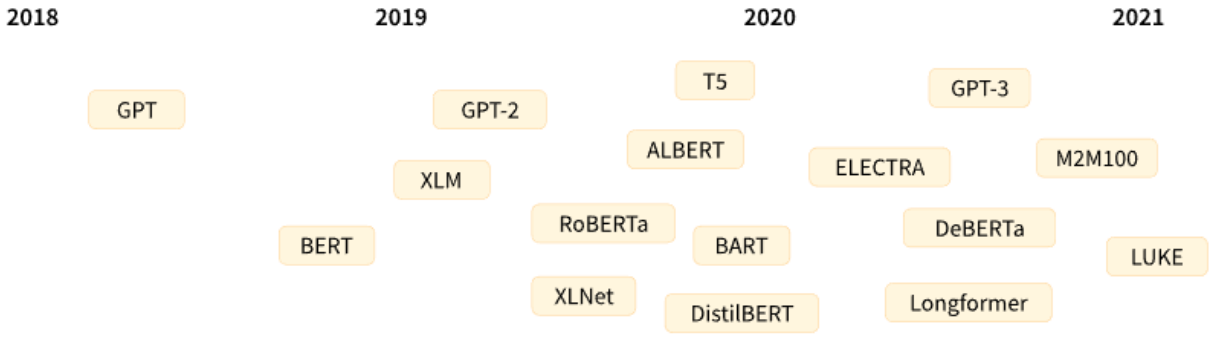
https://huggingface.co/learn/nlp-course/ru/chapter1/4?fw=pt

История трансформеров



Архитектура трансформеров была опубликована в июне 2017. Основной фокус оригинального исследования был сосредоточен на задачах перевода. Эта публикация повлекла за собой несколько влиятельных моделей:

- **Июнь 2018**: **GPT**, первая предобученная модель для тонкой настройки или дообучения (fine-tuning), которая показала результаты высокого качества для многих NLP-задач.
- **Октябрь 2018**: **BERT**, другая большая предобученная модель, была разработана для извлечения более точного содержания из предложений
- **Февраль 2019**: **GPT-2**, улучшенная (и более объемная) версия GPT, которая не была сразу опубликована по этическим соображениям
- **Октябрь 2019**: **DistilBERT**, «дистиллированная» версия BERT, которая на 60% быстрее и на 40% менее объемная, однако сохраняющая 97% производительности BERT
- **Октябрь 2019**: **BART and T5**, две больших модели, повторяющие архитектуру классического трансформера
- **Май 2020**, **GPT-3**, еще более крупная версия GPT-2, способная хорошо справляться с различными задачами без необходимости fine-tuning (так называемое *zero-shot learning*)

Этот список далеко неполный, он всего лишь призван выделить несколько разновидностей моделей трансформеров. В широком смысле трансформеры могут быть классифицированы на три типа:

1. **GPT**-подобные модели (также часто называемые авторегрессионные трансформеры)

2. **BERT**-подобные модели (также часто называемые автокодирующие трансформеры (auto-encoding))

3. **BART/T5** модели (также часто называются модели класса последовательность-последовательность (sequence-to-sequence, seq2seq))

Трансформеры - языковые модели

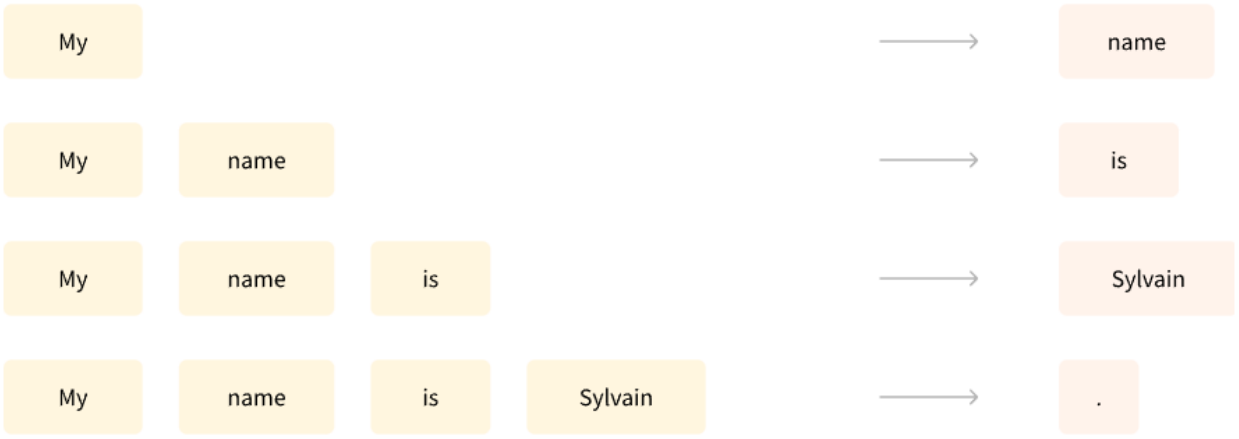
Все модели трансформеров, упомянутые выше (GPT, BERT, BART, T5, etc.) обучены как *языковые модели (англ. language models)*.

Это означает, что они обучены на огромном количестве текста, используя технику самостоятельного обучения (англ. self-supervised learning).

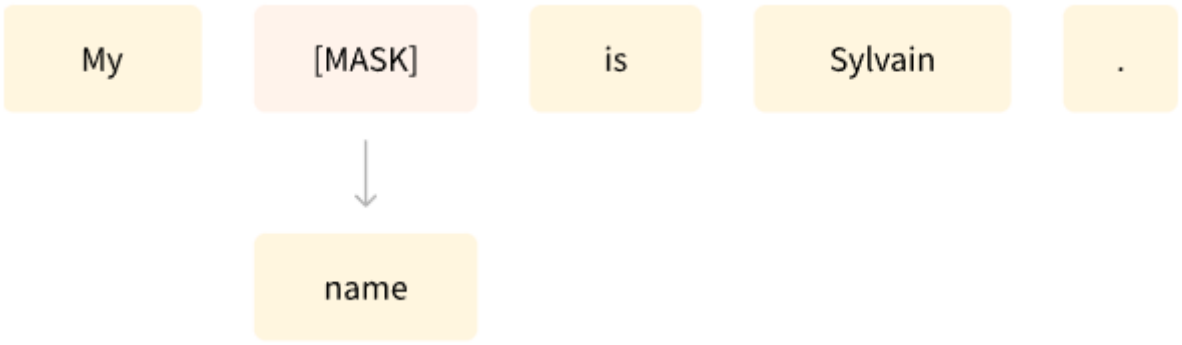
Самостоятельное обучение - это такой способ обучения, в котором цель обучения автоматически вычисляется на основе входных данных. Это означает, что люди не должны размечать данные!

Такой тип моделей реализует статистическое понимание языка, на котором он был обучен, но он не очень полезен для конкретных практических задач. Из-за этого базовая предварительно обученная модель потом подвергается процедуре, называемой *трансферным обучением (англ. transfer learning)*. В ходе этого процесса модель настраивается под конкретные наблюдения, т.е. размеченными человеком данными для конкретной задачи.

В качестве примера можно привести предсказание следующего слова в предложении на основе *n* предыдущих слов. Это называется *каузальным языковым моделированием (англ. causal language modeling)*, потому что модель зависит от прошлых и текущих слов, но не от будущих.

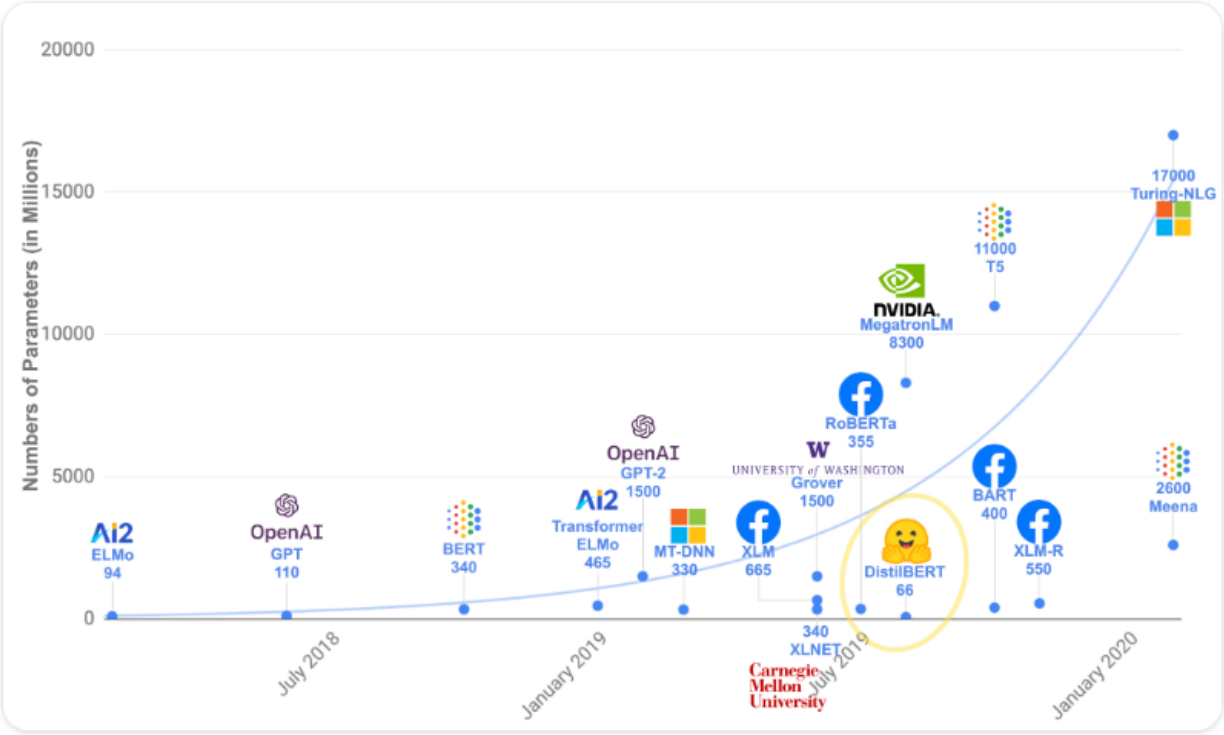


Другой пример - *маскированная языковая модель (англ. masked language modeling)*, которая предсказывает замаскированное слово в предложении.

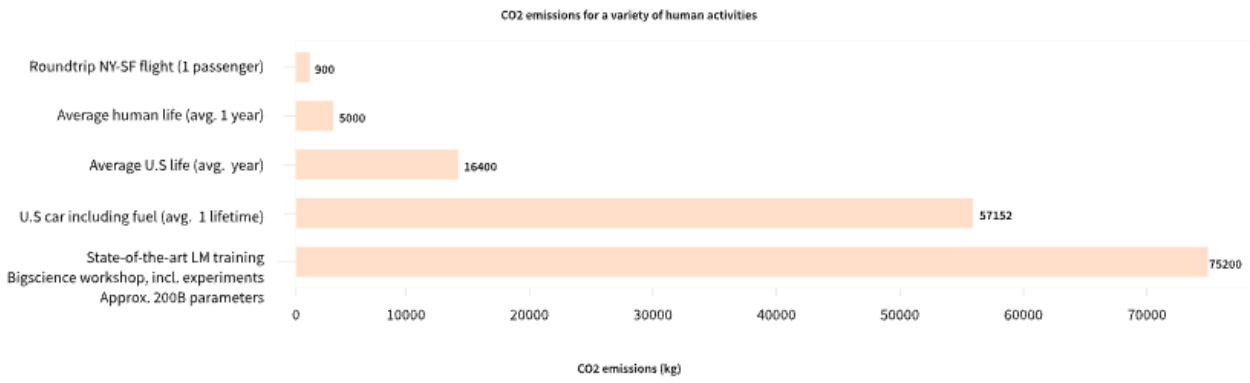


Трансформеры - большие модели

За исключением нескольких моделей (например, **DistilBERT**), общий подход к достижению высокого качества заключается **в увеличении размера моделей и увеличении количества данных для обучения**.



К сожалению, обучение модели, особенно большой, требует большого количества данных. Это приводит к увеличению времени и вычислительных потребностей. Это воздействует даже на окружающую среду, что можно увидеть графике ниже.



Вот почему распространение языковых моделей имеет первостепенное значение: распространение обученных весов и построение новых моделей на их основе снижает общую стоимость вычислений и углеродный след сообщества.

Кстати, вы можете измерить углеродный след, который оставят ваши модели, при помощи нескольких инструментов. Например, интегрированные в Transformers ML CO2 Impact или Code Carbon.

Чтобы узнать больше о этом, вы можете прочитать этот блог-пост (https://huggingface.co/blog/carbon-emissions-on-the-hub), в котором мы рассказываем как сгенерировать файл emissions.csv, содержащий прогноз объемов выброса углерода во время обучения модели, а также документацию (https://huggingface.co/docs/hub/model-cards-co2) Transformers, в которой затрагивается эта тема.

Трансферное обучение

Предобучение *(англ. pretraining)* - это процесс обучения модели с нуля: веса модели случайным образом инициализируются, после начинается обучение без предварительных настроек.

Предобучение обычно происходит на огромных наборах данных, сам процесс может занять несколько недель.

Дообучение (англ. fine-tuning), с другой стороны, это обучение после того, как модель была предобучена. Для дообучения вы сначала должны выбрать предобученную языковую модель, а после продолжить ее обучение ее на данных собственной задачи. Стойте — почему не обучить модель сразу же на данных конкретной задачи? Этому есть несколько причин:

Предобученная модель уже обучена на датасете, который имеет много сходств с датасетом для дообучения. Процесс дообучения может использовать знания, которые были получены моделью в процессе предобучения (например, в задачах NLP предварительно обученная модель будет иметь представление о статистических закономерностях языка, который вы используете в своей задаче).

Так как предобученная модель уже “видела” много данных, процесс дообучения требует меньшего количества данных для получения приемлемых результатов.

По этой же причине требуется и намного меньше времени для получения хороших результатов.

Например, можно использовать предварительно обученную на английском языке модель, а затем дообучить ее на корпусе arXiv, в результате чего получится научно-исследовательская модель. Для дообучения потребуется лишь ограниченный объем данных: знания, которые приобрела предварительно обученная модель, «передаются» (осуществляют трансфер), отсюда и термин «трансферное обучение».

Таким образом, дообучение модели требует меньше времени, данных, финансовых и экологических затрат. Также быстрее и проще перебирать различные схемы дообучения, поскольку оно требует меньше усилий, чем полное предварительное обучение.

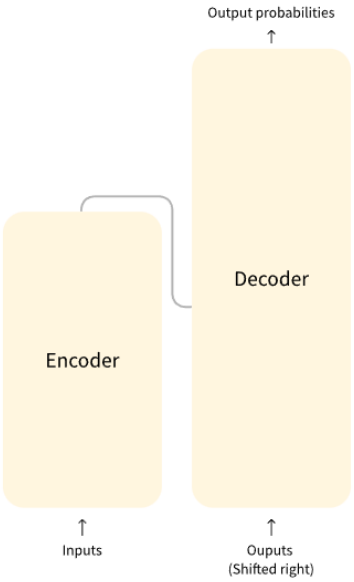
Этот процесс также даст лучшие результаты, чем обучение с нуля (если только у вас нет большого количества данных), поэтому вы всегда должны пытаться использовать предобученную модель — модель, максимально приближенную к поставленной задаче, а потом дообучить ее.

Общая архитектура

Модель состоит из двух блоков:

- Кодировщик (слева) (англ. encoder): кодировщик получает входные данные и строит их репрезентацию (формирует признаки). Это означает, модель нацелена на “понимание” входных данных.

- Декодировщик (справа) (англ. decoder): декодировщик использует репрезентации (признаки) кодировщика с другими входными данными для создания нужной последовательности. Это означает, что модель нацелена на генерацию выходных данных.



Каждая из этих частей может быть использована отдельно, это зависит от задачи:

- Модели-кодировщики: полезны для задач, требующих понимания входных данных, таких как классификация предложений и распознавание именованных сущностей.

- Модели-декодировщики: полезны для генеративных задач, таких как генерация текста.
- Модели типа “кодировщик-декодировщик” или seq2seq-модели: полезны в генеративных задачах, требующих входных данных. Например: перевод или автоматическое реферирование текста.

Слой внимания

Ключевой особенностью трансформеров является наличие в архитектуре специального слоя, называемого слоем внимания (англ. attention layer).

Статья, в которой была впервые представлена архитектура трансформера, называется “Attention Is All You Need" (“Внимание - все, что вам нужно")! https://arxiv.org/abs/1706.03762

Мы изучим детали этого слоя позже. На текущий момент мы сформулируем механизм его работы так: слой внимания помогает модели “обращать внимание" на одни слова в поданном на вход предложении, а другие слова в той или иной степени игнорировать. И это происходит в процессе анализа каждого слова.

Чтобы поместить это в контекст, рассмотрим задачу перевода текста с английского на французский язык. Для предложения “You like this course”, модель должна будет также учитывать соседнее слово “You”, чтобы получить правильный перевод слова “like”, потому что во французском языке глагол “like” спрягается по-разному в зависимости от подлежащего. Однако остальная часть предложения бесполезна для перевода этого слова. В том же духе при переводе “like” также необходимо будет обратить внимание на слово “course”, потому что “this” переводится по-разному в зависимости от того, стоит ли ассоциированное существительное в мужском или женском роде. Опять же, другие слова в предложении не будут иметь значения для перевода “this”. С более сложными предложениями (и более сложными грамматическими правилами) модели потребуется уделять особое внимание словам, которые могут оказаться дальше в предложении, чтобы правильно перевести каждое слово.

Такая же концепция применима к любой задаче, связанной с обработкой естественного языка: **слово само по себе имеет некоторое значение, однако значение очень часто зависит от контекста**, которым может являться слово (или слова), стоящие вокруг искомого слова.

Первоначальная архитектура

https://huggingface.co/learn/nlp-course/ru/chapter1/4?fw=pt

Архитектуры и чекпоинты

По мере погружения в трансформеры, вы будете встречать термины *архитектуры* и *чекпоинты* (англ. checkpoints) в смысле *модели*. Эти термины имеют разный смысл:

Архитектура - скелет модели — слои, связи и операции, которые выполняются в модели.

Чекпоинт - веса модели, которые могут быть загружены для конкретной архитектуры.

Модель - зонтичный термин, который может означать и архитектуру, и веса для конкретной архитектуры.

Например, **BERT** - это архитектура, а **bert-base-cased** - набор весов, подготовленный Google к первому выпуску BERT'а, - это чекпоинт. Однако можно сказать и “модель BERT”, и “модель bert-base-cased”.