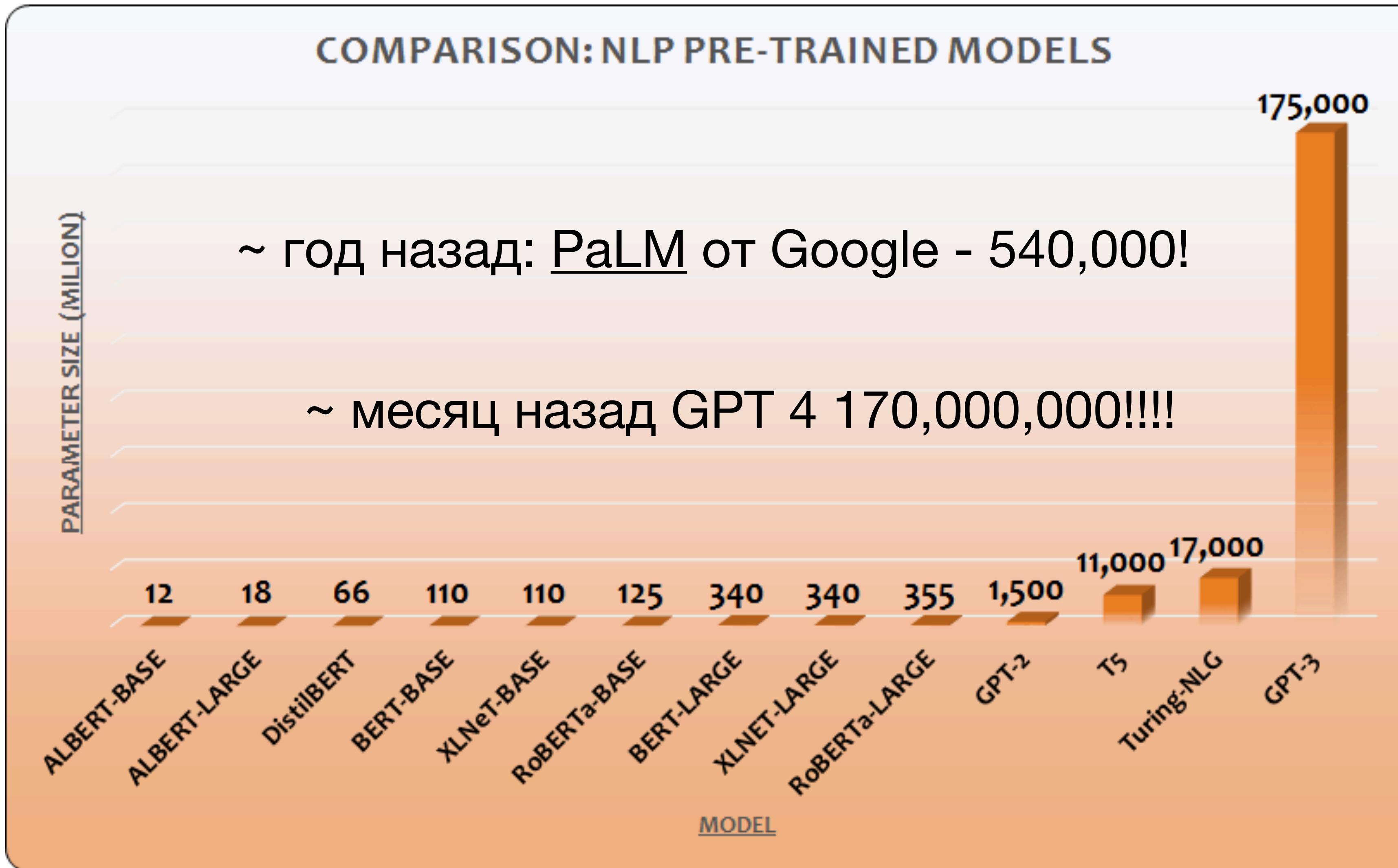
Вместо маски используем
специальные токены



Сравнение



Сравнение



BERT vs GPT vs ChatGPT

ChatGPT – разновидность GPT-3 моделей, дизайн которых создавался для chatbot приложения. Модель обучалась на большой датасете диалогов, так чтобы она могла генерировать ответы, подходящие для контекста мессенджеров, создавая связное обсуждение. При этом ChatGPT не такая мощная модель, как GPT-3, хотя и более быстрая

Bert и GPT-3 – различаются архитектурой, Bert – bidirectional model, GPT-3 – autoregressive. GPT-models учитывают только контекст слева от рассматриваемого слова, каждый токен предсказывается и сопоставляется только с предыдущими токенами. Данная особенность делает GPT модели подходящими для генерации текстов, но хуже справляющимися с классификацией. Bert – рассматривает контекст в обе стороны. Чаще используется для задач классификации, генерирует весь output за раз. Основная задача Bert во время обучения – маскировать случайным образом слова и пытаться их предсказать по контексту справа и слева

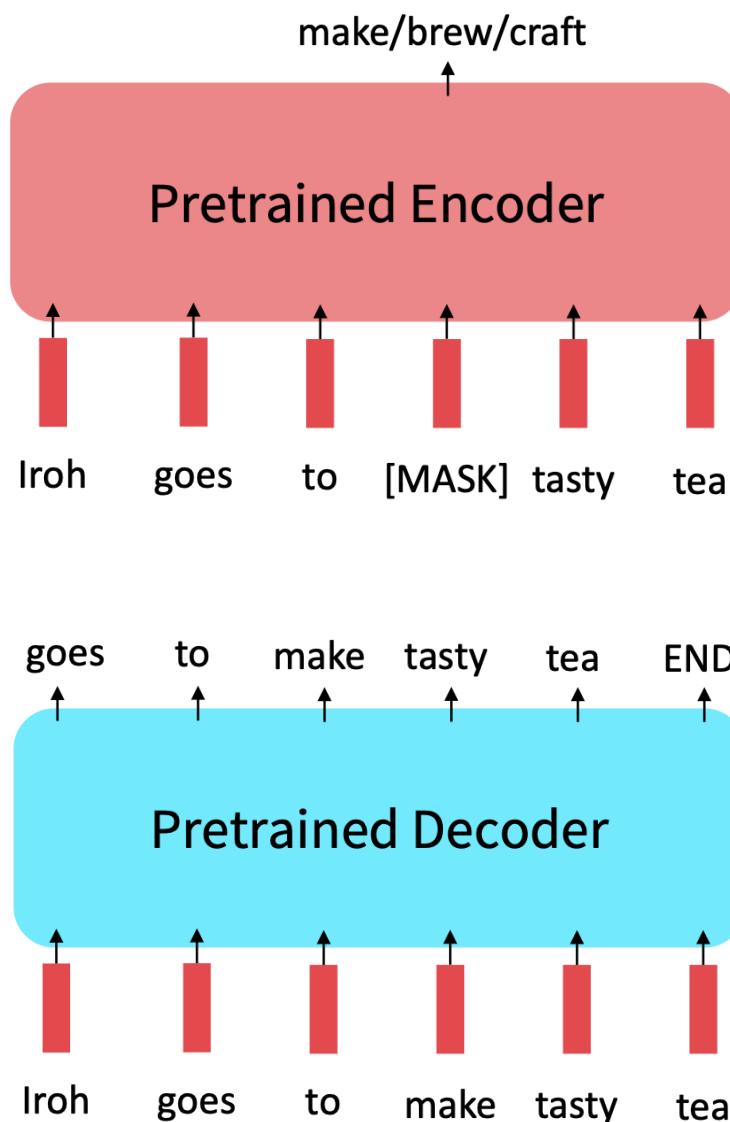
Bert

- **Encoder only**
- **Uses Self Attention**
- **Multiple models for different NLP tasks**
- **Fine Tuning on Custom Data**

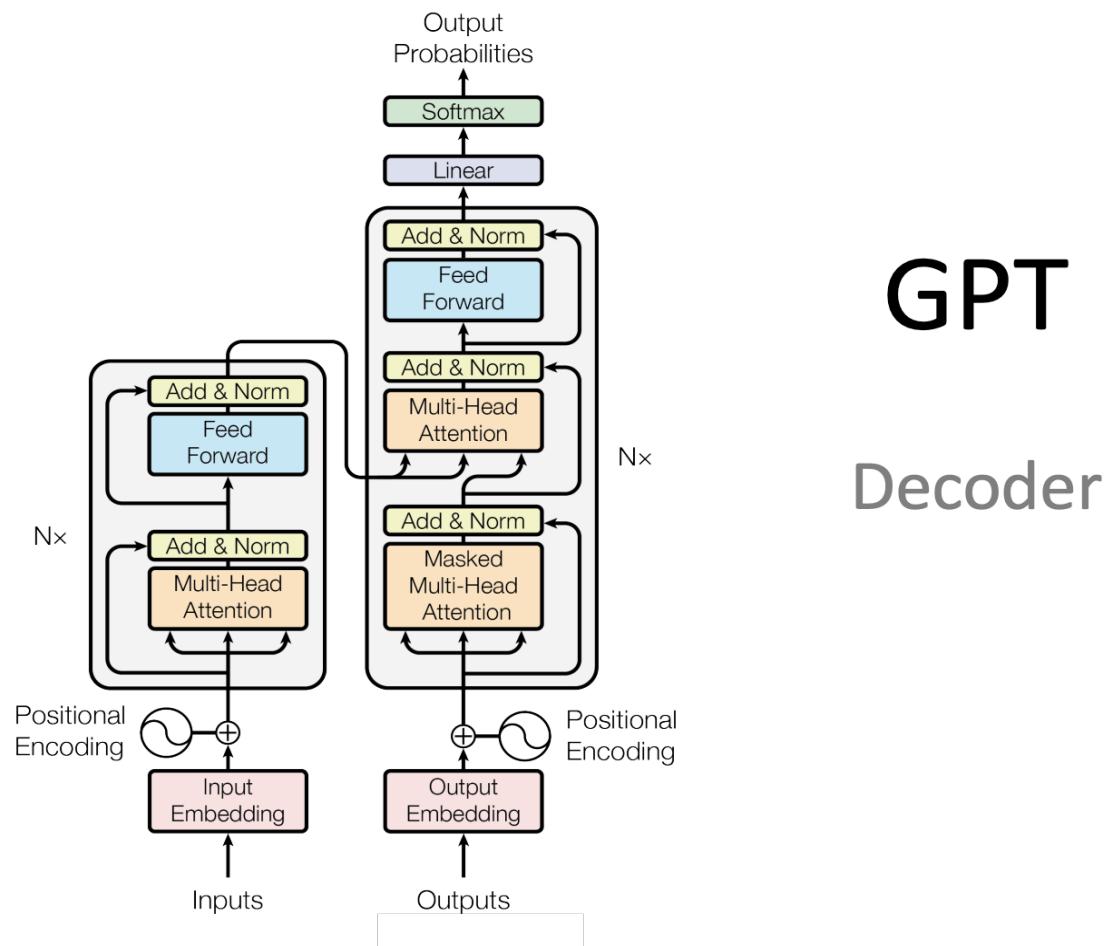
GPT

- **Decoder blocks only**
- **Uses Masked Self-Attention**
- **Single model for all NLP tasks**
- **Doesn't require fine tuning**

BERT vs GPT



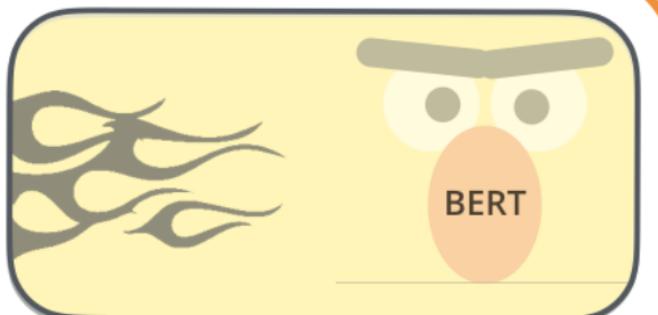
BERT
Encoder



1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

Semi-supervised Learning Step



Model:



Dataset:

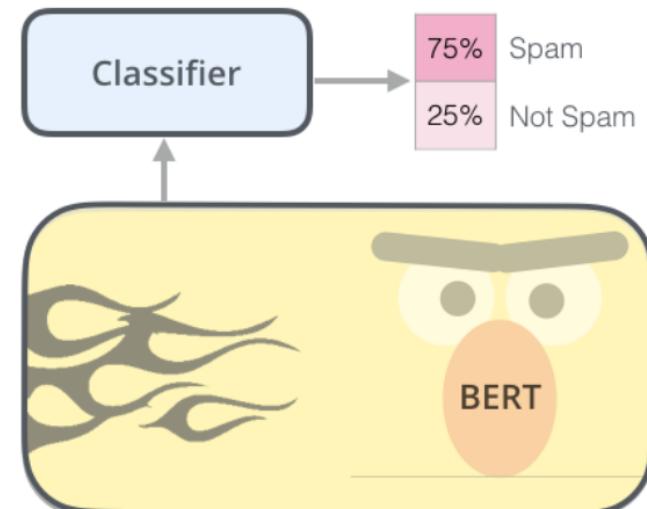


Objective:

Predict the masked word
(language modeling)

2 - Supervised training on a specific task with a labeled dataset.

Supervised Learning Step



Model:
(pre-trained
in step #1)

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached...	Not Spam

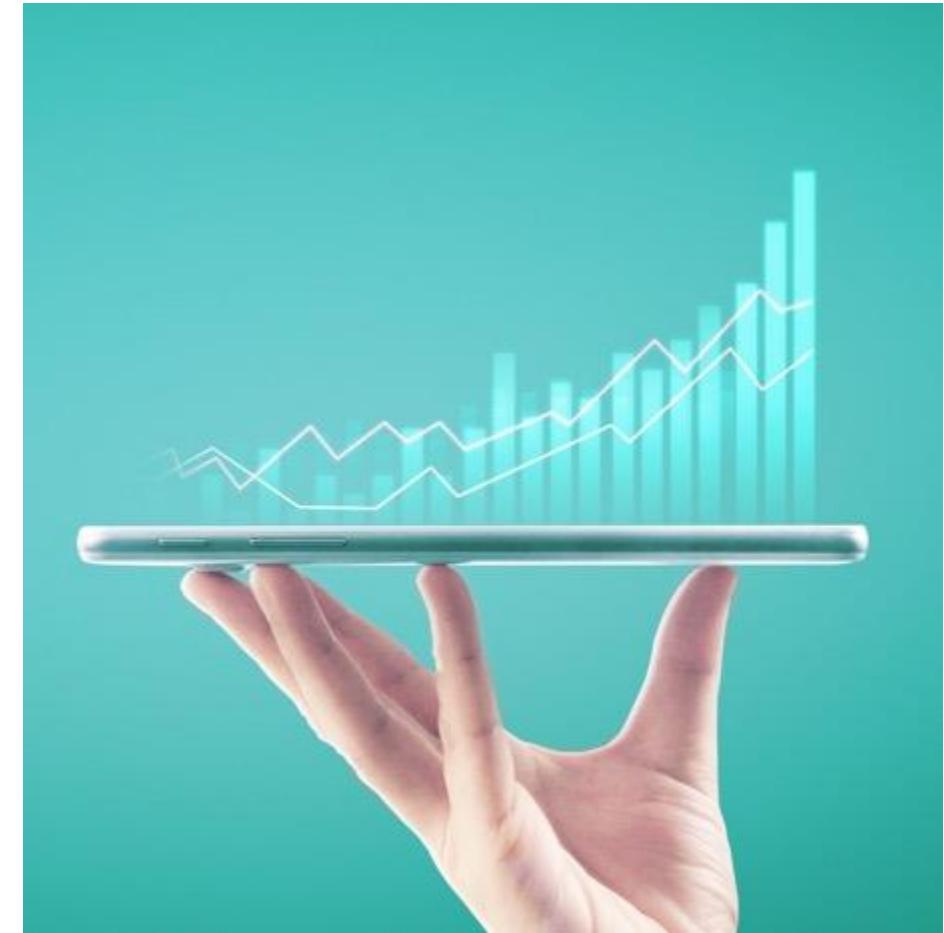
The two steps of how BERT is developed. You can download the model pre-trained in step 1 (trained on un-annotated data), and only worry about fine-tuning it for step 2. [Source for book icon].

Основные задачи, решаемые трансформерами

- General question answering, language understanding and classification (Bert)
- Text generation and understanding (Bart)
- Text generation, code generation, as well as image and audio generation (GPT-3)
- Object classification (CLIP)
- Dialog agent (ChatGPT)
- Text to image(GLIDE, DALL-E-2)
- Mathematical reasoning (Minerva)
- Summarization (Pegasus)
- Translation and other cross-lingual language tasks (XLM-RoBERTa)

Основные задачи, решаемые трансформерами 2

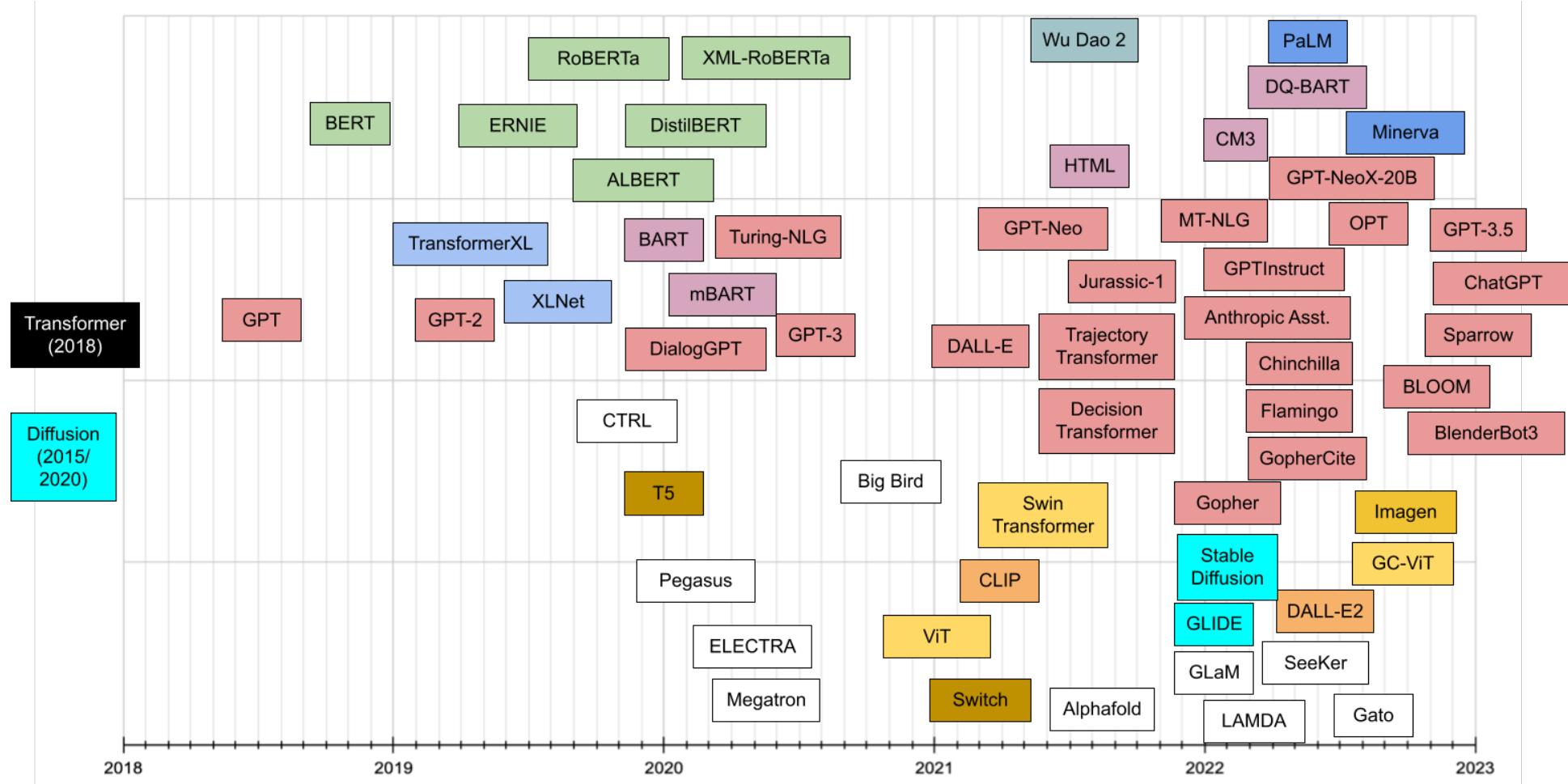
- Трансформеры применяются и для анализа табличных данных, например, в статье «Revisiting Deep Learning Models for Tabular Data»*, подготовленной командой Яндекса сравниваются различные модели (MLP, ResNet, FT-Transformer и.т.д.) и предпочтение отдается трансформерам, однако результат зависит от специфики задачи, например, для Long Time Series forecasting существуют модели проще по структуре и превосходящие трансформеры: « Are Transformers Effective for Time Series Forecasting? »**. Кроме того, кол-во данных существенно влияет на выбор модели, так как увеличивается риск переобучения



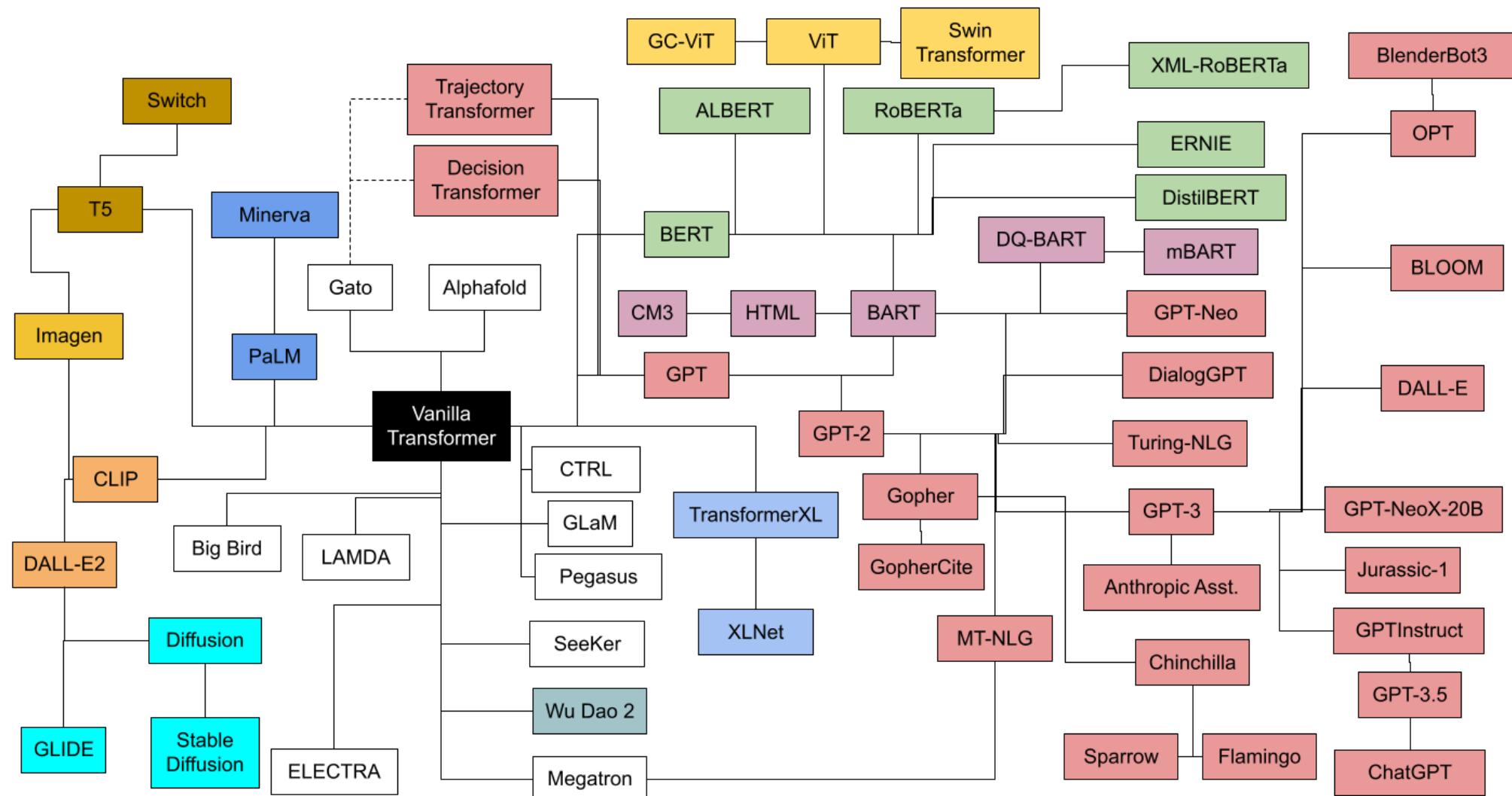
*https://openreview.net/pdf?id=i_Q1yrOegLY

**<https://arxiv.org/pdf/2205.13504.pdf>

Приложение: Хронология развития трансформеров



Приложение: Дерево связей



Как использовать?

Большой список доступных моделей и удобный интерфейс - библиотека
HuggingFace Transformers 

Приложение: Маскирование в декодерах

Маскирование в декодерах отличается от обучения с [MASK] токенами в энкодерах

В декодерах перекрываетяется всё, что расположено левее рассматриваемого токена, а значение скалярного произведения закладывается как -inf

- To enable parallelization, we **mask out attention** to future words by setting attention scores to $-\infty$.

$$e_{ij} = \begin{cases} q_i^\top k_j, & j \leq i \\ -\infty, & j > i \end{cases}$$

For encoding
these words

We can look at these
(not greyed out) words

