|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Фундаментальные науки»

КАФЕДРА «Вычислительная математика и математическая физика» (ФН-11)

**РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

*Исследование и разработка глубоких нейросетей для решения задач кластерного анализа текстов*

Дисциплина: *Численные методы*

Студент группы ФН11-62Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** И.Д. Борисов

(Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Д.А. Чебаков

(Подпись, дата) (И.О.Фамилия)

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**РЕФЕРАТ**

Отчёт 36 с., 17 рис., 8 источника.

НЕЙРОННЫЕ СЕТИ, ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА, ATTENTION, TRANSFORMER, BERT, USE, DAN.

Цель работы – исследовать нейросетевые модели USE и BERT для кластеризации текстов, разработать ансамблевую нейросетевую модель.

**СОДЕРЖАНИЕ**

**с.**

**РЕФЕРАТ……...…………………………………………………………...2**

**ВВЕДЕНИЕ………...………………………………………………………4**

**1 Введение в машинное обучение……………………………………….5**

* 1. **Основные термины………………………………………………...5**
  2. **Введение в нейронные сети……………………………………….8**

**2 USE………………………………………………………………………11**

**2.1 Deep Averaging Networks (DAN)…………………………………13**

**2.1.1 Neural Bag-of-Words Models (NBOW)…………………...15**

**2.1.2 Идея DAN…………………………………………………...16**

**2.1.3 Улучшение DAN…………………………………………...17**

**2.1.4 DAN выводы………………………………………………..20**

**3 Механизм attention…………………………………………………….21**

**4 Transformer архитектура……………………………………………...23**

**5 BERT…………………………………………………………………….27**

**5.1 Процесс обучения BERT………………………………………….29**

**5.2** **Задачи предварительного обучения…………………………….30**

**5.3 Точная настройка BERT………………………………………….32**

**5.4 BERT выводы……………………………………………………...33**

**ЗАКЛЮЧЕНИЕ………………………………………………………….35**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ…………………36**

**ВВЕДЕНИЕ**

Такие общеизвестные бренды, как Echo (Alexa), Siri и Google Translate, объединяет по крайней мере одно: это все программные продукты, производные от приложе­ний для обработки написанных на естественном языке текстов (Natural Language Processing, NLP). Термин NLP относится к решению практических задач с помощью приемов понимания текстов, включающих применение статистических методов (с использованием воз­можностей лингвистики или без). Это «понимание» текстов достигается главным образом за счет их преобразования в пригодные для вычислений представления (representations) в виде дискретных или непрерывных комбинаторных структур, таких как векторы, тензоры, графы и деревья.

Обучение подходящих для конкретной задачи представлений на основе дан­ных (в данном случае текста) — предмет машинного обучения (machine learning). Машинное обучение (МО) применяется для анализа текстовых данных уже более трех десятилетий, но в последние десять лет набор методов машинного обучения, известный под названием глубокого обучения (deep learning), особенно эволюцио­нировал и доказал свою эффективность для различных задач искусственного ин­теллекта (ИИ) в сферах NLP, распознавания речи и машинного зрения. Именно поэтому в данной курсовой работе мы будем исследовать сферу NLP с помощью глубокого обучения.

Целью данной курсовой работы является исследование области обработки текстов на естественном языке.

Задачами курсовой работы являются:

- исследование нейросетевых моделей USE и BERT для кластеризации текстов;

- разработка ансамблевой нейросетевой модели;

- проведение экспериментов по улучшению качества ансамблевой нейросетевой модели.

**1 Введение в машинное обучение**

**1.1 Основные термины**

Наблюдаемые величины (наблюдения, observations) – сущности, поведение ко­торых мы хотим предсказать. Наблюдаемые величины обозначаются x. Иногда мы будем называть их входными данными (inputs).

Целевые переменные (targets) – соответствующие наблюдаемым величинам метки. Обычно именно их мы и предсказываем. Следуя общепринятой нотации в машинном/глубоком обучении, мы будем их обозначать y. Иногда эти метки называют контрольными значениями (ground truth).

Модель представляет собой математическое выражение или функцию, при­нимающую на входе наблюдаемую величину x и предсказывающую значение целевой метки.

Параметры – иногда называются также весами и служат для параметризации модели. Их обычно обозначают (от англ. weights) или .

Предсказания, называемые также оценками (estimates), представляют собой зна­чения целевых переменных, предсказанные моделью по наблюдаемым величи­нам. Для их обозначения мы будем использовать «шляпку». Так, предсказание целевой переменной обозначается .

Функция потерь (loss function) – это функция, служащая мерой отклонения предсказания от целевой переменной для наблюдений из обучающей последо­вательности. Функция потерь ставит целевой переменной и ее предсказанию в соответствие скалярное вещественное значение, называемое потерями (loss). Чем меньше значение потерь, тем лучше модель предсказывает целевую пере­менную. Мы будем обозначать функцию потерь L.

Embedding – в NLP означает процесс или, чаще, результат процесса преобразования языковой сущности – слова, предложения, параграфа или целого текста в набор чисел – числовой вектор

ИЛИ

это сопоставление произвольной сущности (в нашем случае слову или предложениям) некоторому вектору.

Meta-embeddings – это совокупность embedding’ов.

Encoders – кодировщики.

Auto-encoders – это нейронные сети прямого распространения, которые восстанавливают входной сигнал на выходе. Внутри у них имеется скрытый слой, который представляет собой код, описывающий модель. Auto-encoders  конструируются таким образом, чтобы не иметь возможность точно скопировать вход на выходе. Обычно их ограничивают в размерности кода (он меньше, чем размерность сигнала) или штрафуют за активации в коде. Входной сигнал восстанавливается с ошибками из-за потерь при кодировании, но, чтобы их минимизировать, сеть вынуждена учиться отбирать наиболее важные признаки.

ИЛИ

Auto-encoders – специальная архитектура искусственных нейронных сетей, позволяющая применять обучение без учителя при использовании метода обратного распространения ошибки. Простейшая архитектура auto-encoders – сеть прямого распространения, без обратных связей, наиболее схожая с перцептроном и содержащая входной слой, промежуточный слой и выходной слой. В отличие от перцептрона, выходной слой auto-encoders должен содержать столько же нейронов, сколько и входной слой.

Основной принцип работы и обучения сети auto-encoders – получить на выходном слое отклик, наиболее близкий к входному. Чтобы решение не оказалось тривиальным, на промежуточный слой auto-encoders накладывают ограничения: промежуточный слой должен быть или меньшей размерности, чем входной и выходной слои, или искусственно ограничивается количество одновременно активных нейронов промежуточного слоя – разрежённая активация. Эти ограничения заставляют нейросеть искать обобщения и корреляцию в поступающих на вход данных, выполнять их сжатие. Таким образом, нейросеть автоматически обучается выделять из входных данных общие признаки, которые кодируются в значениях весов искусственной нейронной сети. Так, при обучении сети на наборе различных входных изображений, нейросеть может самостоятельно обучиться распознавать линии и полосы под различными углами.

Чаще всего auto-encoders применяют каскадно для обучения глубоких (многослойных) сетей. auto-encoders применяют для предварительного обучения глубокой сети без учителя. Для этого слои обучаются друг за другом, начиная с первых. К каждому новому необученному слою на время обучения подключается дополнительный выходной слой, дополняющий сеть до архитектуры auto-encoders, после чего на вход сети подается набор данных для обучения. Веса необученного слоя и дополнительного слоя auto-encoders обучаются при помощи метода обратного распространения ошибки. Затем слой auto-encoders отключается и создается новый, соответствующий следующему необученному слою сети. На вход сети снова подается тот же набор данных, обученные первые слои сети остаются без изменений и работают в качестве входных для очередного обучаемого auto-encoders слоя. Так обучение продолжается для всех слоев сети за исключением последних. Последние слои сети обычно обучаются без использования auto-encoders при помощи того же метода обратного распространения ошибки и на маркированных данных (обучение с учителем).

**1.2 Введение в нейронные сети**

Нейронная сеть — это последовательность нейронов, соединенных между собой синапсами. Эта структура может анализировать, запоминать различную входящую информацию и даже воспроизводить ее из своей памяти.

Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений подобных тем, что делает человеческий мозг. Самыми распространенными применениями нейронных сетей является:

Классификация — распределение данных по параметрам. Например, на вход дается набор людей и нужно решить, кому из них давать кредит, а кому нет. Эту работу может сделать нейронная сеть, анализируя такую информацию как: возраст, платежеспособность, кредитная история и многое другое.

Предсказание — возможность предсказывать следующий шаг. Например, рост или падение акций, основываясь на ситуации на фондовом рынке.

Распознавание — в настоящее время, самое широкое применение нейронных сетей. Нейронные сети задействованы в таких областях, как: автопилотные автомобили, распознавание лиц, различных целей и многое другое.

Нейрон — это вычислительная единица, которая получает информацию, производит над ней различные вычисления и передает ее следующему нейрону через синапс. Они делятся на три основных типа: входной (синий), скрытый (красный) и выходной (зеленый) (см. Рисунок 1). В том случае, когда нейронная сеть состоит из большого количества нейронов, вводят термин слоя. Соответственно, есть входной слой, который получает информацию, n скрытых слоев, которые обрабатывают входную информацию, и выходной слой, который выводит результат. У каждого из нейронов есть 2 основных параметра: входные данные (input data) и выходные данные (output data). В случае входного нейрона: input=output. В остальных, в поле input попадает суммарная информация всех нейронов с предыдущего слоя, после чего, она нормализуется, с помощью функции активации и попадает в поле output.



Рисунок 1 – Пример нейронной сети

Синапс – это связь между парой нейронов. Каждый синапс обладает одним параметром, называемым весом синапса. С помощью этого параметра нейронная сеть изменяет входную информацию при переходе от одного нейрона к другому.

Функция активации — это способ нормализации входных данных (мы уже говорили об этом ранее). То есть, если на входе у вас будет большое число, пропустив его через функцию активации, вы получите выход в нужном вам диапазоне. Функций активации достаточно много поэтому мы рассмотрим самые основные: сигмоида и гиперболический тангенс.

Сигмоида:

Это самая распространенная функция активации, ее диапазон значений [0,1].

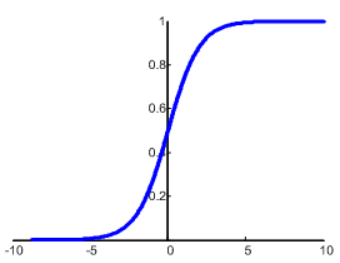


Рисунок 2 – сигмоида

Гиперболический тангенс:

Диапазон значений этой функции [-1,1].

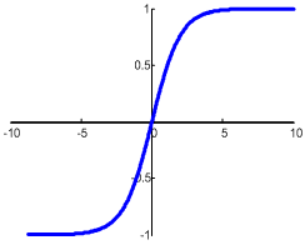
**

Рисунок 3 – гиперболический тангенс

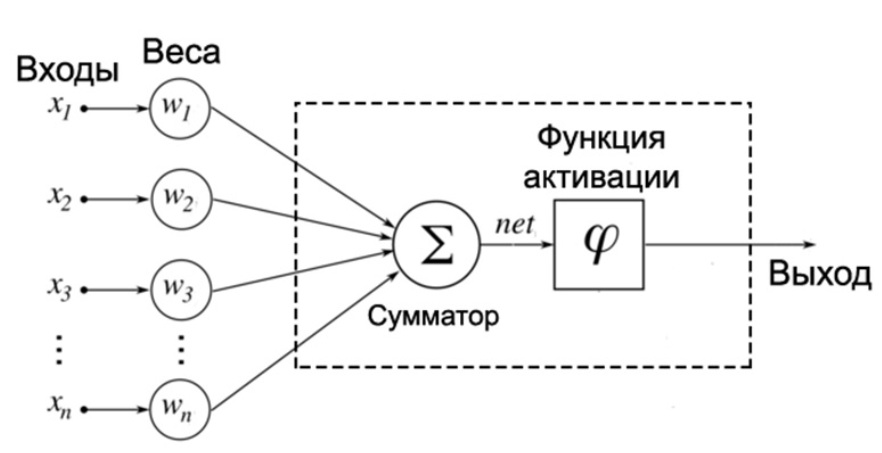


Рисунок 4 – схема работы нейрона

**2 USE**

Для многих задач NLP доступны ограниченные объемы обучающих данных. Это представляет собой проблему для методов глубокого обучения, ориентированных на данных. Учитывая высокую стоимость аннотирования данных контролируемого обучения, очень большие наборы обучения обычно недоступны для большинства исследовательских или отраслевых задач NLP. Многие модели решают эту проблему, неявно выполняя ограниченное обучение передаче с помощью предварительно обученных embedding’ов слов, таких как word2vec или GloVe. Однако недавняя работа продемонстрировала высокую производительность задач переноса с использованием предварительно обученных embedding’ов предложений. Universal Sentence Encoder кодирует текст в многомерные векторы, которые могут быть использованы для классификации текста, семантического сходства, кластеризации и других задач на естественном языке. Под USE подразумевается две модели для создания embedding’ов предложений, которые демонстрируют хороший перенос на ряд других задач NLP. Первая модель использует transformer архитектуру, а вторая – называется сеть глубокого усреднения (DAN). Используя embedding’и предложений из USE, мы можем получить хорошую производительность задач с небольшим количеством обучающих данных для конкретных задач.

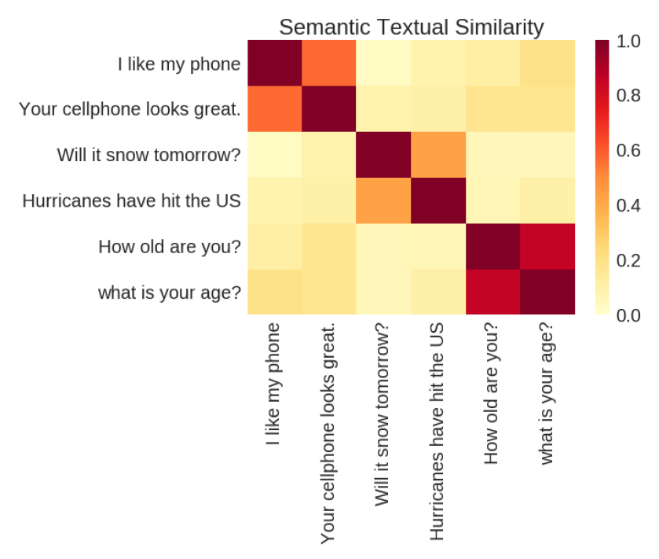


Рисунок 5 – Оценки сходства предложений с использованием embedding’ов из USE.

Два embedding’а данной модели имеют разные цели проектирования. Один из них, основанный на transformer архитектуре, нацелен на высокую точность за счет большей сложности модели и потребления ресурсов. DAN обеспечивает эффективный вывод с несколько сниженной точностью.

**2.1 Deep Averaging Networks (DAN)**

Многие существующие модели глубокого обучения для задач обработки естественного языка фокусируются на изучении композиционности их входных данных, что требует многих дорогостоящих вычислений. Мы представляем простую глубокую нейронную сеть, которая конкурирует с такими моделями и, в некоторых случаях, превосходит их по анализу настроений и ответам на вопросы, обучаясь при этом за меньшее время. Эта модель работает лучше, чем синтаксические модели для наборов данных с высокой синтаксической дисперсией. Также модель DNA допускает ошибки, аналогичные синтаксически ориентированным моделям, указывая, что для рассматриваемых нами задач нелинейное преобразование входных данных более важно, чем адаптация сети для включения порядка слов и синтаксиса.

Модели векторного пространства для обработки естественного языка (NLP) представляют слова с использованием низкоразмерных векторов, называемых embedding’и. Чтобы применить модели векторного пространства к предложениям или документам, необходимо сначала выбрать соответствующую композиционную функцию, которая представляет собой математический процесс объединения нескольких слов в один вектор. Композиционные функции делятся на два класса: неупорядоченные и синтаксические. Неупорядоченные функции обрабатывают входные тексты как пакеты вложений слов, в то время как синтаксические функции учитывают порядок слов и структуру предложения. Ранее опубликованные экспериментальные результаты показали, что синтаксические функции превосходят неупорядоченные функции во многих задачах. Однако существует компромисс: синтаксические функции требуют больше времени на обучение, чем неупорядоченные функции композиции, и являются непомерно дорогими в случае огромных наборов данных или ограниченных вычислительных ресурсов. Например, рекурсивная нейронная сеть вычисляет дорогостоящие тензорные произведения и нелинейности в каждом узле дерева синтаксического анализа, что ограничивает его меньшими наборами данных, которые могут быть надежно проанализированы. Мы вводим глубокую неупорядоченную модель, которая обеспечивает почти самую современную точность в различных задачах на уровне предложений и документов всего за несколько минут обучения на среднем портативном компьютере. Эта модель, сеть глубокого усреднения (DAN), работает в три простых шага:

1. возьмите среднее векторное значение вложений, связанных с входной последовательностью токенов;
2. передайте это среднее значение через один или несколько слоев прямой передачи;
3. выполните (линейную) классификацию представления конечного слоя.

DANS, в отличие от более сложных композиционных функций, могут быть эффективно обучены на данных с высокой синтаксической дисперсией. Качественный анализ изученных слоев показывает, что модель работает, увеличивая крошечные, но значимые различия в среднем векторе через несколько скрытых слоев, а подробный анализ ошибок показывает, что синтаксически осведомленные модели на самом деле допускают очень похожие ошибки, чем более наивные модели DAN.

**2.1.1 Neural Bag-of-Words Models (NBOW)**

Рассмотрим неупорядоченную нейронную модель, называемую «мешок слов» (NBOW). Для простоты рассмотрим классификацию текста, задача которой состоит в том, чтобы сопоставить входную последовательность токенов с одной из меток. Сначала мы применяем композиционную функцию к последовательности вложений слов для . Выход этой композиционной функции представляет собой вектор , который служит входным сигналом для функции логистической регрессии. В нашем экземпляре NBOW усредняет embedding’и слов

Подача в слой softmax индуцирует оцененные вероятности для каждой выходной метки

где функция softmax равна

– это матрица для набора данных с выходными метками, а b –смещение.

Мы обучаем модель NBOW так, чтобы минимизировать ошибку перекрестной энтропии, которая для одного обучающего экземпляра с меткой истинности равна

**2.1.2 Идея DAN**

Идея, лежащая в основе нейронных сетей с глубокой обратной связью (DAN), заключается в том, что каждый слой изучает более абстрактное представление входных данных, чем предыдущий. Мы можем применить эту концепцию к модели NBOW ожидая, что каждый слой будет все больше увеличивать небольшие, но значимые различия в среднем по embedding’у слов. Чтобы показать более наглядно, возьмем за предложение «I really loved Rosamund Pike’s performance in the movie Gone Girl» и создадим и , заменив «loved» на «liked», а затем снова на «despised». Векторные средние значения этих трех предложений почти идентичны, но средние значения, связанные с синонимичными предложениями и , немного больше похожи друг на друга, чем на среднее значение .

В уравнении (1) мы вычисляем , векторное представление для входного текста , путем усреднения векторов слов . Вместо того, чтобы напрямую передавать это представление в выходной слой, мы можем дополнительно преобразовать , добавив дополнительные слои перед применением softmax. Предположим, у нас есть слоев, . Мы вычисляем каждый слой

и передаем представление конечного слоя, , в слой softmax для прогнозирования. Эта модель, которую мы называем сетью глубокого усреднения (DAN), по-прежнему неупорядочена, но ее глубина позволяет ей улавливать тонкие изменения во входных данных лучше, чем стандартная модель NBOW. Кроме того, вычисление каждого слоя требует только одного умножения матрицы, поэтому сложность масштабируется с количеством слоев, а не с количеством узлов в дереве синтаксического анализа. На практике мы не находим существенной разницы между временем обучения DAN и временем обучения модели с неглубоким NBOW.

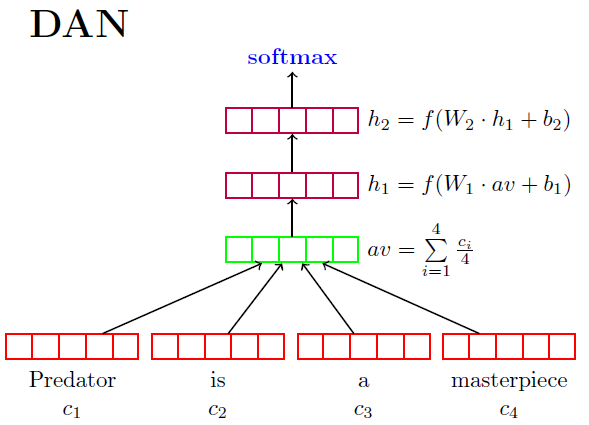


Рисунок 6 –двухслойная модель DAN, принимающая один и тот же вход. DAN вычисляет только два нелинейных слоя для каждого возможного ввода.

**2.1.3 Улучшение DAN**

«Отсеивание» регуляризует нейронные сети путем случайной установки скрытых и/или входных единиц на ноль с некоторой вероятностью . Учитывая нейронную сеть с единицами, отсев предотвращает переобучение, создавая ансамбль из различных сетей с общими параметрами, где каждая сеть состоит из некоторой комбинации отброшенных и не отброшенных единиц. Вместо отбрасывания единиц измерения естественным расширением модели DAN является случайное отбрасывание всех токенов слов embedding’ов слов из среднего значения вектора. Используя этот метод, который называется отсевом слов, наша сеть теоретически видит различных последовательностей токенов для каждого входа .

Мы помещаем вектор с независимыми испытаниями Бернулли, каждое из которых равно 1 с вероятностью . Embedding для токена в отбрасывается от среднего значения, если равен 0, что экспоненциально увеличивает количество уникальных примеров, которые сеть видит во время обучения. Это позволяет нам изменить уравнение (1):

В зависимости от выбора многие из «отброшенных» версий исходного обучающего экземпляра будут очень похожи друг на друга, но для более коротких входных данных это менее вероятно. Мы могли бы отбросить очень важный маркер, такой как «ужасно» в «крабовое рагу было особенно ужасно»; однако, поскольку количество типов слов, которые предсказывают выходные метки, невелико по сравнению с непрогнозируемыми (например, нейтральные слова в анализе настроений), мы всегда видим улучшения, используя этот метод.

**2.1.4 DAN выводы (можно в ЗАКЛЮЧЕНИЕ вставить)**

Сеть глубокого усреднения передает невзвешенное среднее значение векторов слов через несколько скрытых слоев перед классификацией. DAN конкурирует с более сложными нейронными сетями, которые явно моделируют семантическую и синтаксическую композиционность. Это еще больше усиливается за счет выпадения слов, регуляризатора, который уменьшает избыточность ввода. DANs получают близкую к современной точность как в анализе настроений на уровне предложений, так и на уровне документов, а также в задачах с ответами на вопросы с гораздо меньшим временем обучения, чем конкурирующие методы. Мы обнаруживаем, что и DANs, и синтаксические функции допускают аналогичные ошибки при синтаксически сложном вводе, что мотивирует исследования более мощных моделей композиционности.

**3 Механизм attention**

Для того, чтобы объяснить transformer архитектуру, нужно сначала познакомиться с механизмом attention, на котором основана transformer архитектура. Главная задача, ради который был придуман подход обработки текста attention – это машинный перевод. Для обучения нейросети этим методом, необходим набор данных, содержащий n-ое количество предложений, написанных на разных языках, но обозначающих одно и тоже.

Суть данного подхода заключается в том, что мы проходим по нашему предложению не только слева направо, но и в обратную сторону, при этом мы получаем новый вес значимости для каждого слова равный сумме весов слова, когда мы шли в одном направлении и в обратном

.

Вес означает вероятность того, что данное слово должно стоять на этом месте. Также мы высчитываем дополнительную информацию, которую мы получаем из предыдущих слов нашего предложения, по следующей формуле:

В результате наш decoder получает намного больше полезной информации, с помощью которой он и определяет вероятность нахождения слова на каждом месте.

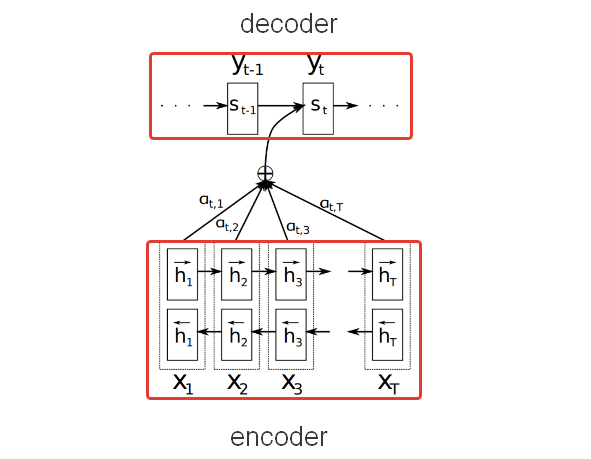


Рисунок 7 – схема взаимосвязи encoder’а и decoder’а

Этот подход также можно хорошо визуализировать, собственно, из-за визуализации и появилось название attention, так как алгоритм обращает внимание не только на «сухой» перевод предложения, но и на местоположение слов в нем.

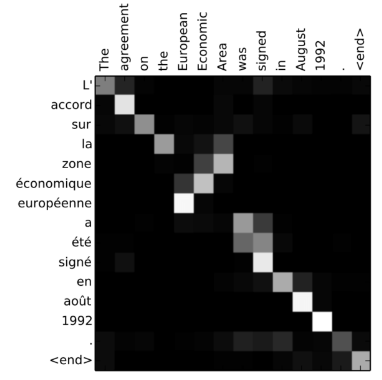


Рисунок 8 – пример работы механизма attention

**4 Transformer архитектура**

Данная модель состоит из двух модулей:

1. Encoder и multi-head attention

Данная часть архитектуры получает на вход слова и выдает их embedding’и.

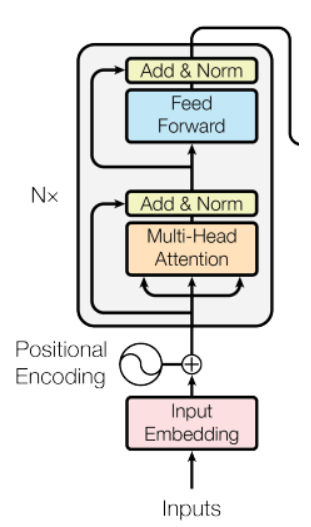


Рисунок 9 –Encoder и multi-head attention в Transformer архитектуре

Идея состоит в том, что каждое слово параллельно проходит через каждый слой. Но особым слоем является multi-head attention. Этот слой предоставляет возможность каждому входному вектору взаимодействовать с другими словами через attention механизм. Ему на вход поступают вектора Query, Key и Value.  Каждый из них преобразуется обучаемым линейным преобразованием, а потом вычисляется скалярное произведение Q со всеми K по очереди, затем прогоняется результат этих скалярных произведений через softmax, и с полученными весами все вектора V суммируются в единый вектор. Этот алгоритм работы с методом attention идентичен обычному методу attention, единственное отличие – таких attention'ов параллельно тренируется несколько. После чего результат всех этих параллельных attention'ов конкатенируется, еще раз прогоняется через обучаемое линейное преобразование и идет на выход. Но в целом, каждый такой модуль получает на вход вектор Query и набор векторов для Key и Value, и выдает один вектор того же размера, что и каждый из входов. Так как на выход такой блок выдает вектор того же размера, что и был на входе, то этот блок можно вставлять в сеть несколько раз, добавляя сети глубину. Также благодаря этому подходу у нейросети появилось новая способность для каждого слова – positional encoding, то есть его позиция в предложении. Благодаря этому в процессе обработки слова легко "обращать внимание" на соседние слова, если они важны.

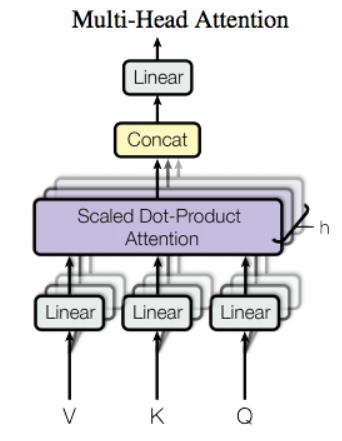


Рисунок 10 – multi-head attention в Transformer архитектуре

1. Decoder

Декодер тоже обрабатывает по слову за раз. Получая на вход предыдущее слово, он должен выдать текущее (на первой итерации получает специальный токен <start>). В декодере есть два разных способа использования Multi-head attention:

Первый – это возможность обратиться к векторам прошлых декодированных слов, также как и было в обычном процессе encoding, но можно обращаться не ко всем, а только к уже декодированным.

Второй – возможность обратиться к выходу encoder’а. B этом случае Query –это вектор входа в декодере, а пары Key/Value – это финальные embedding’и encoder’а, где опять же один и тот же вектор идет и как key, и как value, но линейные преобразования внутри attention module для них разные.  
Далее выполняются уже известные слои из привычных нам encoder’ов, а в конце сети стоит обычный softmax, который выдает вероятности слов. Сэмплирование из него – это текущее слово в предложении. Мы его даем на вход следующему запуску decoder’а и процесс повторяется, пока декодер не выдаст токен <end of sentence>. Во время процесса encoding’а каждый вектор взаимодействует со всеми другими. Во время процесса decoding’а каждое следующее слово взаимодействует с предыдущими и с векторами decoder’а.

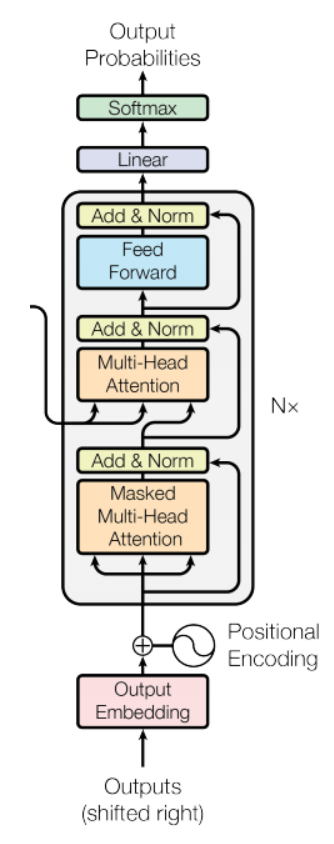


Рисунок 11 – decoder в Transformer архитектуре

Соединив обе части, описанные выше, мы получим transformer архитектуру.

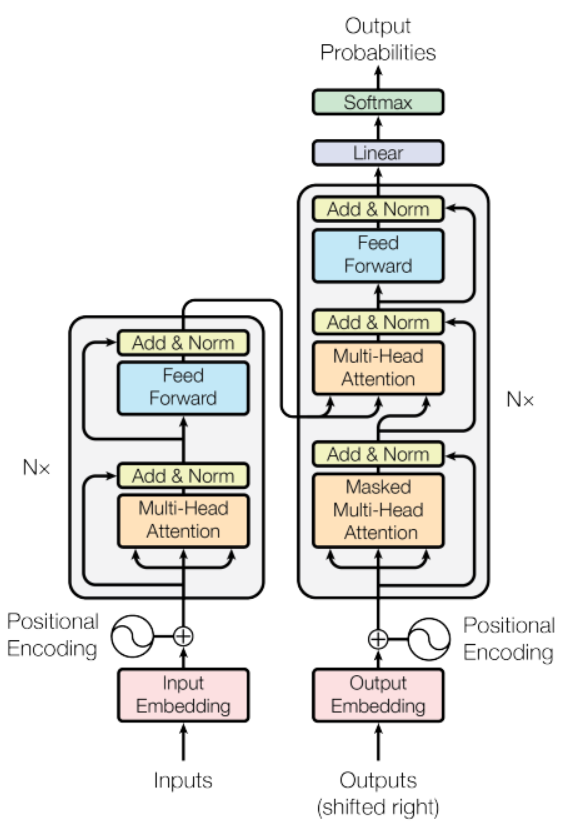


Рисунок 12 – Transformer архитектура

**5 BERT**

BERT (Bidirectional Encoder Representation Transformers), в переводе на русский «двунаправленная нейронная сеть-кодировщик» – модель представления языка, которая предназначена для предварительного обучения глубоких двунаправленных представлений на простых немаркированных текстах путем совмещения левого и правого контекстов во всех слоях. Это позволяет настраивать предварительно обученную модель BERT с помощью лишь одного дополнительного выходного слоя и получать наиболее актуальные результаты для широкого спектра задач таких как определение эмоциональной окраски (тональности) текста, вопросно-ответные системы, классификация текстов, построение выводов по тексту, создавать чат-ботов, автоматические переводчики и многое другое.

Большая часть применяемых моделей до BERT были однонаправленными, что сильно ограничивало выбор архитектур, которые можно использовать для предварительного обучения.

Тематическое моделирование – это технология статистического анализа текстов для автоматического выявления тематики в больших коллекциях документов определяет, к каким темам относится каждый документ, и какими словами описывается каждая тема. При этом не требуется ручной разметки текстов, а обучение ML-модели происходит без учителя. Тематическое моделирование допускает многоклассовую классификацию, т.е. чтобы документ относился одновременно к нескольким кластерам-темам, и позволяет отвечать на вопросы «о чём этот текст» или «какие общие темы имеет эта пара текстов». Тематическая модель формирует сжатое векторное представление текста, которое помогает классифицировать, рубрицировать, аннотировать, сегментировать тексты. В отличие от известных векторных представлений семейства x2vec (word2vec, paragraph2vec, graph2vec и т.д.), в тематических векторах каждая координата соответствует теме и имеет содержательную интерпретацию. Тематическая модель привязывает к каждой теме список ключевых слов или фраз, который описывает ее семантику.

В отличие от тематической модели, BERT работает по принципу векторного представления слов, основанном на контекстной близости, когда слова, встречающиеся в тексте рядом с одинаковыми словами (а, следовательно, имеющие схожий смысл), в векторном представлении будут иметь близкие координаты векторов. Полученные векторы могут быть использованы для обработки естественного языка и машинного обучения, в частности, для прогнозирования слов [8]. Таким образом, BERT выполняет также и предиктивные функции, в отличие от тематической ML-модели. Это свойство векторных NLP-технологий может использоваться в некоторых специфических задачах анализа текста, например, для определения авторства. Для каждого человека характерны некоторые специфические словосочетания, клише и прочие лексические конструкции, которые можно сгруппировать в устойчивые вектора и вычислить частоту их повторения в определенных текстах, определив авторскую принадлежность.

**5.1 Процесс обучения BERT**

BERT обучается в два этапа:

1. Предварительное обучение – модель обучается на немаркированных данных, выполняя различные задачи.

2. Точная настройка – модель загружается с предварительно обученными параметрами и обучается на помеченных данных из последующих задач.

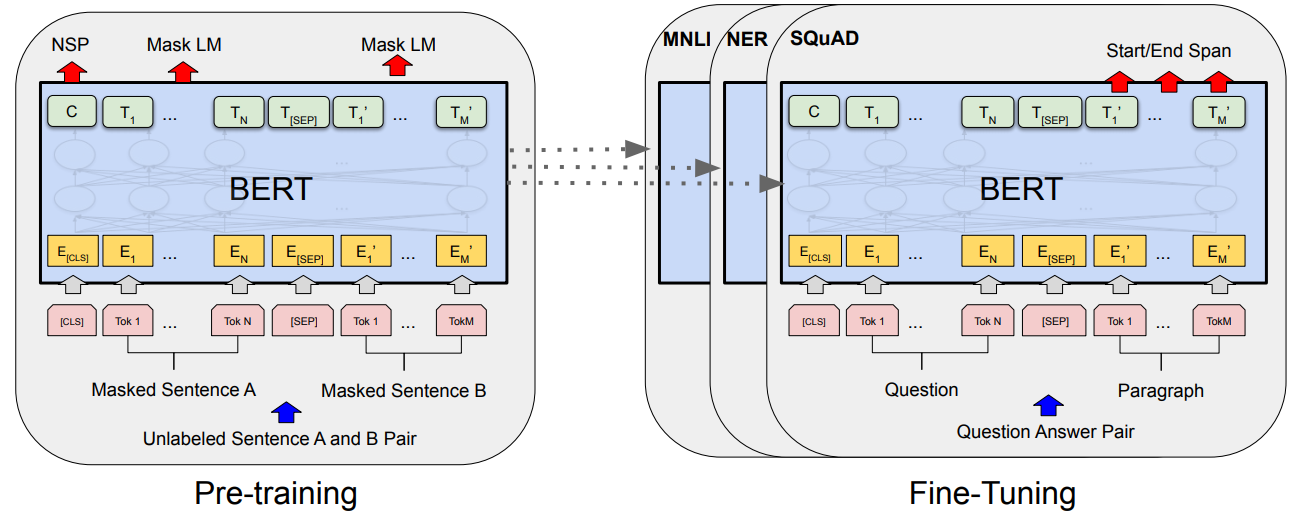


Рисунок 13 – схема общего предварительного обучения и точной настройки для BERT.

Помимо выходных слоев, одни и те же архитектуры используются как для предварительного обучения, так и для точной настройки. Одни и те же предварительно обученные параметры модели используются для инициализации моделей для различных последующих задач. Во время точной настройки все параметры настраиваются вручную. [CLS] – это специальный символ, добавляемый перед каждым входным примером, а [SEP] – специальный разделительный токен (например, разделяющий вопросы/ответы).

### 5.2 Задачи предварительного обучения

### Существует 2 задачи предварительного обучения: маскированная языковая модель (MLM) и предсказание следующего предложения (NSP). Чтобы натренировать BERT на предсказывание слов, на вход нейросети подаются фразы, где часть слов заменена на маску [MASK]. Например, получив на входе предложение «Я пришел в [MASK] и купил [MASK]», BERT на выходе должна показать слова «магазин» и «молоко». Это упрощенный пример с официальной страницы BERT, на более длинных предложениях разброс возможных вариантов становится меньше, а ответ нейросети однозначнее.

Второй тип задач основан на понимании отношений между предложениями. Это в какой-то степени отражает суть языкового моделирования. Чтобы обучить модель понимать связь между предложениями, ее предварительно обучают бинаризованной задаче прогнозирования следующего предложения. При подготовке примеров предложений  и  для предварительного обучения в 50% случаев  – это фактическое следующее предложение, которое следует за , а в 50% случаев  – случайное предложение из корпуса. Корпуса, используемые для предварительного обучения: BooksCorpus (800 млн слов) и English Wikipedia (2500 млн слов)

При подаче текста на вход BERT-модели сначала выполняется его токенизация – разбиение на более мелкие единицы (токены): абзацы делятся на предложения, предложения на слова и так далее. Входной текст разбивается на список токенов, доступных в словаре. Например, уже упомянутая выше модель BERT-Base использует словарь из 30522 слов. Если в словаре отсутствует слово, то оно постепенно разбивается на более мелкие части, которые уже находятся в словаре. Таким образом контекст нового слова будет комбинацией смыслов его частей.

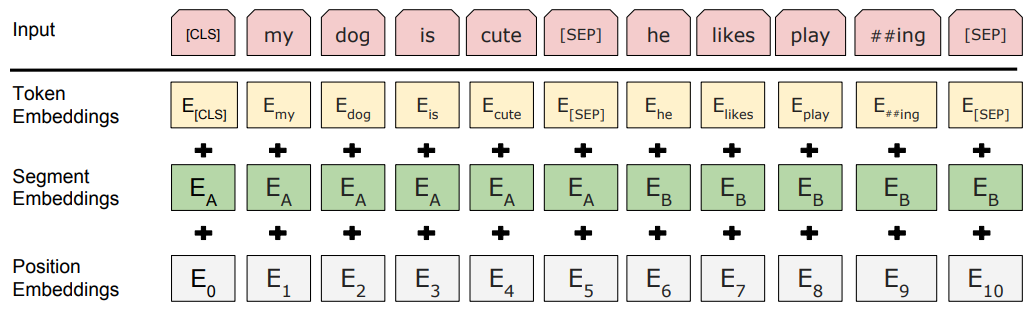


Рисунок 14 – токенизация в BERT.

Таким образом, BERT является autoencoder’ом, который скрывает и изменяет некоторые слова, пытаясь восстановить изначальную последовательность слов из контекста. Это приводит к следующим недостаткам работы модели:

1. каждое скрытое слово предсказывается в отдельности, из-за чего теряется информация о возможных связях между маскированными словами, например, «New York» является устойчивым сочетанием слов, при разделении которого на независимые части первоначальный смысл теряется полностью;
2. несоответствие между фазами тренировки и использования предобученной модели BERT: при тренировке применяются скрытые слова ([MASK]), а при использовании предобученной модели такие токены уже не подаются на ее вход.

Тем не менее, несмотря на вышеотмеченные проблемы, BERT называют последним достижением (state-of-the-art) в NLP-области.

**5.3 Точная настройка BERT**

В BERT точная настройка выполняется путем простой замены соответствующих входных и выходных данных в зависимости от того, включают ли последующие задачи один текст или текстовые пары. Для каждой конкретной задачи подключают входные и выходные данные соответственно и полностью вручную настраивают модель.

По сравнению с предварительной подготовкой, тонкая настройка относительно недорогая. Все результаты, приведенные в статье, могут быть воспроизведены не более чем за 1 час на одном облачном TPU или за несколько часов на GPU, начиная с одной и той же предварительно обученной модели.

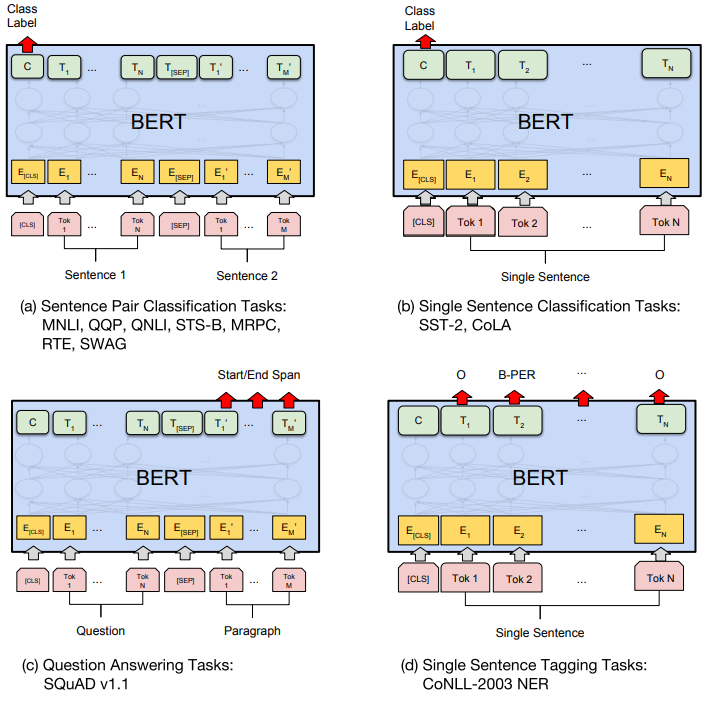


Рисунок 15 – примеры точечной настройки BERT для разных задач

**5.4 BERT выводы**

Тестовые исследования по оценке BERT, проведенные в 2019 году доказали эффективность этого DL-метода, достигнув наивысших отметок в классических испытаниях по пониманию естественного языка.

Однако, BERT – это не единственная DL-сеть, показывающая отличные результаты в решении NLP-задач, хотя, возможно и самая популярная. XLNet, преодобученная ML-модель многослойной transformer-архитектуры, показывает лучшие, по сравнению с BERT, показатели в тесте RACE (Reading Comprehension From Examinations). На рисунке 5 приведены результаты точности понимания содержимого текста на двух датасетах разного объема: среднего и большого. Сети BERT и XLNet имели 24 слоя и были аналогичны по размерам. В других задачах текстовой классификации DL-модель XLNet также показала лучшие результаты.

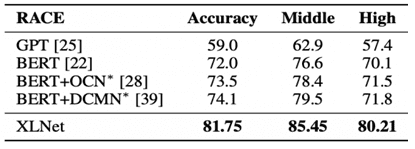


Рис. 5. Результаты нейросетевых моделей на задаче RACE

 (Думаю, что можно это убрать)

Такие отличные результаты XLNet обусловлены следующими факторами:

1. XLNet не маскирует слова в последовательности, благодаря чему отсутствует проблема несоответствия модели на предобучении и на тюнинге для отдельной задачи, что свойственно BERT;
2. XLNet не использует фиксированные прямонаправленный и обратнонаправленный порядки факторизации. Вместо этого XLNet максимизирует ожидаемый логарифм вероятности последовательности слов с учетом всех перестановок порядков слов. Благодаря шагу с перестановками, контекст для каждой позиции в последовательности может состоять из слов с правой и левой сторон. Таким образом, слово на каждой позиции в последовательности использует контекстную информацию со всех остальных позиций (bidirectional context).

В итоге, XLNet интегрирует в себе свойства авторегрессивных языковых моделей и autoencoder’ов, минуя недостатки обоих методов.

**ЗАКЛЮЧЕНИЕ**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Universal Sentence Encoder Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-C´espedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, Ray Kurzweil
2. Deep Unordered Composition Rivals Syntactic Methods for Text Classification Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, Hal Daum´e III
3. Attention Is All You Need Ashish Vaswani Noam Shazeer Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Łukasz Kaiser Illia Polosukhin
4. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
5. Universal Sentence Encoder  |  TensorFlow Hub
6. Deep Averaging network in Universal sentence encoder | by Aditya Kumar | tech-that-works | Medium
7. Google AI Blog: Transformer: A Novel Neural Network Architecture for Language Understanding
8. Transformer — новая архитектура нейросетей для работы с последовательностями / Хабр