

Дата поступления рукописи в редакцию: 18.06.2026 г.
Дата принятия рукописи в печать: 25.06.2026 г.

DOI: 10.24412/2782-3849-2026-6-602-612

СЕМЕНОВ Александр Александрович,

научный сотрудник ФКУ НПО «СТИС» МВД России,

e-mail: mail@law-books.ru

АВETИСЯН Карэн Рафаелович,

Старший преподаватель кафедры информационных технологий и организации расследования киберпреступлений Московская академия Следственного комитета Российской Федерации имени А.Я. Сухарева,

e-mail: Karen-Avetisyan-1989@bk.ru

МЕТОДЫ ПРЕДСТАВЛЕНИЯ ТЕКСТОВОЙ ИНФОРМАЦИИ И АРХИТЕКТУРА ТРАНСФОРМЕРОВ В ЗАДАЧАХ ОБРАБОТКИ ЯЗЫКА

Аннотация. Статья посвящена методам преобразования текстовой информации в числовую форму, включая токенизацию и эмбеддинги. Рассматриваются алгоритмы Word2Vec и BPE, а также подробно описывается механизм внимания и архитектура трансформеров. Отдельное внимание уделено большим языковым моделям, их обучению и ограничениям, включая проблему галлюцинаций.

Ключевые слова: обработка текста, токенизация, эмбеддинги, Word2Vec, BPE, механизм внимания, трансформер, attention, большие языковые модели, генерация текста, нейросети.

SEMENOV Aleksandr Aleksandrovich,

Chief Researcher, Federal State Budgetary Institution "STIS" of the Ministry of Internal Affairs of the Russian Federation

AVETISYAN Karen Rafaelovich,

Senior Lecturer, Department of Information Technology and Cybercrime Investigation Organization Moscow Academy of the Investigative Committee of the Russian Federation named after A. Ya. Sukharev

METHODS OF REPRESENTING TEXT INFORMATION AND TRANSFORMER ARCHITECTURE IN LANGUAGE PROCESSING PROBLEMS

Annotation. This article discusses methods for converting textual information into numerical form, including tokenization and embedding. The Word2Vec and BPE algorithms are discussed, along with a detailed description of the attention mechanism and Transformer architecture. Special attention is given to large language models, their training, and limitations, including the problem of hallucinations.

Key words: text processing, tokenization, embedding, Word2Vec, BPE, attention mechanism, Transformer, attention, large language models, text generation, neural networks.

Очевидная сложность применения нейросетей для анализа или генерации текста заключается в том, что текстовая информация сама по себе не поддается измерению. Очевидно, что если просто назначить каждому слову в словаре какое-то числовое значение, то никакая нейросеть не сможет выучиться на такого рода данных, так как близость двух номеров отнюдь не будет означать смысловую близость двух слов. Можно сказать, что текстовая

информация сама по себе обладает нулевой аналитичностью, поэтому построение нейросети, которая бы обрабатывала такую информацию в исходном виде, совершенно бессмысленно. Отсюда возникает необходимость сначала как-то превратить текстовую информацию в информацию, обладающую какой-то аналитичностью.

Наиболее примитивный подход заключается в том, чтобы вместо одного

текстового (или любого другого не числового) значения передавать в нейросеть несколько числовых значений, каждое из которых будет отвечать за конкретное возможное значение текстового параметра, равняться 1 в случае равенства ему переданного текстового значения и 0 в обратном случае. Например, если нейросеть обрабатывает текстовый параметр «автомобиль», который может принимать только значения «легковой», «грузовой» и «пассажирский транспорт», то вместо одного текстового параметра можно передавать нейросети три числа: (1,0,0) для легкового автомобиля, (0,1,0) для грузового и (0,0,1) для пассажирского транспорта. Этот подход кажется избыточным, однако он позволяет нейросети математическим образом «почувствовать» влияние каждого из параметров на результат. Однако для обработки текстов, в которых могут встречаться сотни тысяч разных слов, этот подход не очень хорош, так как неизбежно ведет к раздутию нейросети до огромных размеров.

В случае, если текстовые входные данные могут принимать множество разных значений, правильнее научиться ставить каждому значению некоторый числовой вектор ассоциаций. Например, если мы возьмем большое множество фильмов, то их названия не будут говорить совершенно ничего человеку, их не смотревшему. С другой стороны, если, например, каждому фильму сопоставить вектор из значений от 0 до 1, где каждое значение будет говорить о степени выраженности в фильме некоторого жанра или тематики, то эти вектора уже могут быть математически обработаны: с их помощью можно находить близкие по содержанию фильмы и, соответственно, давать рекомендации новых

фильмов на основе просмотренных.

Итак, для применения нейросетей к задачам с текстовой информацией нужно сначала поставить каждой единице текстовой информации некоторый эмбеддинг, именно эмбеддинги будут обрабатываться (или генерироваться) нейросетью. При этом нет однозначно лучшего способа определить эту самую единицу информации. Если определить в качестве единицы информации одну букву, то каждая такая единица не будет нести в себе никакого смысла, поскольку носителями смысла все же являются слова. Можно определить в качестве единицы информации слово, но у одного слова может быть множество форм. Казалось бы, можно предварительно привести по словарю все слова к своим наиболее простым формам (глаголы к неопределенной форме, прилагательные к мужскому роду и т. д.), чтобы сократить объем работы.

Однако такой прием может работать по-разному в разных языках. Например, в английском языке форм слов немного и порядок членов предложения в большинстве случаев определен, поэтому такое упрощение предложений может показаться относительно безвредным; напротив, в русском языке масса форм слов и произвольный порядок членов предложения, поэтому приведение всех слов в предложении к простейшим формам легче приведет к потере содержания. Человечество пробовало разные подходы. В настоящее время наиболее распространенным является смешанный подход, при котором единицами представления информации могут быть наравне как отдельные буквы или символы, так и слоги, слова или даже словосочетания.

Токен – единица представления текстовой информации.

Токенизация – процесс разбиения текста на токены.

Корпус текстов – синоним выражения «набор текстов».

Byte-pair encoding/BPE – широко применяемый алгоритм токенизации, где заранее определяется число токенов, на которое будет разбит текст, а выбор букв, символов, слогов или слов, которым будут назначены свои собственные токены, делается на основе того, насколько часто соответствующие буквы, символы, слоги или слова встречаются в корпусе текстов, на котором выполняется алгоритм. При необходимости алгоритмом гарантируется, что все содержимое корпуса можно будет представить через выбранные алгоритмом токены.

Word2vec – алгоритм назначения эмбеддингов токенам, в котором на основе анализа того, какие токены с какими другими токенами имеют тенденцию встречаться рядом в корпусе текстов, выделяют ряд числовых признаков Word2vec.

Отметим, что, помимо букв, символов, слогов, слов и словосочетаний вводятся еще специальные «технические» токены. Например, токены начала и конца предложения. После того, как токены определены, нужно назначить им значения эмбеддингов. Этот процесс в каком-то смысле напоминает начальную инициализацию весов. Можно применить специальные приемы для выбора значений эмбеддингов (примером алгоритма является word2vec), а можно задать их случайно.

$$e \leftarrow e - a \cdot L'_e$$

Рисунок 1. Формула эмбеддинга.

Теперь представим себе рекуррентную нейросеть, классифицирующую текст. Слов в языке сотни тысяч, для их передачи понадобятся большие эмбеддинги, например, из 1000 числовых значений. Это число определяет характерный размер матриц, возникающих в рекуррентных слоях, которых наверняка несколько. Обработка одного текста потребует вызова нейросети столько раз, на сколько токенов разбит текст. Для небольшого абзаца текста это значение может быть равно, например, нескольким сотням. Рекуррентная нейросеть, конечно, предоставляет определенные удобства при обработке текста по сравнению с полносвязной

На практике эмбеддинги чаще задают случайным образом и интерпретируют их как веса нейросети, соответственно, позволяют им изменяться в процессе обучения, для чего во время градиентного спуска доводят процесс обратного распространения ошибки до входных значений, т. е. вычисляют производные лосса L по значениям эмбеддинга e и вычитают соответствующее значение на шаге градиентного спуска, как будто это тоже веса (рисунок 1).

нейросетью, ведь последней пришлось бы передать на вход один большой вектор из сразу всех эмбеддингов текста, к тому же пришлось бы длину текста зафиксировать. Однако она представляет собой противоположную крайность, так как предполагает, что очередной токен зависит в первую очередь от предыдущего. С алгоритмической точки зрения это не безупречная конструкция, к тому же обучение такой нейросети неудобно распараллеливать. В попытке найти для работы с текстами какое-то более совершенное решение, чем рекуррентные нейросети, попробовали применить так называемый механизм внимания.

Механизм внимания/attention (в контексте нейросетей) – идея, заключающаяся в том, чтобы позволить нейросети каким-то математическим образом определять из множества обрабатываемых внутри слоя данных наиболее содержательный фрагмент. Эта идея может реализовываться в самых разнообразных формах.

Можно сказать, что в полносвязном и сверточном слоях механизм внимания полностью отсутствует, так как в них все данные входного тензора обрабатываются одинаково. В рекуррентных слоях ворота (рисунок 4) можно в каком-то смысле интерпретировать как механизм внимания, ведь они по задумке должны какие-то данные обнулять, а какие-то выводить на первый план. В то же время про рекуррентные нейросети

в целом можно сказать, что в них реализован своего рода механизм внимания, указывающий на результаты обработки последнего входного значения.

Более совершенный механизм внимания смогли разработать, отталкиваясь от идеи поиска нужной информации по базе данных. Представим себе простую систему поиска, которая по запросу q обращается в базу данных и ищет ключ k ,

равный запросу q , и выдает значение v , которое в базе данных соответствует данному ключу k (обозначения Q , K и V являются типовыми и часто встречаются, они происходят от слов «query», «key» и «value» и применяются для того, чтобы условно выделить в механизме внимания что (K) ищется для чего (Q), и что будет результатом (V). Например, база данных может содержать названия городов мира в качестве ключей и названия стран, где эти города находятся, в качестве значений. Следовательно, если мы обратимся к базе данных с запросом в виде названия города, она найдет его в базе и выдаст соответствующее название страны. Такая схема для каждого запроса q выдает соответствующее ему значение v (рисунок 2, слева), но не сможет дать значение, если запросу не равен ни один ключ k . Представим себе теперь, что вместо непосредственно текстовой информации мы храним в базе в качестве ключей и значений эмбеддинги слов, равно как и запрос передаем в виде эмбеддинга. Это не мешает использовать прежнюю схему поиска, но позволяет ее

расширить. Представим себе, что некоторая строительная компания хранит информацию о построенных объектах, где в качестве ключа хранится вектор характеристик строительного объекта, а в качестве соответствующего ключу значения – вектор потраченных при строительстве объекта материалов. С помощью некоторой функции f можно определить, насколько запрос q похож на каждый ключ k , и вычислить некоторый весовой коэффициент, полученный коэффициент умножить на вектор значений v , все полученные таким образом вектора сложить и выдать сумму в качестве результата поиска (рисунок 2, справа).

В примере со строительной компанией такой механизм позволит по информации о свойствах объекта оценивать приблизительно объем материалов, нужный для его строительства. При этом ни один из ключей в базе данных может быть не равен исходному запросу; получаемый результат не будет равен никакому из значений в базе, а будет представлять собой некоторую их комбинацию.

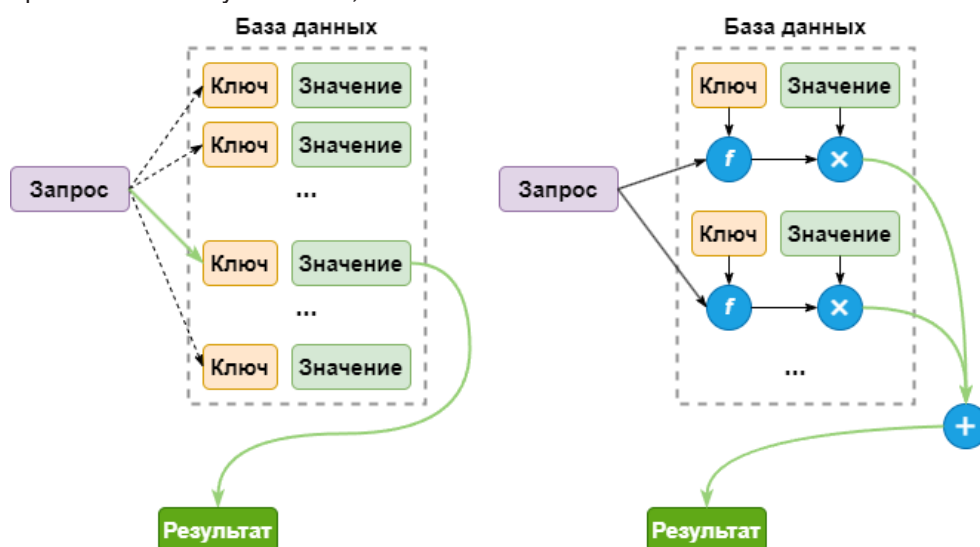


Рисунок 2. Схема для каждого запроса q .

Обратим внимание, что в предложенной схеме векторы запроса и ключей должны обладать одинаковой размерностью d_K , а размерность векторов значений d_V может от них

$$A(q) = \sum_i \text{softmax}_i \left(\frac{q \cdot k_i}{\sqrt{d_K}} \right) \cdot v_i$$

Рисунок 3. Формула для вычисления результата A .

В приведенной формуле используется операция софтмакс, примененная к вектору скалярных произведений вектора q на каждый из векторов k_i для того, чтобы сумма множителей, стоящих перед векторами v_i , была равна единице.

отличаться. Формула для вычисления результата A может быть следующей (обратим внимание, что q , k_i и v_i – это целые вектора) (рисунок 3).

Скалярное произведение выбрано в качестве меры похожести векторов равно по той же причине. Деление скалярного произведения на корень из длины вектора добавлено авторами оригинального подхода без необходимости.

Описанный формулой на рисунок 3 механизм внимания авторами был назван «scaled dot-product attention», т. е. «механизм внимания на основе скалярного произведения с масштабированием».

Казалось бы, чтобы применить описанный механизм внимания нужно определить, что с чем будет сравниваться и зачем, т. е. какое значение будет придаваться входным и выходным значениям в формуле на рисунке 3. Вспомним теперь, что в машинном обучении не пытаются задать конкретные значения всем параметрам алгоритма, а позволяют машине найти для них значения. Также и интерпретацию механизма внимания при его применении в составе нейросети можно оставить открытым.

Для этого входные векторы перед применением механизма формулы на рисунке 3 подвергаются умножению на соответствующие

им матрицы весов, результат вычислений также умножается на матрицу весов. В процессе машинного обучения для этих весов будут определены какие-то оптимальные значения, которые будут нести в себе какую-то непотную человеку трактовку происходящих внутри механизма внимания вычислений. Авторы этого подхода также обнаружили, что лучше иметь несколько параллельных наборов таких матриц, что можно примерно понимать, как наличие одновременно нескольких параллельных механизмов внимания, которым придан разный смысл (рисунок 4).

На рисунке 4 кружком с буквой «А» обозначается применение механизма внимания, описанного на рисунке 3, а под Q, K и V понимаются матрицы «запросов», «ключей» и «значений» составленные из некоторых векторов.

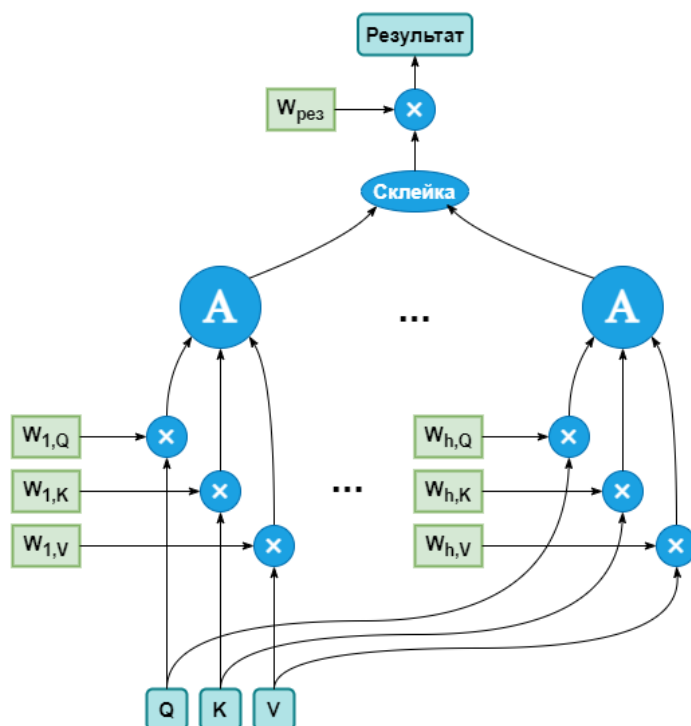


Рисунок 4. Наличие одновременно нескольких параллельных механизмов внимания.

Длины этих векторов уже необязательно должны быть одинаковы, достаточно, чтобы были одинаковыми длины векторов, получающихся после матриц весов $W_{i,Q}$ и $W_{i,K}$. Результат применения операции A для одного вектора q является вектором, соответственно, если применить эту операцию к матрице

Q из нескольких векторов, то результатом будет некоторая матрица. Описанный блок назван авторами «multi-head attention», т. е. «многоголовый механизм внимания», под числом «голов» при этом понимается значение h, т. е. число параллельно задействованных механизмов внимания.

Трансформер – архитектура нейросети, построенная на базе механизма внимания. В настоящее время наиболее продвинутые нейросети, применяемые для анализа или генерации текстов, имеют эту архитектуру.

Первая нейросеть с этой архитектурой [15] была предназначена для перевода текстов с одного языка на другой, разберем ее устройство подробнее (далее, пока не оговорено обратное, рисунок 5).

Нейросеть состоит из двух частей – энкодера (слева) и декодера (по середине). В контексте трансформеров использование терминов «энкодер» и «декодер» несколько искажает смысл и используется по аналогии с предыдущими поколениями нейросетей. Более правильно понимать так: задача энкодерной части – проанализировать входные данные, а

задача декодерной – сгенерировать выходные. Энкодер может обрабатывать весь исходный текст (на языке, с которого переводят), а декодер сделал таким образом, чтобы он генерировал выходной текст (на языке, в который нужно осуществить перевод) по одному токenu за раз на основе информации от энкодера и всех ранее сгенерированных им самим выходных токенов. Таким образом, энкодер полностью лишен рекурсии, а в декодере она частично присутствует в том смысле, что каждый выходной токен генерируется на основе всех предыдущих.

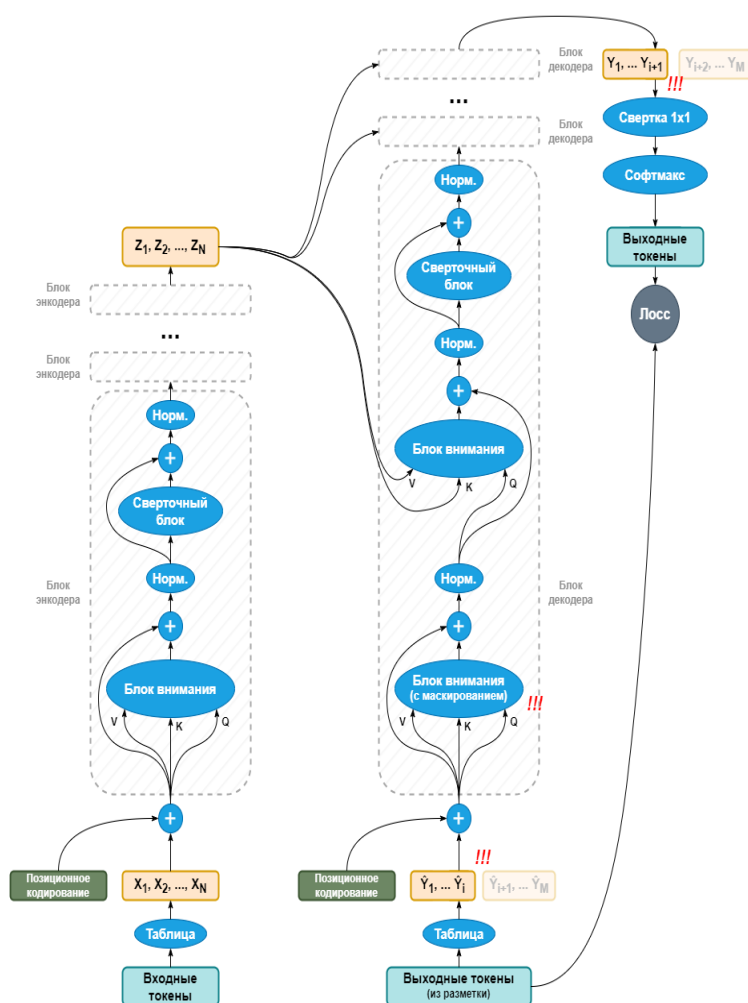


Рисунок 5. Устройство нейросети для перевода текстов с одного языка на другой.

Сначала входной текст в соответствии с выбранным алгоритмом токенизации разбивается на токены, каждому из которых по таблице ставится в соответствие эмбеддинг. Таким образом, получается последовательность из N векторов X_i некоторой длины d , объединяющими в матрицу. В этой матрице номер строки соответствует номеру токена в тексте. Очевидно, что позиция токена в тексте имеет значение для восприятия смысла предложения и должна

учитываться нейросетью. Чтобы нейросеть могла отличать друг от друга одинаковые слова, стоящие в разных местах в предложении, используют следующий прием: к каждому эмбеддингу прибавляют некоторый вектор, вычисляемый по заданному алгоритму, который избыточно кодирует внутри себя информацию о номере токена в предложении. Данный прием кажется очень контринтуитивным, поскольку фактически он засоряет каждое значение

эмбеддинга лишней информацией вместо того, чтобы просто прибавить к эмбеддингу одно дополнительное значение номера токена. Однако следует помнить, что нейросеть – не человек, и если некоторая лишняя информация добавляется понятным обратимым алгоритмом, то нейросети с полученной информацией иметь дело проще, чем человеку может казаться.

Полученная матрица передается в энкодер. Энкодер состоит из нескольких одинаковых блоков (имеющих, разумеется, свои наборы весов). В начале блока расположен блок внимания, реализующий схему, проиллюстрированную на рисунке 4. Может показаться странным, что входная матрица блока энкодера используется в качестве каждой из трех матриц «запросов», «ключей» и «значений», однако следует помнить, что в начале блока внимания каждая из матриц домножается на свои собственные весовые матрицы. Результат работы блока внимания складывается с его выходным значением для того же, для чего в нейросети ResNet, - для улучшения способности градиентного спуска «проталкивать» изменение весов к первым слоям. После этого применяется нормализация, приводящая все выходные значения в диапазон от -1 до 1 . Полученный результат передается в сверточный блок, т. е. в последовательность из нескольких сверточных слоев, разделенных нелинейностью типа ReLU. Результат вычислений сверточного блока складывается с входной матрицей, далее опять применяется нормализация. На этом вычисления энкодерного блока заканчиваются, при этом размеры входной и выходной матрицы энкодерного блока совпадают.

Важно отметить, что при описании нейросетей, отличных от сверточных, существует определенная каша, разные люди называют одни и те же слои кто «полносвязными» или «линейными», а кто «сверточными» в зависимости от того, как удобнее их вообразить. Автор старается называть полносвязным слой, который представляет собой умножение весовой матрицы на вектор, а сверточным слой, в котором входные данные представляют собой как минимум двумерный массив, а вычисления слоя представляют собой, соответственно, перемножение двух матриц. В оригинальной статье сверточный блок называется *feedforward network*.

Если блок внимания выполняет сопоставление эмбеддингов разных токенов друг с другом и как бы обогащает информацию о каждом токене информацией о его взаимосвязях с другими токенами, то сверточный блок обрабатывает каждый эмбеддинг одинаково, но отдельно от других. Считается, что именно

в сверточных блоках выполняется основная содержательная обработка информации, в связи с чем размеры промежуточных карт признаков, возникающих внутри сверточного блока, устанавливают побольше, в результате чего именно веса слоев сверточных блоков составляют основной объем весов всей нейросети.

Результат вычислений последнего энкодерного блока передается в каждый декодерный блок. Декодерный блок обрабатывает последовательность не входных токенов, а выходных (во время обучения – из разметки, во время инференса – ранее сгенерированных). Так же, как с входными токенами, выходным токенам ставятся в соответствии некоторые эмбеддинги $\hat{Y}_j$ (но уже другие, так как выходные токены вообще взяты из другого языка), к которым добавляется позиционное кодирование (опять же другое, в общем случае число слов N во входном тексте и число слов M в выходном тексте не совпадают). Далее матрица из эмбеддингов попадает в декодер, состоящий из нескольких декодерных блоков. Декодерный блок выполняет те же действия, что и энкодерный, но, чтобы использовать информацию от энкодера, в него добавлен еще один блок внимания. В этом блоке внимания, в отличие от первого, в качестве матрицы «запросов» используется полученная от предыдущего блока матрица, а в качестве матриц «ключей» и «запросов» используется матрица значений, полученная энкодером. Таким образом, информация о взаимосвязях между токенами во входном тексте попадает в промежуточную сущность, из которой декодер должен сделать выходной текст.

Вспомним, что декодеру ставится задача сгенерировать один следующий токен выходного текста на основе всех ранее сгенерированных токенов. Это означает, что во время инференса число эмбеддингов во входной матрице для декодера будет определяться числом уже сгенерированных токенов, а сам инференс будет запускаться, как у рекуррентной нейросети, несколько раз. В результате j -го инференса нейросети будет получаться некоторый эмбеддинг Y_{j+1} , которому нужно сопоставить токен. Если использовать таблицу, сопоставляющую токены выходного текста с эмбеддингами, в ней в общем случае не будет эмбеддинга, в точности равного Y_{j+1} , поскольку нейросеть выдает результат с некоторыми погрешностями, т. е. понадобится еще найти в этой таблице эмбеддинг, больше всего похожий на Y_{j+1} . На практике так не делают, вместо этого в нейросети после декодера добавляют сверточный слой (матричное умножение), весовая матрица которого после умножения на вектор-эмбеддинг дает вектор вероятностей того, что

эмбеддинг относится к токenu, для каждого токена в используемом алгоритме токенизации. Решается задача классификации полученного эмбеддинга, при этом каждому токenu соответствует свой класс, который определяется после применения софтмакса. Соответственно, в лоссе оценивается, был ли очередной токен определен правильно или нет.

Обратим внимание на то, что в описанной архитектуре преобладают операции умножения матричного умножения, это теоретически позволяет эффективно использовать TPU для выполнения вычислений этих нейросетей.

Чтобы распараллелить процесс обучения (это на самом деле и было целью разработчиков оригинальной нейросети), во время обучения нейросети на вход декодеру подается сразу вся выходная последовательность токенов, на выходе также ожидается генерация этой же самой последовательности. На первый взгляд это кажется нелепым, но здесь подразумевается, что нейросеть параллельно учится предсказывать

второй токен по первому, третий по первым двум, четвертый по первым трем и т. д. Чтобы реализовать эту схему, в первом блоке внимания в каждом блоке декодера маскируется (блокируется) учет взаимосвязи токена с токенами, идущими после него (но не наоборот). Модифицированный таким образом механизм внимания называется *masked multi-head attention*.

На самом деле именно в том, используется маскирование или нет, заключается разница между энкодером и декодером. Для решения задач, отличных от перевода текста с языка на язык, может использоваться только энкодерная или только декодерная часть. Например, для классификации текстов можно использовать только энкодер, для обработки выходных значений которого приделаны полносвязный слой с софтмаксом. Для генерации текстов используют, наоборот, только декодерную часть (при этом из декодерных блоков пропадают механизмы внимания, использующие данные от энкодера).

Большая языковая модель / large language model – общее название, которым называют нейросети, генерирующие тексты. В настоящее время такие модели, как правило, представляют собой трансформер, состоящий только из декодера.

К большим языковым моделям относится семейство популярных нейросетей для чат-ботов Chat GPT. Такие нейросети в общем случае учатся в две стадии. На наиболее трудоемкой первой стадии (чаще всего ее называют *pre-train*) нейросеть учит предсказывать следующее слово в тексте (получается, текст автоматически содержит в себе разметку).

При этом используют огромный корпус текстов, чаще всего из всего подряд, что можно найти в интернете. После этого нейросеть становится способной продолжить любой текст человекоподобным образом, но бесцельно. Чтобы от нейросети могла быть какая-то польза, на второй стадии ее целенаправленно учат быть стороной диалога. Общение с нейросетью реализуется так: человек начинает диалог, его текст преобразуется в последовательность токенов, на их основе нейросеть генерирует еще некоторое количество токенов, при этом токены из пользовательского текста и токены, сгенерированные нейросетью, хранятся и обрабатываются в виде единой последовательности. Отсюда становится интуитивно понятно ключевое отличие трансформера-декодера от трансформера-

энкодера: в живом диалоге нейросеть не знает заранее, как продолжит диалог человек, соответственно, она не может генерировать свой ответ, опираясь на токены, которые возникнут в диалоге позже, поэтому должна опираться только на предыдущие токены.

Обратим внимание, что на практике число токенов в последовательности, сформированной в результате общения нейросети с человеком, все же ограничивают, иначе возникает необходимость раздуть до бесконечности некоторые весовые матрицы. После того, как диалог преодолевает этот предел, добавление новых токенов становится возможным только за счет забывания такого же числа токенов в начале последовательности. Поскольку нейросеть обучена вести диалог в динамике, в некоторых случаях она может продолжать успешно выполнять свою задачу, в других же случаях, когда начало диалога содержало какую-нибудь ключевую информацию, осмысленность генерируемого текста может резко уменьшиться. Переполнение контекста, который помнит языковая модель, – наиболее простой способ ее «сломать».

Галлюцинирование нейросети – явление, при котором нейросеть выдает не просто неточный, а очевидно бессмысленный результат. В контексте больших языковых моделей галлюцинирование может наблюдаться в виде генерации несуществующих слов или заведомо ложных фактов, как будто она «сломана».

Наблюдение за галлюцинированием нейросети – наиболее простой способ почувствовать разницу между интеллектуальными способностями человека и искусственным интеллектом нейросети. Следует помнить, что даже самая продвинутая нейросеть не умеет думать, она только подражает этому процессу, а ее знания ограничены той массой информации, которая получилась в результате сжатия обучающего датасета.

Для нейросетей, обрабатывающих тексты, нашли способ бороться с ограниченностью ее знаний путем добавления специальных технических токенов, которые инициируют определенные действия программного обеспечения, реализующего вычисления нейронной сети. Так, например, учат нейросеть в особых случаях генерировать специальный токен, указывающий на то, что последнее сгенерированное предложение должно быть использовано в качестве поискового запроса в интернете. При появлении такого токена программное обеспечение осуществляет запрос в интернет, после чего нужный результат передает обратно нейросети, опять-таки, в виде токенов. Таким способом можно организовывать взаимодействие нейросети с множеством различных инструментов, в том числе, например, позволять ей вызывать другие нейросети.

Взаимосвязи слов в тексте являются наиболее естественным для человека способом выражения мысли. Даже творческий человек с преимущественно образным мышлением не может совсем обойтись без размышлений в форме речи как бы вслух у себя в голове. Поэтому способность нейросетей обрабатывать и генерировать текст в большей степени приближает их к интеллектуальным способностям человека, чем обработка или генерация информации любого другого вида. Этим объясняется повышенный интерес к такого рода нейросетям. На их основе можно создавать более сложные нейросетевые комбинации, обрабатывающие или генерирующие информацию другого вида с лучшим качеством, нежели предыдущие нейросети. Например, можно обучить сверточную нейросеть находить на изображении объекты и создавать их эмбеддинги, после чего эти эмбеддинги будут обрабатываться трансформером для создания текстового описания картинки. Такое текстовое описание,

В свою очередь, может само быть подвергнуто анализу, в том числе, например, с учетом текстовых подсказок от человека, и т. д. В частности, это теоретически может позволить отказаться от огромных датасетов для обработки изображений и использовать вместо них небольшие датасеты с текстовыми пояснениями.

Большие языковые модели представляют особый интерес по той причине, что они позволяют человеку взаимодействовать с нейросетями не просто через процесс машинного обучения, а с помощью естественного для него способа общения.

В погоне за большими интеллектуальными способностями разработчики пробуют все более и более сложные большие языковые модели. Некоторые современные трансформеры содержат триллионы весов (т. е. в миллионы раз больше, чем у некоторых сверточных нейросетей), их обучение может занимать недели на суперкомпьютерах, состоящих из десятков и сотен видеокарт. Даже обученная такая нейросеть может поместиться не на каждом компьютере, в связи с чем наравне с наиболее продвинутыми

вариантами нейросетей создают их меньшие аналоги, содержащие меньшее количество весов и, соответственно, обладающие более скромными характеристиками.

Список литературы:

[1] Chen Zhang, Joohee Kim Video object detection with two-path convolutional LSTM pyramid. 2016. – URL: https://www.researchgate.net/publication/343702579_Video_Object_Detection_With_Two-Path_Convolutional_LSTM_Pyramid/

fulltext/5f3b2658928 51cd302013482/Video-Object-Detection-With-Two-Path-Convolutional-LSTM-Pyramid.pdf/ (дата обращения: 16.01.2026).

[2] Hanson A., Koutilya P., Sanjukt Krishnagopal, Larry Davis Bidirectional Convolutional LSTM for the Detection of Violence in Videos. 2018. – URL: https://openaccess.thecvf.com/content_eccv_2018_workshops/w10/html/Hanson_Bidirectional_Convolutional_LSTM_or_the_Detection_of_Violence_in_Videos_ECCVW_2018_paper.html (дата обращения: 15.01.2026).

[3] Howard A.G., Menglong Zhu, Bo Chen, Kalenichenko D. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. – URL: <https://arxiv.org/pdf/1704.04861/> (дата обращения: 11.01.2026).

[4] Iandola F.N., Han S., Moskewicz M.W., Ashraf K., Dally W.J.,

Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size Recognition. 2016. Published as a conference paper at ICLR. 2015. – URL: https://www.researchgate.net/publication/301878495_SqueezeNet_AlexNet-level_accuracy_with_50x_fewer_parameters_and_05MB_model_size (дата обращения: 10.01.2026).

[5] Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015. –

URL: <https://arxiv.org/pdf/1502.03167/> (дата обращения: 13.01.2026).

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition. 2015. –

URL: <https://arxiv.org/pdf/1512.03385> (дата обращения: 11.01.2026).

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Spatial pyramid pooling in deep convolutional networks for visual recognition. 2015. – URL: <https://arxiv.org/pdf/1406.4729/> (дата обращения: 11.01.2026).

[8] Krizhevsky I., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks. 2012. – URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (дата обращения: 12.01.2026).

[9] Lavin A., Scott Gray Fast Algorithms for Convolutional Neural Networks. 2015. – URL: <https://arxiv.org/abs/1509.09308> (дата обращения: 11.01.2026).

[10] Mikolov T., Kai Chen, Corrado G., Jeffrey Dean Efficient Estimation of Word Representations in Vector Space. 2013. –

URL: <https://arxiv.org/abs/1301.3781> (дата обращения: 15.01.2026).

[11] Norman P. Jouppi, Cliff Young, Nishant Patil and others In-Datcenter Performance Analysis

of a Tensor Processing Unit. 2017. –

URL: <https://arxiv.org/pdf/1704.04760> (дата обращения: 11.01.2026).

[12] Ronneberger O., Fischer Ph., Brox Th. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. –

URL: <https://arxiv.org/pdf/1505.04597/> (дата обращения: 13.01.2026).

[13] Szegedy C., Wei Liu, Chapel Hill, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich Going Deeper with Convolutions. 2014. –

URL: <https://arxiv.org/pdf/1409.4842> (дата обращения: 12.01.2026)

[14] Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2015. Published as a conference paper at ICLR 2015. – URL: <https://arxiv.org/pdf/1409.1556> (дата обращения: 10.01.2026).

[15] Wei Liu, Anguelov D., Erhan D., Szegedy C., Reed S., Cheng-Yang Fu, Berg A.C. SSD: Single Shot MultiBox Detector. 2016. –

URL: <https://arxiv.org/pdf/1512.02325> (дата обращения: 11.01.2026).

[16] Официальный сайт ONNX. – URL: <https://onnx.ai/>.

[17] Основы технологий искусственного интеллекта: учебное пособие/ А.А. Семенов, В.В. Гончар, А.С. Эрднеев. – Московский университет МВД России имени В.Я. Кикотя, 2025. – 190с.

[18] Финансовый мониторинг: учебное пособие / В.И. Глотов, А.У. Альбеков, О.Н. Тисен и др. – КноРус, 2022. – 198 с.

[19] Тисен О.Н. Противодействие современным приемам и способам вербовки в международные террористические организации // Российская юстиция. 2018, №9. С. 51-54.

[20] Гончар В.В. О важности формирования единого понятийного аппарата необходимого для расследования преступлений в сфере компьютерной информации// Вестник Экономической безопасности. 2018. №1. С. 225-230.

[21] Орлова А.А., Гончар В.В. Особенности расследования преступлений, совершаемых в сфере информационных технологий // Безопасность бизнеса. 2021. № 4. С. 25-29.

References:

[1] Chen Zhang, Joohee Kim Video object detection with two-path convolutional LSTM pyramid. 2016. – URL: https://www.researchgate.net/publication/343702579_Video_Object_Detection_With_Two-Path_Convolutional_LSTM_Pyramid/fulltext/5f3b2658928 51cd302013482/Video-Object-Detection-With-Two-Path-Convolutional-LSTM-Pyramid.pdf/ (дата обращения: 16.01.2026).

- [2] Hanson A., Koutilya PNV, Sanjukta Krishnagopal, Larry Davis Bidirectional Convolutional LSTM for the Detection of Violence in Videos. 2018. – URL: https://openaccess.thecvf.com/content_eccv_2018_workshops/w10/html/Hanson_Bidirectional_Convolutional_LSTM_or_the_Detection_of_Violence_in_Videos_ECCVW_2018_paper.html (дата обращения: 15.01.2026).
- [3] Howard A.G., Menglong Zhu, Bo Chen, Kalenichenko D. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. – URL: <https://arxiv.org/pdf/1704.04861/> (access date: 11.01.2026).
- [4] Iandola F.N., Han S., Moskewicz M.W., Ashraf K., Dally W.J., Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size Recognition. 2016. Published as a conference paper at ICLR. 2015. – URL: https://www.researchgate.net/publication/301878495_SqueezeNet_AlexNet-level_accuracy_with_50x_fewer_parameters_and_05MB_model_size (дата обращения: 10.01.2026).
- [5] Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015. – URL: <https://arxiv.org/pdf/1502.03167/> (access date: 13.01.2026).
- [6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition. 2015. – URL: <https://arxiv.org/pdf/1512.03385> (access date: 11.01.2026).
- [7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Spatial pyramid pooling in deep convolutional networks for visual recognition. 2015. – URL: <https://arxiv.org/pdf/1406.4729/> (access date: 11.01.2026).
- [8] Krizhevsky I., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks. 2012. – URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (accessed on: 12.01.2026).
- [9] Lavin A., Scott Gray Fast Algorithms for Convolutional Neural Networks. 2015. – URL: <https://arxiv.org/abs/1509.09308> (access date: 11.01.2026).
- [10] Mikolov T., Kai Chen, Corrado G., Jeffrey Dean Efficient Estimation of Word Representations in Vector Space. 2013. – URL: <https://arxiv.org/abs/1301.3781> (access date: 15.01.2026).
- [11] Norman P. Jouppi, Cliff Young, Nishant Patil and others In-Datcenter Performance Analysis of a Tensor Processing Unit. 2017. – URL: <https://arxiv.org/pdf/1704.04760> (access date: 11.01.2026).
- [12] Ronneberger O., Fischer Ph., Brox Th. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. – URL: <https://arxiv.org/pdf/1505.04597/> (access date: 13.01.2026).
- [13] Szegedy C., Wei Liu, Chapel Hill, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich Going Deeper with Convolutions. 2014. – URL: <https://arxiv.org/pdf/1409.4842> (accessed on: 12.01.2026).
- [14] Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2015. Published as a conference paper at ICLR 2015. – URL: <https://arxiv.org/pdf/1409.1556> (access date: 10.01.2026).
- [15] Wei Liu, Anguelov D., Erhan D., Szegedy C., Reed S., Cheng-Yang Fu, Berg A.C. SSD: Single Shot MultiBox Detector. 2016. – URL: <https://arxiv.org/pdf/1512.02325> (access date: 11.01.2026).
- [16] ONNX official website. – URL: <https://onnx.ai/>.
- [17] Fundamentals of artificial intelligence technologies: a textbook/A.A. Semenov, V.V. Gonchar, A.S. Erdniev. – Moscow University of the Ministry of Internal Affairs of Russia named after V.Ya. Kikotya, 2025. – 190s.
- [18] Financial monitoring: training manual/V.I. Glotov, A.U. Albekov, O.N. Tisen et al. – KnoRus, 2022. – 198 s.
- [19] Tisen O.N. Countering modern methods and methods of recruitment to international terrorist organizations//Russian Justice. 2018, №9. S. 51-54.
- [20] Gonchar V.V. On the importance of forming a uniform conceptual apparatus necessary for investigating crimes in the field of computer information//Bulletin of Economic Security. 2018. №1. S. 225-230.
- [21] Orlova A.A., Gonchar V.V. Features of the investigation of crimes committed in the field of information technology//Business security. 2021. № 4. S. 25-29.

