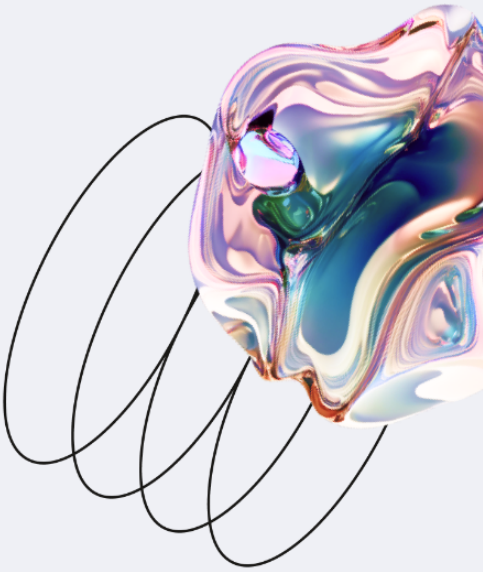
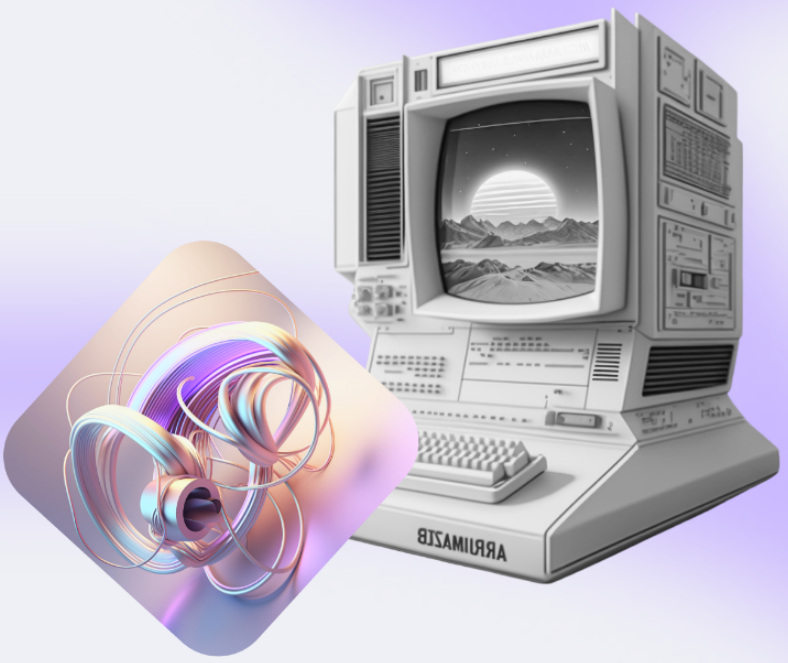
GeekBrains



# Model as a service и разные применения Transfer learning

## Transfer learning



# Введение

Добрый день, дорогие слушатели!

К этому занятию вы уже погрузились в предысторию и предназначение нейросетей, которые участвуют в переносе обучения, он же transfer learning.

В третьей лекции мы коснемся вопросов, которые позволяют внедрять такие нейросети в реальное производство. Это будет лекция о важных идеях применения переноса обучения.

Не будем отрицать, для продакшена очень важны и технические вопросы. Трудности в подготовке данных и в размещении нейросети в памяти видеокарты. Но на этой лекции эти вопросы будут для нас побочными. В основном мы поговорим разных применениях, придуманных для преобученных нейросетей.

В основном, я рассчитываю что вы научитесь соизмерять свои ожидания от сервисов с нейросетями, с их реальными возможностями. Вы сможете поверхностно анализировать их работу и понимать, из чего они состоят. Будете знать, как именно инженеры собирают из нейросетей нечто новое, не обучая сети заново. А еще, поскольку тема весьма модная, этот материал должен пригодится на собеседованиях.

Если вы еще не проходили сам предмет нейросетей, то у вас накопится много интересных вопросов по ходу этого курса. Это будет отлично, давайте их обсудим. Как минимум, я бы хотел чтобы в комментариях вы бы спросили – а как конкретно нам это всё пригодится в жизни. В команде поддержки есть разные люди, в том числе инженеры вроде меня. А инженер как раз должен быть в состоянии ответить на такой вопрос. Но и на остальные вопросы мы постараемся ответить.

# Использование переноса обучения в реальных задачах

Итак, о чем мы поговорим сегодня.

Первым делом, мы оглянемся на пройденный материал. В первой части лекции у нас будет резюме по темам файнтюнинга и преобразования признаков. Это темы двух прошлых лекций – и, собственно, это два основных приложения, которые должны нам приходить на ум когда речь идет о transfer learning. К ним я добавлю мультимодальные сети и еще пару трюков, которые нужны в переносе обучения.

Во второй части лекции мы поговорим о многозадачных нейросетях. На примере больших лингвистических моделей, посмотрим, как одну и ту же нейросеть можно использовать в разных задачах. Даже модели прошлого поколения, такие как BERT можно использовать в реальных решениях.

И в конце мы поговорим о том, где можно найти эти нейросети или сервисы на их основе. Сейчас можно найти много списков “нейросетей для бизнеса” или “нейросетей для творчества”, но часто там собран отнюдь не перечень нейросетей, которые технически можно скачать и использовать в проекте. Обычно там речь идет о сервисах, к которым можно обращаться. Не стоит делать вывод, что одна нейросеть может решить заявленную задачу, например управлять вашими встречами и календарями. Эту задачу скорее всего решают и обычные алгоритмы, и одновременно – несколько нейросетей для синтеза текста.

Перейдем к лекции.

# Файнтюнинг

Наш первый прием назывался “дообучение” или “файнтюнинг”. Напомню – это тот прием, где мы брали нейросети, завораживали большую часть ее компонентов, то есть отказывались обучать большую часть ее параметров, или, как часто говорят – весов. И мы экономили ресурсы, выделяя минимум параметров для переобучения – либо последние слои нейросети, либо какой-нибудь дополнительный обучающийся алгоритм из набора классических методов машинного обучения.

Веса мы замораживали по двум причинам. Во-первых, , потому что то дообучение целой нейросети – не очень хорошая идея. Большие нейросети скорее всего не влезут в наши с вами видеокарты, тем более что для кратного ускорения обучения, в память нужно поместить столько копий, сколько получится. При том что ChatGPT невозможно даже запустить на рабочем лэптопе, не то чтобы дообучить. А маленькие нейросети легко забывают то, чему их учили раньше.

Если новые данные сильно отличаются от данных, на которые сеть раньше обучалась, лучше не переобучать всю нейросеть, а поискать другую стратегию. Дообучение, в ходе которого нейросеть рискует забыть все, чему ее учили раньше – это уже задача исследовательского спектра, а мы тут пытаемся создать нужный сервис из того, что уже доступно.

Во-вторых, замороженная часть нейросети ведет себя как генератор признаков, для этого мы ее и оставили за скобками. Но как определить, сколько параметров нужно дообучить? Считается, что время, потраченное на поиск наилучшего места для разделения сети на обучаемую и необучаемую часть, к сожалению, никогда не окупается. Лучше надстроить над нейросетью новый классификатор или метод регрессии из классических методов.

Каким простым путем лучше пойти при дообучении? Если целевой датасет не очень большой и состоит из данных, которые похожи на исходные данные, то лучшее решение – заморозить всю нейросеть, заменив только последний слой. На что заменить? Например, на слой, который позволяет отвечать да или нет. Или выбирать значение из списка. На какой-то слой, который подходит для вашей целевой задачи.

Другой экстремум – если целевой датасет достаточно большой. Тогда можно попробовать разморозить всю нейросеть, но использовать предобученные веса как отправную точку для обучения. Это всё равно будет лучше, чем расставить случайные числа вместо всех коэффициентов и обучать сеть с нуля. В средах для обучения нейросетей вы скорее всего найдете найдете параметр pretrain, или его варианты. Когда вы создаете нейросеть по лекалам какой-то существующей архитектуры, претрейн указывает, откуда взять предобученные значения для инициализации сети.

Другие случаи – когда целевой датасет не похож на датасет исходной задачи, не выглядят как хорошие кандидаты для переноса обучения. Лучше подобрать другую сеть с похожим датасетом на ваш, целевой.

Еще скажу кое-что из технических соображений. В режиме обучения, нейросеть занимает в памяти гораздо больше места из-за разных коэффициентов и кэшированных значений, которые позволяют нейронам обучаться или делать это быстрее. Можно просто брать размер нейросети в мегабайтах и умножать хотя бы на 4. Столько в памяти видеокарты будет занимать BERT, который на диске занимает 100 или 200 мегабайт. Для ускорения обучения нужно поместить в память видеокарты, она же VRAM, как можно больше копий обучаемой модели. Так что разделите количество VRAM на объем одной копии нейросети и посмотрите, на сколько примерно можно будет умножить скорость обучения. Вроде бы это совсем технические вопросы. Но обсуждая файнтюнинг нужно учитывать время и ресурсы.

## Дообучать нейросеть, не дообучая ее веса

Большие лингвистические модели занимают огромное количество памяти. Но не обязательно дообучать всю модель. Можно обучать какую-нибудь структуру, которая умножается или добавляется ко всем весам, и было бы отлично, если бы эта структура содержала намного меньше параметров, чем исходная нейросеть. Если поместить все веса в матрицу и сделать ее декомпозицию, например методом SVD, можно получить пару-тройку матриц более низкого ранга, то есть в которых меньше независимых элементов. Перемножив эти матрицы, мы снова получим огромную матрицу коэффициентов. Но, что приятно, если обучать коэффициенты не исходной матрицы, а низкорагновых матриц, обучение пройдет гораздо быстрее, а обучаемые структуры легче поместятся в память. Все это я говорил про метод LoRA, или метод низкоранговой адаптации, который отлично работает для дообучения огромных нейросетей. Исходная нейросеть замораживается, обучаются коэффициенты, размещенные в низкоранговых матрицах, из которых потом собирается исходная матрица весов.

Пожалуй, это звучит сложновато. Но метод LoRA работет и с лингвистическими нейросетями, и с графическими. С его помощью можно обучить нейросеть сочинять более адекватную поэзию или анекдоты, или рисовать персонажей в определенном, нужном вам стиле, используя одни и те же замороженные нейросети в качестве backbone, то есть исходных сетей.

## Нейросеть как преобразователь признаков

Вторая тема, которую мы обсудили на прошлой лекции – это было использование нейросети как преобразователя признаков.

Напоминаю, в чем там дело. Глубокие нейросети состоят из большого количества слоёв. Вся нейросеть – это алгоритм, и каждый слой по-отдельности – тоже небольшой отдельный алгоритм. Что он принимает на вход? Первый слой принимает картинки, или текст, или еще какое-то описание изучаемых объектов. Второй слой принимает на вход то, что находится на выходе первого слоя. Это тоже описание входного объекта. Но оно уже было переработано первым слоем. И так далее, каждый следующий слой получает немного модифицированное описание исходного объекта. Оно доходит до нас в виде набора чисел – вектора, матрицы, нескольких матриц... На прошлой лекции мы обсуждали идею, что с этим численным описанием тоже интересно работать, и на разных уровнях нейросети нас потенциально ждет много полезной информации.

Ладно, не на всех. В нейросетях есть технические слои, которые даже не обучаются, а просто помогают делать вычисления. Есть и слои, которые обучаются, но состоят не из нейронов. Например, слои нормализации или пулинга. С ними мы тоже поработаем, но я постараюсь держаться подальше от специфической математики нейросетей.

Возьмем глубокую, то есть многослойную сеть для классификации изображений. Первые слои преобразуют картинку в сотни новых изображений. Чем ближе к выходу из нейросети, тем больший отпечаток финальной задачи несут эти сотни фасеточных изображений. И это верно, ведь нейросеть обучена меньше обращать внимание на несущественные, случайные характеристики объектов, и доставать из изображений сложные признаки, которые комбинируются в предсказанный класс. Итак, почти на каждом слое глубокой нейросети находится интересная для нас информация, и вектор, в который можно собрать всю эту информацию, называется эмбеддингом.

## Как исследовать такие описания

С эмбеддингами можно сделать много интересного. Например, исследовать их интерполяцию, находить новые объекты на некотором расстоянии от уже известных. Или использовать их в качестве метрики – слабые значения выходных эмбеддингов VGG могут говорить о том, что на картинке не изображено ничего реалистичного.

Исследовать эмбеддинги может быть полезно для валидации нейросети. Такие испытания позволяют определить, нет ли паразитных зависимостей которые выучила нейросеть, и на которые она будет обращать внимание вопреки вашему

обучению. Однако это отдельное очень сложное ответвление на стыке инженерии нейросетей и информационной безопасности.

Как исследовать чувствительность каждого слоя нейросети? Это изображение – результат исследования активации отдельно взятых нейронов в нейросети. Исследователи порождали изображения, которые заметнее всего активировали именно этот нейрон. Просто изменяя окраску каждого пикселя искали, какое изображение вызовет максимальный отклик этого нейрона. Причем это делалось довольно давно, задолго до MidJourney и StableDiffusion. Таким образом получаются довольно абстрактные изображения, которые предположительно заставляют нейросеть создавать яркие эмбеддинги, с разбивкой по слоям. Ближе ко входу это более абстрактные признаки, линии, круги. Ближе к выходу мы уже видим высокоуровневые детали объектов, которые сеть будет классифицировать.

В ходе этой лекции мы будем переключаться между картинками и текстом. Что касается чувствительности текстовых нейросетей, большие лингвистические модели открывают совершенно новый пласт исследований для вычислительной лингвистики. Например можно изучать, какие знания заключены в нейросети, умеет ли ли она соотносить верх и низ, прошлое и будущее, как она видит карту мира... Правильно ли она располагает города на карте мира и насколько точна корреляция между реальными событиями и датами, которые возвращает нейросеть для этих событий. Я приведу примеры в библиографии, но такие исследования ведутся скорее методом опроса нейросети, то есть в режиме чат-бота, так что это не совсем про перенос обучения.

## Работа с каждым слоем по-отдельности

Пара слов про приемы дообучения. Кроме заморозки, можно также изменять скорость обучения для разных слоёв. Скорость обучения – это параметр, который регламентирует, насколько сильно вообще могут меняться параметры слоев, когда им встречается пример, на который нейросеть даёт неправильный ответ. Довольно трудно подобрать правильную скорость обучения, тем более изменять ее в ходе тренировки. А еще сложнее – подбирать разную скорость для разных слоев, например чтобы выходные слои обучались быстрее, а более глубокие, близкие к входу слои обучались медленнее. Я находил упоминание этого метода, пусть он будет в вашем инструментарии, но он не очень промышленный.

Дальше, при создании предобученных нейросетей, например YOLO, в ходе обучения иногда переключают нейросеть с одной задачи на другую. Первые 90 эпох обучают одной задаче, последние 10 эпох – более медленной и скрупулезной задаче. Сразу можете представить, что этот метод дообучения тоже не очень подходит для промышленности. Когда у вас есть исследовательская команда которая может неделями искать архитектуру или делать абляционное исследование, это может сработать. Абляционное исследование – это когда в архитектуре так много нагромождений, что вы начинаете удалять случайные элементы и проверять, нужны ли они были в архитектуре, или нет.

А еще в техническом плане нужно знать, что нейросети для обработки текста обычно выдают ужасно много информации. По одному эмбеддингу для каждого слова в предложении, а точнее – для каждой тестовой единицы, для каждого токена. Токенами могут быть и слова, и их кусочки, и отдельные буквы. Как получить эмбеддинг слова или фразы, если нейросеть возвращает длинное полотно эмбеддингов? В обработке текста часто берут среднее значение или максимум от всех эмбеддингов, чтобы попробовать понять смысл слова или смысл текста. Этот прием называется агрегацией информации.

Но выполнять агрегацию не всегда обязательно. При подаче текста в языковые нейросети он обычно снабжается специальными метками, например “начало фразы” или “классифицируй”. Они воспринимаются как самостоятельные слова, и их эмбеддинги заведомо имеют определенный смысл. Например, для классификации предложений можно агрегировать эмбеддинги всех слов, а можно воспользовться эмбеддингом [CLS] в квадратных скобках.

А еще эмбеддинги можно использовать в таких задачах, как:

- Поиск, когда производится ранжирование по мере близости искомой строки и потенциальных результатов
- Кластаризация, когда объекты комбинируются по сходству репрезентаций
- Выдача рекомендаций, когда объекты со схожей репрезентацией можно рекомендовать пользователю
- Детекция аномалий, когда по степени близости текущих и эталонных наблюдений судят о корректности и надежности новых данных
- Определение рассогласования, когда по разности репрезентаций судят о близости объектов

# Мультимодальность

Еще один трюк, который работает во многом благодаря переносу обучения – это создание мультимодальных нейросетей. Под модальностью понимают определенный тип информации, а также формат хранения, который отличает ее от других. Текст, видео, аудио, изображения, графы, и так далее.

Задача обучения мультимодальных нейросетей состоит в том, чтобы обучать алгоритмы на базе данных разной модальности. Можно себе представить такие примеры применений мультимодальных нейросетей:

Чаще всего мультимодальные сети работают с сочетанием наиболее популярных модальностей – текстом, изображениями и аудио.

Предобученные нейросети позволяют перескочить через этап совмещения несовместимых данных. Достаточно просто взять несколько предобученных нейросетей и сместить эмбеддинги объектов. Можно либо просто поставить один вектор вслед за другим, либо обучить линейную функцию, которая делает какой-то третий вектор из двух предыдущих. Это первый прием, который позволяет совместить нейросети разной модальности ради одной общей цели.

Второй прием, очень трудный, это последовательно склеить несколько нейросетей при помощи линейного слоя. Это гораздо сложнее, чем обучить что-либо на сочетании двух эмбеддингов. Так получаются мультидоменные чатботы, которые могут и картинку вернуть, и создать текстовое описание. Вы можете посмотреть, как это сделано в нейросетях FROMAGE или Flamingo, но давайте я приведу пример осязаемого метода, который вы можете использовать в своих проектах.

Я хотел рассказать про яркий пример успеха последних лет – метод CLIP, Contrastive Language Image Pretraining. Можно перевести как “конфронтационное обучение на изображениях и тексте”, но у Сбера встречается и “контрастивное”.

CLIP – это прием дообучения двух нейросетей. Берется набор данных с изображениями и подписями к ним. И берется две нейросети, одна для изображений, другая для текста. Например, ResNet и BERT. Их эмбеддинги не будут иметь одинаковый размер, поэтому после каждой нейросети ставится линейный слой, который преобразует признаки к нужной размерности. Как это делается? Если первая нейросеть выдает 700 признаков, а вторая – 300, то можно обучить переходник, функцию y = A * x + B, которая сделает из вектора x вектор y нужной размерности. Положим, у нас оба вектора стали размерностью 300.

Итак, в сценарии CLIP нейросети дообучаются на картинках и их описаниях, а цель обучения заключается в том, чтобы эмбеддинги обеих нейросетей стали максимально похожими. То есть сближаем нужные эмбеддинги и удаляем те, которые не должны сочетаться, отсюда и конфронтация или Мы получаем гибридную нейросеть, куда можно подать текст и получить эмбеддинг соответствующей картинки. А можно наоборот, подать картинку и получить эмбеддинг текста. Перебором текстовых затравок можно подобрать адекватную подпись к изображению. Пройтись по словарю и слово за словом подправить текстовую строку так, чтобы эмбеддинг текста лучше всего сочетался с эмбеддингом изображения. Текст может быть таким: фотография собаки, фотография кошки, нарисованная птица, нарисованная мышь... Видите, как подставляя разные слова можно подобрать оптимальную подпись.

CLIP нельзя строго говоря назвать нейросетью, это несколько нейросетей, обученных работать вместе. Но вы можете скачать эту комбинацию и придумать, как ее использовать. Без этой технологии не было бы первых вменяемых генераторов изображений, того самого DALLe и стула в форме авокадо. В 2021 году нейросеть DALLe показала довольно хороший уровень генерации картинок, но она работала в связке с CLIP, который помогал анализировать сгенерированное изображение и выступал в роли дискриминатора. Он ранжировал изображения по степени похожести на текстовый запрос. Он принимал на вход текст и каждое из десятков изображений, которые возвращал DALLe, и очень убедительно позволял определить несколько наилучших результатов.

Приведу пару примеров мультимодальных задач. Например, это распознавание настроения и состояния человека по нескольким источникам данных. Это нужно в медицине или в спорте. Во-вторых, есть приложениях, когда нужно понимать медиа-контент, представленный текстом, видео, аудио или изображениями. В этой роли, например, использовался CLIP. Потом, можно упомянуть создание подписей к видео. Ответ на вопрос, сформулированный частично в виде текста, частично в виде изображения. Мультимодальный поиск информации. И почти любая прикладная сфера – робототехника, автоматическое вождение, везде, где нужен анализ сцены и принятие решений Даже в финансовой сфере используются мультимодальные подходы – когда нужно предсказать риски на основе анализа клиентских обращений и изображений сканированных документов и договоров.

Итак, файнтюнинг, использование эмбеддингов, вариация параметров обучения, аграгация, а еще – мультимодальные сети. Вот уже немного пищи для размышления.

Давайте теперь поговорим о неочевидных применениях текстовых нейросетей. Эти приемы – отличный пример того, что хорошо работающая технология уже не нуждается в рекламе. Сегодня у всех на слуху чат-боты, и конкретно – нейросети вроде ChatGPT и GigaChat или Яндекс Алисы. В той или иной степени, это всё – лингвистические нейросети, из которых путем программной инженерии или дообучения сделали чат-ботов. Я бы хотел, на примере таких нейросетей, рассказать еще о нескольких приемах переноса обучения.

## Что может делать LLM

Итак, во второй части лекции я хотел бы подобраться к инженерным приемам, которые позволяют сделать что-то интересное из лингвистических нейросетей. Их невозможно перечислить все, я лишь дам несколько примеров. В прикладном плане вы будете поймете, какие разные задачи можно реализовать, просто скачав нейросеть. Для некоторых задач нужно дообучение, для других – немного программной инженерии.

Пройдемся по этим задачам.

Во-первых, классификация текста. Она может быть многоклассовой, или бинарной (хороший отзыв, плохой). С одной меткой на выходе или с несколькими, когда вы пытаетесь угадать сразу несколько классов, к которым относится ваш текст. Если из меток можно выстроить шкалу, например если вы предсказываете оценку по бальной шкале, то можно попробовать организовать задачу регрессии. Нейросеть будет принимать на вход предложение и возвращать число. Технически такая задача отличается совсем немного, а если у вас удобная графическая среда для обучения сетей, то задача может отличаться всего парой галочек в настройках. Классификация текста решается поиском хороших эмбеддингов.

Распознавание именованных сущностей – обычно под этим понимают поиск в тексте отдельных имен, названий, брендов, календарных дат, всего что можно выделить и поименовать как интересующую вас сущность.

Ответы на вопросы. Эту задачу мы сегодня видим очень часто. Ее решают частично при помощи огромных языковых моделей, которые предсказывают ответы, частично при помощи поиска по базе данных с ответами. Если удается подать на вход не только вопрос, но и контент разговора, у вас получается полноценный

чатбот. Но ведь система может и просто отвечать на вопросы. Такие системы обучаются на форумах технической поддержки и становятся простейшими синтетическими консультантами. Полноценный чат-бот должен решать Seq2seq задачу, или перевод из одной последовательности в другую. Но таким же образом делают и машинный перевод, и суммаризацию, то есть обобщение текста одним предложением, например.

Если при суммаризации текст синтезируется с нуля, это абстрактивная суммаризация. Но можно организовать и экстрактивную суммаризацию, а таком случае ваша система будет искать релевантные куски текста и возвращать их в качестве резюме (по английски, summary).

Кстати, побеждают ли такие модели людей в языковых задачах? Смотря как измерять. Вот на экране сравнение современных моделей по метрике качества их ответов на датасет, где модель должна была продемонстрировать наличие здравого смысла. Как вы видите, результаты далеки от 100%. Хотя само то, что мы проверяем этот критерий у работающего автоматически, предобученного алгоритма, не имеющего выхода в интернет – это, согласитесь, уже вызывает некоторое восхищение.

Как ни странно, самое простое применение языковых моделей – не самое полезное в производстве. Это уже упомянутое заполнение пропущенных слов.

## Почему все-таки BERT?

Напомню, почему мы пользуемся BERT, а не GPT. Эту нейросеть легко скачать и запустить на собственном компьютере. Она сравнительно небольшая, меньше гигабайта, и выполняет вычисления довольно быстро – обращение к ней обычно занимает меньше секунды. И, что приятно, BERT работает у вас локально, не нужно отправлять чувствительные данные во внешнюю систему.

Работа с более сложными нейросетями, например GPT, строиться через общение с сервером. Вы отправляете запросы на сайт, где работает такая нейросеть, получаете ответы, и тогда уже применяете любую программную обвязку: вставляете ответы в шаблон, или выбираете наилучший вариант из множества, и так далее. В этих моделях слишком много параметров, чтобы запускать их локально, но, на всякий случай скажу – вы можете получать из них эмбеддинги при помощи запросов, если это будет вам полезно. Но я боюсь, что такое исследование займет слишком много времени.

К тому же, не все возможности, обещанные современными диалоговыми нейросетями, работают как нужно. Все, кто пытался создавали рекламные кампании, например, через генераторы изображений и текста, ощутили это на себе. Онлайн сервисы не позволяют заменить нормального специалиста, который адекватно интерпретирует задачу и способен защищать свои решения. В общем, иногда лучше собрать продукт из более простых, опробованных базовых кирпичиков. И уж точно начинать освоение лингвистических моделей лучше с чего-то попроще.

## Заполнение пропущенных слов в предложении

Основная задача, на которой учатся современные нейросети – это задача угадывания пропущенного слова в предложении. Что это вообще за задача?

Может показаться, что эта задача хорошо подойдет для решения каких-нибудь тестов на знание языка. Но это хороший пример того, что задача, на которую нацелена нейросеть, и прикладной сценарий использования нейросети – это совсем не одно и то же. Дело в том, что заполнение пропущенных слов – главная задача, которая и позволила обучить предвестников современных больших нейросетей, а именно BERT.

В интернете очень много текста. Но этот текст совершенно невозможно разметить вручную. Поэтому для прорыва нужна была такая задача, которая не требует разметки на правильные и неправильные примеры. То есть все примеры в интернете – правильные, а нейросеть будет учиться угадывать порции текста, которые от нее искусственно спрятали. Эта постановка характеризует задачи на самообучение, self supervised learning.

Первые сети BERT также обучались на десятке других задач, например должны были угадывать, следуют ли два предложения друг за другом, но исследования показали, что вторая задача не имела такого решающего влияния на качество сети, как первая, предсказывание пропущенного слова. И теперь все нейросети учат дистрибутивную семантику.

Я уже говорил, что при обучении BERT в предложение вставляются особые токены, которые обозначают пропущенные слова, или начало следующей фразы. Такие токены – это тоже слова из словаря, хотя и фиктивные. Например, перед самим предложением в сеть попадает токен CLS. Он обозначает, что последующая задача будет задачей классификации. После каждого предложения добавляется токен-разделитель NSP, next sentence prediction, предсказание следующего предложения. Встретив эти псевдослова, сеть получает сигнал по-особенному

сформировать их эмбеддинг. Эмбеддинг токена CLS можно использовать для классификации текста. А эмбеддинг NSP иногда используют для второстепенных целей, дообучая сеть специально для того, чтобы в этом токене скапливалась информация для решения целевой задачи.

Кстати, повторю, что для каждого токена нейросеть выдает отдельный эмбеддинг, это вектор из нескольких сотен или тысяч чисел. И каждый из них можно использовать в последующей задаче.

## Суммаризация

Кажется, что с помощью нейросети сразу же хочется решить задачу извлечения смысла из текста. Это задача суммаризации. Звучит очень заманчиво. Не нужно читать большой массив текста, нейросеть перескажет вам его в паре предложений.

Увы, для суммаризации сеть нужно дообучать. Датасетов по суммаризации не очень много, но обычно есть обходные пути. Возьмем научные статьи. К каждой статье прилагается резюме, а еще у статей есть названия. Из текста и аннотации, либо из аннотации и названия уже можно собрать датасет для абстрактивной суммаризации. Другой вариант суммаризатора, экстрактивный, должен не генерировать резюме, а подсвечивать в исходном тексте те слова, которые отражают его суть. То есть перед нами практически задача регрессии. Нужно пройтись по всем словам в тексте и каждому назначить какое-то число, а потом взять слова с самым большим счетом.

### Ответы на вопросы

У меня есть лишь опыт суммаризации с BERT без дообучения. И работало это 50/50, как некоторые современные генераторы текста или изображений. Возможно, вам будет нужна совсем другая задача, которая решается чуть лучше. Это ответ на вопросы по тексту. По моим ощущениям, такая задача решается чуть лучше задач по суммаризации. В этом случае на вход нейросети подается исходный текст в качестве контекста. Затем, например через какой-то токен-разделитель, подается вопрос. Нейросеть должна вернуть начало и конец порции текста, которая отвечает на этот вопрос.

Несколько онлайн-демонстраций этой технологии, кажется, убеждают, что вопросно-ответные системы лучше освоены сегодня, чем суммаризация.

https://dida.do/demos/question-answering

Самым распространенным датасетом для обучения нейросетей ответам на вопросы является датасет SQuAD, для русских нейросетей – SQuAD-Ru. Он содержит пары вопрос-ответ на русском языке. Но вам не придется дообучать нейросети на этом датасете, в репозиториях вроде DeepPavlov уже есть сети с названиями вроде "squad_ru_bert". Дальше эти модели нужно использовать согласно спецификации, можно ничего не дообучать.

### Синтаксический и морфологический разбор

Синтаксические деревья – основная информация для анализа смысла предложений. Возьмем два предложения. "Кубок не влез в чемодан потому что он был большим." И второе: "Кубок не влез в чемодан потому что он был маленьким." Без нейросетей довольно сложно ответить на вопрос, к какому слову в предложении относится местоимение "он". Это наглядный случай задачи на разрешение анафоры, то есть пониманию того, к чему относится местоимение в тексте.

Как это может быть устроено внутри? Исследователи берут BERT и дообучают ее так, чтобы по эмбеддингам каждого слова в предложении можно было воссоздать метку части речи, либо грамматическую функцию слова, либо положение в синтаксическом дереве. То есть опять всё упирается в процесс предобучения, а нам, как пользователям которые будут создавать собственные решения на базе этой нейросети, предстоит найти и скачать ту нейросеть, которая обучена согласно нашей целевой задаче.

## Дополнительные инструменты

Для лингвистических сетей существует еще одно направление развития кроме поиска новых задач и обучения на новых датасетах, и это включение функций из внешних инструментов прямо в лингвистическую нейросеть.

У всех лингвистических нейросетей есть общие проблемы.

Например, они плохо ориентируются в математике. Вызов внешнего калькулятора – это возможность использовать аппарат, предназначенный для математических вычислений,

Другая проблема: датасет на котором предобучались нейросети быстро устаревает, им имеет смысл давать доступ к инструментам на внешних серверах, например поисковику или энциклопедии.

В-третьих, чтобы лингвистические модели выдавали более обоснованную информацию, можно дать им доступ к библиографическим базам и просить подкреплять свои выводы цитатами. Таким образом они реже будут выдавать случайные последовательности фактов за истину

Если подключить к нейросети инструменты обработки эмбеддингов и трассировки вычислений, то можно извлечь чуть более информации, чем только финальный текстовый результат. Например, можно извлечь цепочку размышлений или концепции связанные с ответом, чтобы понять, какие смежные концепции обрабатывала нейросеть прежде чем решить, что вам ответить.

Так что подключение внешних инструментов, или аугментация лингвистических нейросетей, позволяет значительно расширить их возможности, но как объединить нейросеть с другим программным инструментарием – нетривиальный вопрос и интересная задача для вас, если вы хотите одновременное развиваться и в программной инженерии, и в машинном обучении.

## Как применять предобученные модели

Мы разобрали несколько применений предобученных моделей. Как можно обобщить их? Чтобы ещё раз пройтись по основным видам возможного использования, давайте разделим их на три класса, которые хорошо повторяют темы нашего курса.

Дообучение нейросетей – это самый затратный путь, в ходе которого мы, надеясь что целевая задача не очень отличается от исходной, обучаем нейросеть или ее часть выполнять какое-то действие, которое не было в нее заложено. Например, если для решения задачи нейросеть должна научиться принимать на вход текст, а на выходе выдавать поток меток грамматических классов, например – частей речи.

Второй подход – креативное использование эмбеддингов. Лингвистические нейросети дают нам очень много вариантов такого использования, мало того что они выдают новый эмбеддинг для каждого нового поданного слова, их еще и тренируют выдавать эмбеддинги по специальным командам. Кроме этого, эмбеддинги можно агрегировать. Например, суммировать, или брать среднее значение, или накапливать один за другим и использовать как описание целого текста. То же самое можно делать для нейросетей, создающих эмбеддинги изображений.

Третий метод использования – самый простой. Это использование сети без каких-либо изменений, когда ничего не требуется усложнять. Чем лучше становятся нейросети, тем легче использовать их просто в качестве алгоритма, который решает свою когнитивную задачу. И лучше всего это работает в современных чат-ботах, которым нужно подать на вход инструкции по формату ответа на вопрос, затем подать сам вопрос. Например, можно проинструктировать нейросеть выдать вам ответ в форме, которую можно передать в калькулятор, ведь мы заведомо знаем, что у нейросетей плохо с арифметикой. То есть чтобы научить нейросеть новым трюкам, не обязательно ее чему-то учить. Можно просто сформулировать запрос определенным образом. А для графических нейросетей – аугментировать или отфильтровать изображение.

## Мультитаскинг в тексте и в изображениях

Некоторые из приложений, которые мы обсудили – это попытки задействовать предобученные нейросети в задачах, на которых они не обучались. Так сказать, сменить целевую задачу. Нагляднее всего это направление, конечно, исследуется на больших лингвистических моделях. Их обучают на разных задачах, но самая популярная задача – это просто предсказание пропущенного слова при условии что мы знаем остальные слова в предложении. После этой задачи иногда проводится файнтюнинг на целевом датасете, а иногда дообучение под совершенно другую целевую задачу.

Нейросети, которые можно использовать сразу в разных задачах, на жаргоне, естественно английском, называются “мультитаск нейросетями”. Но с приходом переноса обучения, дисциплина многозадачных алгоритмов вроде бы получила свое финальное решение. На этом курсе мы как раз учимся использовать нейросети в разных целевых задачах.

На семинаре мы изучим задачу переноса стиля. Это, можно сказать, уже старинная задача, которую мы решим методом 2015 года. В ней мы возьмем готовую нейросеть и смешаем эмбеддинги нескольких уровней. Это можно сделать на популярной нейросети ResNet или VGG, и наверное есть нейросети которые лучше решают эту задачу, но имея такой стройматериал как эти две нейросети, лучше начать свой проект с их использования, чем искать дополнительно и обучать что-то новое.

Казалось бы, нейросеть ResNet обучалась для классификации изображений. Но она используется для извлечения признаков и в такой глобально популярной нейросети

как YOLO. Признаки разных уровней комбинируются, и на базе архитектуры YOLO выстраиваются решения и – для детекции объектов, и – для сегментации объектов нужного класса, и просто для сегментации целой сцены. Так сказать для семантической сегментации.

В задаче сегментации нейросеть изучает каждый пиксель сцены и выдает набор меток, где закодировано предположение – к какому объекту относится этот пиксель. Кстати, почти таким же образом решается задача раскраски изображений в оттенках серого. Нейросеть голосует за то, какой цветовой оттенок будет присвоен тому или иному пикселю, при том что яркость уже закодирована в виде того самого оттенка серого. Остается только угадать цвет, а для этого нужно сегментировать картинку.

Если у вас получится найти сходство между не совсем близкими задачами, такими как детекция и сегментация, то возможно вы сможете найти предобученным нейросетям еще одно новое применение, получив ценное ноу хау.

Думаю, вы уже в достаточной мере готовы обсуждать проекты и рассуждать на тему предобученных лингвистических нейросетей и того, как их использовать в реальных проектах. Прежде чем перейти к практике, поговорим о том, где берутся предобученные нейросети. О зоопарках и репозиториях готовых моделей.

## Каталоги нейросетей

Итак, все уже в курсе, что предобученную сеть можно скачать, а затем, при помощи небольшого оформляющего кода, выполнить с ее помощью что-нибудь полезное.

Вы легко найдете перечни “нейросетей для бизнеса” или “нейросетей на все случаи жизни”. Обычно это не перечни нейросетей, а перечни онлайн-сервисов. Они позволяют сделать много полезного, если с ними уметь работать. Да что там, по ним уже есть целые учебные специальности, например “промпт-инженерия”, или “генеративная иллюстрация”. Такие перечни содержат не совсем "нейросети". Обычно там приводятся наборы сервисов, под капотом у которых запущена большая архитектура для предобработки информации, подачи ее в нейросеть и выдачи результата. Создание таких сервисов – это тоже отдельная специальность, которая называется дата инженерия. Как и дата-аналитик, который пытается извлечь из

данных удобную для анализа информацию, так и дата-инженер делает работу сервисов на базе нейросетей быстрой и удобной. Если вам интересно именно это направление, я уверен что где-то рядом в этом курсе есть лекции по развертке нейросетей в продакшен, организации всяких фича сторов и дата лейков... Посмотрите в эту область, дата инженеры – очень ёмкая специализация, и таких людей всегда не хватает.

Пожалуй, самый респектабельный репозиторий нейросетей наследует название от эмодзи, изображающего улыбку и объятия: hugginface. Перечень нейросетей в этом репозитории, в общем-то, поражает воображение. Из мультимодальных сетей, здесь есть нейросети для генерации изображений по тексту, перевода изображений в текст, ответов на вопросы на базе изображений, перевод текста в видео, и чего здесь только нет. По крупным категориям, здесь еще есть нейросети для компьютерного зрения, для обработки естественного текста, аудио, таблиц, графов, для обучения с подкреплением...

Я бы также порекомендовал начинать поиск нужной нейросети с сайта paperswithcode или какого-нибудь аналога. Бывает не очень легко разузнать, как же ваша задача называется на английском. Есть всякие странные названия, почти жаргонные. Но пройдясь по перечню на paperswithcode вы обязательно что-нибудь найдете, а также изучите историю применения разных моделей для конкретной задачи. Как правило, там же можно посмотреть, как запускать эти модели.

Третий источник, где стоит поискать – это платформа для соревнований kaggle. Скажем, я буду искать задачу сегментации медицинских снимков. Если набрать название этой задачи и слово kaggle, то вы получите список решений. Там же будут работающие примеры использования, упакованные в стандартный формат для анализа данных – в ноутбуки. На всех перечисленных платформах можно найти и решения, и датасеты, и возможность пообщаться.

В библиографии я дам ссылку на репозиторий Deep Pavlov. Этот русский научно-исследовательский проект начался в 2020 году и сейчас разрабатывает, пожалуй, все нейросети для разговорного ИИ, которые можно себе представить. Там они лежат с открытым исходным кодом, чтобы на их основе можно было создавать чат боты, вопросно-ответные системы и сложные ИИ-ассистенты.

В общем, в этой сфере весьма легко утонуть в информации. Но такая уж сегодня специфика когнитивных задач. Давайте немного поговорим о том, в каком формате скачивать готовые нейросети.

## В каком формате лежат нейросети

Формат нейросети, которую вы скачаете, будет зависеть от уже проведенного процесса обучения. Существует несколько крупных инструментов для обучения нейросетей и гораздо больше небольших библиотек, которые тем не менее стараются работать с форматами крупных пакетов. Например, есть много инструментов для визуализации всего, что происходит в сети, слой за слоем.

В ходе длительного обучения, когда процесс может пойти не в ту сторону, принято делать копии текущего состояния нейросети, сохраняя так называемые чекпойнты. Они занимают место на жестком диске, но не в оперативной памяти. Финальная нейросеть будет таким же чекпойнтом процесса обучения, только последним по счету. Чекпойнт – это большой набор параметров нейросети, которые мы сохраняем на диск, с целью восстановить по ним все результаты оптимизации для некоторой архитектуры. Ведь в ходе обучения мы оптимизируем несколько тысяч, а в случае полезных предобученных сетей – много миллионов параметров. Эти числа оказываются в файлах-чекпойнтах. Они так называются, потому что если обучение зайдет в тупик и будет желательно поменять какие-то параметры, проектировщику не обязательно будет нужно начинать с самого начала. Можно будет загрузить чекпойнт, и продолжить обучать заданную архитектуру с одной из последних сохраненных точек, но с другими параметрами – скоростью обучения, целевой функцией... Чтобы загрузить такую модель из файла, нужно сперва создать в памяти пустую архитектуру, потом заполнить ее коэффициентами из файла. Так что нет смысла хранить архитектуру модели в чекпойнте, только несколько миллионов коэффициентов, разложенных по порядку. Но это техническая деталь, которая нам не очень важна.

Итак, форматов хранения моделей может быть множество, если искать один формат, который понимают самые различные среды – то это формат ONNX – открытый формат для обмена нейросетями. Его разработку ведут самые крупные международные компании, он имеет открытый исходный код, в нем хранится как архитектура, так и коэффициенты. И он поддерживает достаточно видов нейросетей, слоев и нейронов.

Но, как правило, вы будете отталкиваться от готового решения. И в таком случае вам не придется волноваться о формате нейросети, ведь в шаблоне решения, которое вы найдете, будут все инструкции для загрузки предобученной нейросети в том формате, в котором это получилось сделать у вашего предшественника. Я лишь хотел обозначить разницу между файлами вроде ONNX или MAT-файлами для среды вроде MATLAB, где будет лежать вся нейросеть целиком, и чекпойнтами

обучения сред вроде PyTorch и TensorFlow, где по большей части будут лежать только коэффициенты, а описание архитектуры модели будет рядом, но в коде.

## Заключение

Давайте подытожим наше длительное обсуждение. Предобученные нейросети позволяют решить простые когнитивные задачи без дополнительного обучения. С их помощью можно реализовать алгоритмы, принимающие решения, или упростить разметку данных и улучшить их качество, или смешать данные разной модальности, или диверсифицировать набор данных благодаря генерации. И они имеют гораздо больше применений, чем я успел озвучить.

Предобучение – это возможность упаковать вычисления заранее, выполнить их на мощном железе в удобном темпе, а потом использовать их в своем продукте. А еще, предобучение – это часть исследовательского процесса, и оно потребляет много вычислительных ресурсов, как и любое исследование. Но одно найденное семейство успешных моделей радикально сокращает затраты на обучение у всех компаний, которые это решение реализуют.

Также что касается информационной безопасности. Не сказать, чтобы использование широко доступных моделей делало систему более устойчивой к атакам. У популярных моделей чаще находятся уязвимости. Например – если специальным образом сконструировать текстовый запрос или добавить специального вида шум в изображение можно вынудить систему выдать неверный результат, что по крайней мере может нанести репутационный ущерб компании-поставщику сервиса, где используется та или иная нейросеть. Поисками методов защиты занимается отдел валидации, который должен проверять модели перед введением в обращение, ну и отдел информационной безопасности, наверное, должен подключаться.

Если вам предстоит карьера в области анализа данных, то успех ваших решений будет зависеть от приемов, которыми вы располагаете. Как подбирать идеальную модель? Есть база. Например, табличные данные лучше попробовать классифицировать методом случайного леса, random forest. Но успех ваших программ для анализа данных будет зависеть от вашей склонности комбинировать методы предобработки, объединять методы и модели между собой, выстраивать длинные цепочки.

Я надеюсь, я убедил вас, что перенос обучения – один из важнейших методов в аналитике, и что теперь вы понимаете, почему он часто используется в анализе данных. Мы разобрали, как делать файнтюнинг, как получать эмбеддинги, и как выстраивать более сложные решения на основе кусочков предобученных моделей. Если вам что-нибудь понравится из библиографии, то почему бы не заглянуть и не почитать статью в оригинале или в переводе.

И, конечно, я желаю вам удачи и надеюсь вас увидеть на семинаре.