Журнал обчислювальної та прикладної математики **2025, № 1** Journal of Numerical & Applied Mathematics

УДК 519.6
MSC 35R11, 65M06

PRINCIPLES OF LARGE LANGUAGE MODELS (LLM)

P. O. Lysyi

Faculty of Computer Science and Cybernetics, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine, E-mail: pavlo.lysyi@knu.ua, ORCID: 0009-0002-2306-9652

ПРИНЦИПИ РОБОТИ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ (LLM)

П. О. Лисий

Факультет комп'ютерних наук та кібернетики, Київський національний університет імені Тараса Шевченка, Київ, Україна, E-mail: pavlo.lysyi@knu.ua, ORCID: 0009-0002-2306-9652

Abstract. This paper explores the operational principles of large language models (LLMs), focusing in particular on the mechanism of next-token generation within the process of autoregressive modeling. It outlines the theoretical foundations of neural language models, the transformer architecture with its self-attention mechanism, and the roles of tokenization and embedding in forming the input representation of text. The study analyzes the main methods for selecting the next token (greedy decoding, top-k sampling, top-p sampling, temperature), their impact on the stochasticity of results, and the trade-off between coherence and creativity. It also examines context length limitations, sources of training data, and challenges related to interpretability and the likelihood of «hallucinations». The article provides a comprehensive overview of the architectural and algorithmic foundations behind text generation in LLMs.
Keywords: large language models, transformer, autoregressive generation, tokenization, temperature, probabilistic text modeling.

Анотація. У статті досліджено принципи роботи великих мовних моделей (LLM), зокрема механізм генерації наступного токена в процесі автогресивного моделювання. Описано теоретичні основи нейронних мовних моделей, трансформерну архітектуру з механізмом самоуваги, а також роль токенізації та ембеддингу у формуванні вхідного представлення тексту. Проаналізовано основні методи вибору наступного токена (жадібне декодування, top-k, top-p семплінг, температура), їхній вплив на стохастичність результатів і баланс між узгодженістю та креативністю.

Розглянуто обмеження довжини контексту, джерела тренувальних даних і виклики, пов'язані з інтерпретованістю та ймовірністю «галюцинацій». Стаття є цілісним оглядом архітектурних та алгоритмічних рішень, що лежать в основі генерації тексту LLM.
Ключові слова: великі мовні моделі, трансформер, автогресивна генерація, токенізація, температура, ймовірнісне моделювання тексту.

Вступ

Великі мовні моделі (LLM) представляють собою потужні архітектури штучного інтелекту на основі глибоких нейронних мереж, натренованих на масштабних корпусах текстових даних.

Завдяки здатності моделювати умовний розподіл наступного токена в послідовності, LLM демонструють високі результати в широкому спектрі задач природної мовної обробки: від автозавершення фраз і машинного перекладу до побудови розгорнутих логічних відповідей на запити. Їхній фундаментальний принцип — авторегресивне моделювання [1] — передбачає поетапне прогнозування наступного токена на основі всього попереднього контексту. Саме тому питання механізмів вибору наступного токена є центральним для розуміння природи генеративних можливостей таких моделей.

У даній роботі здійснено системний огляд архітектурних і алгоритмічних засад, що лежать в основі процесу генерації тексту у LLM. Висвітлено теоретичні передумови нейронних мовних моделей, охоплено ключові компоненти трансформерної архітектури, зокрема механізм самоуваги (self-attention) [2], а також проаналізовано роль токенізації й ембеддингів у формуванні вхідного представлення.

Окрему увагу приділено деталізації процедури авторегресивного декодування: від обчислення логітів та застосування softmax-функції до покрокового вибору токенів у процесі генерації.

Розглянуто вплив довжини контекстного вікна на якість вихідного тексту, а також обговорено основні стратегії вибору наступного токена — жадібне декодування, top-k, nucleus (top-p) семплінг та температурне масштабування [3] — й охарактеризовано їхній ефект на детермінованість, варіативність і когерентність результатів.

У завершальній частині роботи проаналізовано ключові виклики, притаманні LLM: схильність до галюцинацій, обмежена інтерпретованість внутрішніх представлень, а також компроміси між креативністю і точністю.

Запропонована структура статті відповідає логіці викладу зазначених компонентів, що сприяє її цілісності та зручності навігації.

1. Теоретичні засади LLM

Великі нейронні мовні моделі (LLM) призначені для оцінювання ймовірності послідовності токенів $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, де кожен токен x_t належить до словника $\mathcal{V}$.

Основною метою є моделювання умовної ймовірності генерації тексту:

$$P(\mathbf{x}) = \prod_{t=1}^{T} P(x_t \mid x_1, x_2, \ldots, x_{t-1}).$$

Така факторизація, відома як авторегресивна модель, дозволяє ефективно прогнозувати наступний токен на основі попередніх, що є фундаментом для завдань генерації тексту, перекладу та узагальнення.

Історично для моделювання послідовностей застосовувалися рекурентні нейронні мережі (RNN) [4], зокрема довга короткочасна пам'ять (LSTM) [5] та гейтовані рекурентні одиниці (GRU) [6]. Математично, рекурентна модель оновлює прихований стан $\mathbf{h}_t$ за формулою:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{E}(x_t)),$$

де $\mathbf{E}(x_t)$ — ембедінг токена x_t, а f — нелінійна трансформація (LSTM, GRU).

Проте рекурентні моделі мають обмеження у захопленні довготривалих залежностей через проблему згасання градієнтів. Цю проблему вдало розв'язали трансформерні архітектури, запропоновані в роботі [2].

Трансформер базується на механізмі самоуваги (self-attention), який для кожного позиційного токена t обчислює ваги уваги до всіх інших токенів у послідовності. Формально, для заданих ключів $\mathbf{K}$, запитів $\mathbf{Q}$ і значень $\mathbf{V}$ обчислення уваги має вигляд:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V},$$

де d_k — розмірність ключів, а softmax застосовується по рядках матриці.

У трансформері для кожного шару послідовність токенів представлена у вигляді матриці ембедінгів $\mathbf{X} \in \mathbb{R}^{T \times d}$. Для отримання $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ застосовують лінійні проєкції:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}^K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}^V,$$

де $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d_k}$ — параметри навчання.

Завдяки механізму уваги модель здатна фокусуватися на релевантних попередніх токенах незалежно від їх позиції, що дає змогу захоплювати довготривалі залежності в тексті.

Кожен шар трансформера включає багатоголову увагу (Multi-Head Attention), яка агрегує інформацію з кількох підпросторів ознак:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O,$$

де

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V),$$

а $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V, \mathbf{W}^O$ — матриці ваг для i-ї голови.

Після блока уваги застосовується позиційно-залежний багатошаровий перцептрон (feed-forward network, FFN):

$$\text{FFN}(\mathbf{z}) = \text{ReLU}(\mathbf{z}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2,$$

що дає додаткову нелінійність і підсилює здатність моделі до апроксимації складних функцій.

Великі мовні моделі, такі як GPT (Generative Pre-trained Transformer) [7], BERT (Bidirectional Encoder Representations from Transformers) [8] та їх численні варіації, мають сотні мільйонів або навіть мільярди параметрів, що дозволяє моделювати глибокі статистичні закономірності мови.

Навчання здійснюється на великих корпусах текстів із застосуванням різних цілей:

– Автогресивне навчання наступного токена:

$$\mathcal{L}_{\text{causal}} = -\sum_{t=1}^{T} \log P(x_t \mid x_1, \ldots, x_{t-1}),$$

– Масковане відновлення токенів (masked language modeling) [9]:

$$\mathcal{L}_{\text{masked}} = -\sum_{t \in \mathcal{M}} \log P(x_t \mid \mathbf{x}_{\setminus \mathcal{M}}),$$

де $\mathcal{M} \subset \{1, \ldots, T\}$ — множина замаскованих позицій.

Таким чином, LLM формують глибоке семантичне і синтаксичне розуміння мови, що робить їх універсальними інструментами у завданнях обробки природної мови.

2. Створення датасету для навчання LLM

Створення датасету для навчання великих мовних моделей (LLM) — це ключовий етап, що визначає ефективність і якість подальшої роботи моделі. Попереднє навчання є першою та найважливішою стадією у створенні LLM, що закладає фундамент їхньої здатності розуміти та генерувати людський текст. Воно займає близько 99% часу та обчислювальних ресурсів, необхідних для навчання LLM.

Під час цього етапу модель навчається на величезних датасетах, зібраних із різноманітних джерел інтернету, що дає змогу їй засвоїти закономірності, структури та нюанси людської мови. Успішність попереднього навчання значною мірою залежить від якості та різноманітності використаних даних.

Для навчання LLM застосовуються так звані *Data Mixtures* — великі та різноманітні корпуси текстів, що охоплюють широкий спектр стилів, тем і форматів. Основні джерела таких даних включають:

– GitHub — для коду та технічного контенту;
– Wikipedia — для загальних знань;
– ArXiv — для наукових досліджень;
– Книги — для довгих наративів та структурованих текстів;
– StackExchange — для формату запитань і відповідей;
– CommonCrawl та C4 — для загального контенту веб-мережі.

Застосування таких різноманітних джерел забезпечує широке охоплення стилів та контекстів, що дозволяє моделі розвивати глибоке та багатогранне розуміння мови.

3. Кодування тексту в LLM

На відміну від людини, велика мовна модель (LLM) не оперує безпосередньо словами в їхній лінгвістичній формі. Для обробки тексту природної мови на рівні нейронної архітектури необхідне його попереднє перетворення в числове представлення. Цей процес складається з двох ключових етапів: токенізації та ембедінгу.

На етапі токенізації вхідний текст розбивається на послідовність дискретних одиниць — токенів, які можуть відповідати словам, частинам слів або навіть окремим байтам. У сучасних LLM, як-от GPT-2 чи GPT-4, типовим підходом є субсловесна токенізація [10]. Зокрема, метод Byte Pair Encoding (BPE) [11] дозволяє ефективно працювати з відкритим словником: частовживані лексеми кодуються як окремі токени, тоді як рідкісні або неологізми розкладаються на коротші підслівні послідовності. Така стратегія забезпечує компроміс між продуктивністю моделі та її здатністю до генералізації на нові слова. Наприклад, GPT-2 використовує модифікований байтовий BPE з фіксованим словником розміром 50 257 токенів.

Після токенізації кожен токен перетворюється на векторне представлення у багатовимірному просторі через ембедінг-шар. До цього вектору додається позиційне кодування (positional encoding), яке забезпечує модель інформацією про порядок токенів у послідовності. Це критично важливо, оскільки базова трансформерна архітектура є порядково-агностичною і не має вбудованого механізму врахування послідовності без додаткового позиційного сигналу.

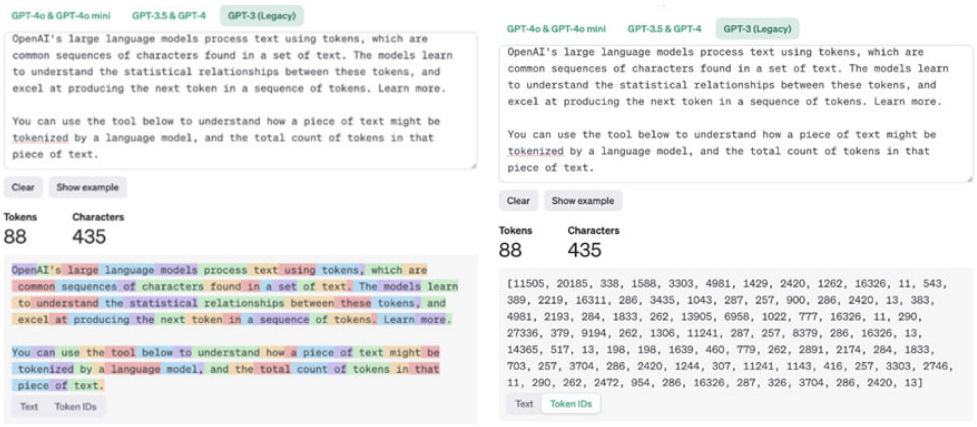

Рис. 1. Приклад як працює токенізатор та енкодер в моделі GPT-3.

4. Побудова розподілу ймовірностей у великих мовних моделях

Мовні моделі сформульовані в ймовірнісному підході: вони навчаються оцінювати спільну ймовірність послідовності токенів. Завдяки природній порядковості мови, спільну ймовірність послідовності можна розкладати за

правилом множення на умовні ймовірності кожного токена з урахуванням попередніх. Формально, для послідовності токенів $s_1, \ldots, s_n$ ймовірність усієї послідовності моделюється як добуток умовних ймовірностей кожного токена з урахуванням попередніх:

$$p(s_1, \ldots, s_n) = \prod_{t=1}^{n} p\big(s_t \mid s_1, \ldots, s_{t-1}\big).$$

Трансформер навчається оптимізовувати ці умовні ймовірності на великій кількості текстів. По суті, йому ставиться задача правильно передбачати наступний токен як можна ближче до фактичного, що забезпечується мінімізацією перехресної ентропії між передбаченим та фактичним розподілом. Це створює загальну основу: модель є генеративною ймовірнісною моделлю тексту.

На рисунку 2 демонструється ключовий принцип функціонування мовної моделі: фраза «я люблю грати в» може бути продовжена множиною семантично доречних токенів. Це зумовлено тим, що подібні завершення, як-от «футбол», «хокей», «CS:GO» (популярна комп'ютерна гра), «хованки» тощо, з високою ймовірністю траплялися у тренувальному корпусі, і тому мають високу апостеріорну ймовірність бути згенерованими. Навпаки, комбінація «я люблю грати в меркурій» імовірно не представлена в навчальних даних, а отже, ймовірність генерації токена «меркурій» у цьому контексті прямує до нуля.

Таким чином, задача мовної моделі полягає у побудові умовного розподілу ймовірностей наступного токена, базуючись на попередньому контексті, який вона засвоїла шляхом навчання на великомасштабному корпусі текстових даних. Саме це дозволяє моделі генерувати змістовні та граматично коректні послідовності, що узгоджуються з імовірнісною структурою природної мови.

5. Механізм генерації наступного токена

Генерація тексту у великих мовних моделях базується на авторегресивному підході. Модель послідовно генерує текст, передбачаючи наступний токен на основі всього контексту, згенерованого на попередніх кроках.

На кожному кроці вхідними даними є послідовність токенів — це можуть бути слова, підслова або символи, які формують контекст. Модель обробляє цей контекст і обчислює вихідний вектор логітів — числових значень, що відповідають кожному можливому токену у словнику [12]. Логіт z_i для токена i є оцінкою відносної ймовірності його появи на поточному кроці генерації.

Для отримання коректного розподілу ймовірностей по всіх токенах застосовується функція softmax:

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}},$$

де p_i — ймовірність вибору токена i.

Рис. 2. Ілюстрація побудови умовного розподілу ймовірностей у мовній моделі на прикладі продовження фрази «я люблю грати в».

Після цього модель може обрати наступний токен: жадібно (максимум p_i) або вибірково (семплінг за розподілом ймовірностей). Отриманий токен додається до контексту, і цикл повторюється — модель генерує наступний токен, враховуючи вже збільшену послідовність. Такий ітеративний процес триває до досягнення заданої довжини тексту або до появи спеціального токена завершення.

Таким чином, генерація тексту у великих мовних моделях — це поступове, поетапне прогнозування наступного токена на основі усього доступного контексту з урахуванням статистичних розподілів ймовірностей, що формує цілісний та зв'язний текст.

6. Вибір токена

Після обчислення ймовірнісного розподілу наступного токена виникає питання: як саме з нього вибирати слово? Існує кілька стратегій декодування:

Жадібний (greedy) декодинг: модель завжди бере токен з найвищою ймовірністю. Такий підхід детерміністичний, гарантує вивід найдостовірнішого з погляду моделі варіанту, але часто призводить до одноманітних текстів. Оскільки ймовірність одного слова домінує, модель може зациклюватися на найбільш поширених фразах [13]

$$s_t = \operatorname{argmax} P(s \mid s_1, \ldots, s_{t-1})$$

Top-k семплінг: спершу відбирають k токенів з найвищими ймовірностями і нормалізують їх розподіл. Саме з цієї скороченої множини випадково вибирають токен. Наприклад, $k = 5$ означає, що на кожному кроці модель

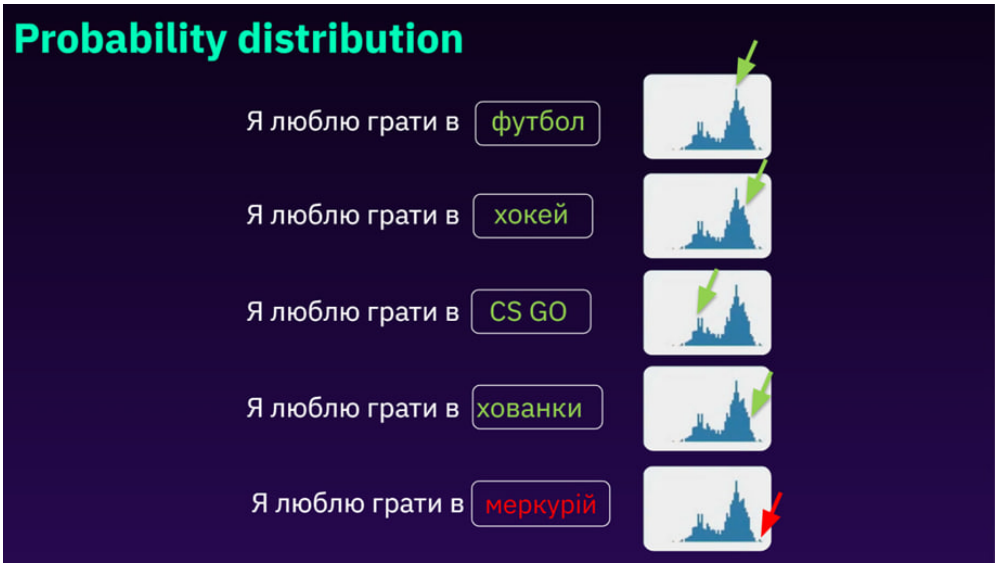

Рис. 3. Ілюстрація вибору наступного токена через модель GPT. Жовта послідовність індексів — контекст речення, зелений блок — випадковий підсвічений токен, червоний — токен, який модель намагається передбачити серед розподілу з 50 257 унікальних токенів.

обмежена п'ятьма найімовірнішими токенами. Це зменшує шанс надмірної диверсифікації або вибору дуже рідкісних слів, зберігаючи певну варіативність [14]

$$s_t \sim \text{Categorical}(\tilde{P}_k(s \mid s_1, \ldots, s_{t-1}, x))$$

Top-p (nucleus) семплінг: Обирається найбільша множина токенів так, щоб їх сумарна ймовірність дорівнювала або перевищувала поріг p (наприклад, 0.9 або 0.95). Тобто збирають найімовірніші слова, поки їх кумулятивна ймовірність не досягне p, а решту «довгого хвоста» відкидають. Це динамічне обмеження забезпечує гнучкість: якщо розподіл рівномірніший, до вибірки потрапляє більше слів, якщо ж один логіт домінує — лише кілька. Nucleus sampling часто дає більш креативний та зрозумілий текст у порівнянні з top-k, оскільки адаптується до форми розподілу [15]

$$s_t \sim \text{Categorical}(\tilde{P}_p(s \mid s_1, \ldots, s_{t-1}, x))$$

Параметр температури (temperature): налаштовує «гостроту» розподілу ймовірностей. При $T = 1$ розподіл залишається незмінним. Зниження $T < 1$ «нахиляє» розподіл, підвищуючи ймовірність найбільш вірогідних токенів, що призводить до більш консервативних і детерміністичних текстів. Збільшення $T > 1$ робить розподіл рівномірнішим, даючи рідкісним

словам більше шансів. Висока температура стимулює більш творчі, незвичайні відповіді. Наприклад, без температури модель може завжди відповідати «зеленим» на запитання про улюблений колір, а із підвищеною температурою час від часу вибиратиме рідкісніші варіанти (яскраві відтінки), роблячи відповідь менш банальною [16].

Вибір стратегії декодування критично впливає на результуючий текст: жадібний метод дає стабільні, але однотипні відповіді, у той час як стохастичні методи (top-k, top-p, налаштування температури) підвищують різноманіття і креативність за рахунок зменшення узгодженості. Підбір параметрів зазвичай залежить від задачі: для точних технічних відповідей частіше використовують низьку температуру та невеликі значення top-k, а для генерації творчих ідей — більшу температуру та швидкість top-p.

7. Всі кроки генерації тексту великою мовною моделлю

Загальна процедура автогресивної генерації тексту у великих мовних моделях (LLM) включає такі кроки:

1. **Формування контексту.** Модель приймає як вхідну послідовність токенів, яка може складатися з початкового запиту або вже згенерованих раніше токенів.
2. **Обчислення логітів.** На основі вхідного контексту трансформер генерує вектор логітів — ненормалізованих скалярних значень, що відповідають кожному токену у словнику. Ці значення відображають «сирі» оцінки ймовірності появи відповідного токена на поточній позиції.
3. **Нормалізація та вибір токена.** Функція softmax перетворює логіти на розподіл ймовірностей p_i по всіх можливих токенах:

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}},$$

де z_i — логіт для токена i. На цьому етапі здійснюється вибір наступного токена — або детерміновано (жадібний вибір токена з максимальною ймовірністю, *greedy decoding*), або стохастично (семплінг із отриманого розподілу), або із застосуванням вдосконалених стратегій, таких як top-k, nucleus (top-p) семплінг чи температурне масштабування.
4. **Оновлення контексту.** Обраний токен додається до послідовності, що подається на вхід моделі, після чого процедура повторюється для прогнозування наступного токена.

Розглянемо приклад: вхідна фраза «Кіт сидів на» після токенізації подається на вхід моделі. На основі контексту попередніх токенів модель обчислює розподіл ймовірностей для всіх можливих наступних токенів. Якщо, наприклад, токен «дивані» отримує найвищу умовну ймовірність, то в разі використання жадібного підходу (greedy decoding) саме цей токен буде обрано як наступний. У результаті сформується фраза «Кіт сидів на

дивані», яка, своєю чергою, може бути продовжена наступними кроками генерації.

Зазначимо, що хоча сама модель навчається шляхом максимізації правдоподібності (likelihood) спостережуваних послідовностей у тренувальних даних, пряме використання найбільш ймовірного токена на кожному кроці під час генерації часто призводить до монотонних, передбачуваних або надмірно повторюваних відповідей.

Для підвищення різноманітності та природності згенерованого тексту застосовуються стохастичні методи вибору токена, такі як top-k, nucleus (top-p) семплінг та температурне масштабування, які будуть розглянуті нижче.

8. Вплив контексту та механізму уваги

Одним із центральних чинників, що визначає якість текстової генерації у великих мовних моделях, є обсяг і структура контексту.

Архітектура трансформера реалізує механізм самоуваги (self-attention), завдяки якому кожен токен вхідної послідовності на кожному кроці може «уважно» розглядати усі попередні токени, зважуючи їхню релевантність. Формально, для кожної позиції моделі обчислюються коефіцієнти важливості (attention weights) між усіма парами токенів, що дозволяє захоплювати як локальні, так і віддалені залежності у тексті.

Цей механізм дає змогу ефективно моделювати складні зв'язки, зокрема кореференції, віддалені граматичні конструкції та семантичні відповідності. Наприклад, модель може коректно узгоджувати займенники з відповідними іменниками, навіть якщо вони знаходяться на значній відстані один від одного в тексті.

Втім, обсяг контексту обмежений довжиною контекстного вікна — максимальною кількістю токенів, яку модель здатна опрацювати одночасно. Наприклад, GPT-2 підтримує контекст до 1024 токенів, GPT-3 — до приблизно 2048, тоді як сучасні конфігурації GPT-4 здатні оперувати 8192 токенами або більше. Збільшення контекстного вікна підвищує здатність моделі працювати з довгими текстами, однак суттєво зростає і обчислювальна вартість: обчислення самоуваги має квадратичну складність $O(n^2)$ відносно довжини вхідної послідовності.

Наслідком цього є те, що практична здатність моделі «пам'ятати» великі обсяги інформації обмежується апаратними ресурсами. Більше того, у ряді задач внесок далеких токенів у генерацію нових швидко зменшується, а сама модель має тенденцію зосереджувати обчислювальну увагу на останніх елементах послідовності.

Аби розширити ефективну довжину контексту без квадратичного зростання обчислень, активно досліджуються архітектурні модифікації: Sparse Attention [17], Longformer [18], Performer [19], Recurrent Memory Transformers [20] та інші підходи. Вони спрямовані на те, щоб зменшити складність механізму самоуваги або ж зберігати проміжні представлення у вигляді узагальненої пам'яті.

Таким чином, хоча механізм самоуваги забезпечує гнучке й динамічне використання контексту при кожному кроці генерації, реальні обмеження контекстного вікна залишаються важливим технічним бар'єром для довгострокової когерентності тексту, що генерується.

9. Обмеження та виклики

Незважаючи на значні успіхи у розвитку великих мовних моделей (LLM), їх генеративні можливості супроводжуються низкою суттєвих обмежень.

По-перше, обмеження довжини контекстного вікна залишається фундаментальним фактором. Максимальна кількість токенів у контексті зазвичай не перевищує кілька тисяч, і подвоєння цієї довжини спричиняє квадратичне збільшення обчислювальних витрат, що суттєво впливає на ресурси системи. Унаслідок, для роботи з довгими текстами застосовують стратегії їх розбиття або компресії, що, однак, пов'язано з ризиком втрати важливої інформації та контекстуальних нюансів.

По-друге, проблемою є явище так званих «галюцинацій» — генерація впевнених, але хибних або неперевірених фактів. Незважаючи на те, що текст, створений моделлю, часто виглядає правдоподібним і когерентним, він може містити суттєві неточності. Наукові дослідження демонструють, що феномен «галюцинації» є внутрішньою властивістю моделей LLM: навіть за умови ідеальних навчальних даних та архітектур повністю виключити помилкові відповіді неможливо. Для мінімізації негативних наслідків у практичних застосуваннях використовують додаткові механізми контролю якості, такі як жорсткі фільтри, системи інформаційної верифікації (наприклад, Retrieval-Augmented Generation — RAG) [21] та зв'язок із зовнішніми базами знань.

Ще однією суттєвою проблемою є низька інтерпретованість моделей. Сучасні LLM здебільшого є «чорними ящиками», що ускладнює пояснення внутрішніх рішень і вибору відповіді.

Відсутність прозорості істотно обмежує їх застосування в галузях, де критично важлива обґрунтованість висновків, зокрема в медицині, юриспруденції та інших сферах, що вимагають високої відповідальності.

Відсутність інтерпретованості також ускладнює нормативне регулювання та формування довіри до таких систем.

Нарешті, існують неминучі компроміси між різними цілями оптимізації: продуктивністю та якістю відповіді, швидкістю обробки та глибиною моделі, креативністю та точністю.

Збільшення розмірів моделі та обсягів навчальних даних покращує результати, але водночас призводить до значних витрат ресурсів та часу на навчання.

У практичних системах часто доводиться балансувати ці фактори, інколи штучно скорочуючи контекст або застосовуючи інші евристики для збереження швидкості, що потенційно може негативно впливати на якість генерації.

10. Важливість глибинного розуміння роботи великих мовних моделей

Ми провели експериментальне дослідження з метою оцінити, наскільки незначні, на перший погляд, зміни у структурі запиту впливають на якість відповідей великої мовної моделі (LLM). Результати дослідження підкреслюють критичну важливість глибокого розуміння механізмів генерації наступного токена, а також принципів роботи трансформерних моделей.

У межах експерименту були сформульовані логічні задачі, що вимагали точного математичного мислення. Модель мала надати розв'язок на низку логічних питань. Приклад задачі наведено нижче:

> «Начальник наймає працівника з умовою: за кожен робочий день — +20$, за кожен неробочий — –30$. Через 60 днів працівник нічого не заробив. Скільки було робочих днів?»

Із цими задачами були сформовані промпти, які відрізнялися лише порядком полів у очікуваному JSON-виводі:

- Промпт 1: «Повертай JSON з 2 полями: відповідь та пояснення»
- Промпт 2: «Повертай JSON з 2 полями: пояснення та відповідь»

На кожен промпт було надіслано по 300 ідентичних запитів з фіксованими параметрами (модель: `gpt-3.5-turbo`, температура: 0).

Результати:

- Промпт 1 (відповідь ? пояснення): в середньому 97 правильних відповідей із 300
- Промпт 2 (пояснення ? відповідь): в середньому 124 правильних відповідей із 300

Спостерігається покращення якості відповідей приблизно на 27% у другому випадку. Сам факт, що зміна порядку полів у JSON-виводі вплинула на здатність моделі знайти правильне розв'язання, є показовим. Це свідчить, що модель приймає рішення не лише на основі змісту задачі, а й опирається на структурні сигнали у промпті, які модулюють розподіл ймовірностей генерації наступного токена.

Це явище має логічне пояснення: великі мовні моделі генерують текст послідовно, токен за токеном, і порядок полів у форматі JSON визначає логіку формування відповіді. Якщо поле «пояснення» розміщується перед полем «відповідь», модель спочатку генерує змістовне пояснення, яке потім слугує основою для формування відповіді. У зворотній послідовності модель спершу формує відповідь, а потім підбирає відповідне пояснення, що найкраще узгоджується з уже сформованою відповіддю.

Цей феномен має аналогії з когнітивними процесами людини, оскільки у повсякденному житті пояснення часто конструюються заднім числом, формуючи раціональні обґрунтування на основі вже наявних фактів. Таким чином, навіть якщо істинні причини подій інші, логічне пояснення підлаштовується під сформований факт для створення когерентної наративної структури.

11. Висновки

У даній статті було досліджено принципи роботи великих мовних моделей (LLM), зокрема механізми генерації тексту на основі автогресивного моделювання. Розглянуто ключові компоненти архітектури трансформерів, такі як механізм самоуваги (self-attention), який дозволяє моделям ефективно захоплювати довготривалі залежності в тексті. Проаналізовано процес токенізації та ембеддингу, що забезпечують перехід від тексту до числового представлення, придатного для обробки нейронними мережами.

Окрему увагу приділено стратегіям вибору наступного токена, включаючи детерміновані (жадібне декодування) та стохастичні (top-k, top-p семплінг, температурне масштабування) методи. Показано, що ці підходи впливають на баланс між когерентністю та варіативністю згенерованого тексту. Також обговорено обмеження LLM, такі як скінченна довжина контексту, схильність до «галюцинацій» і труднощі інтерпретації внутрішніх механізмів моделі.

Експериментальні результати підтвердили, що навіть незначні зміни у структурі запиту (наприклад, порядок полів у JSON-відповіді) можуть суттєво впливати на якість генерації. Це свідчить про те, що LLM не просто пасивно відтворюють інформацію, а активно інтерпретують контекст, формуючи послідовність токенів на основі ймовірнісних закономірностей.

Перспективи подальших досліджень:

1. Оптимізація стратегій декодування — пошук методів, що забезпечують оптимальний баланс між креативністю і точністю.
2. Розширення контекстного вікна — подолання обмежень довжини контексту за рахунок ефективніших архітектур (наприклад, sparse attention).
3. Зменшення галюцинацій — інтеграція зовнішніх баз знань (RAG) та механізмів верифікації фактів.
4. Підвищення інтерпретованості — розробка методів візуалізації та аналізу внутрішніх представлень моделі.

Таким чином, розуміння принципів роботи LLM є критично важливим для їх ефективного застосування в реальних задачах, від автоматизації контенту до наукового аналізу. Майбутні дослідження мають сприяти подоланню поточних обмежень і розширенню можливостей генеративних мовних моделей.

Автор заявляє про відсутність конфлікту інтересів щодо публікації цієї статті.

Література

1. Schuurmans D., Dai H., Zanini F. Autoregressive Large Language Models are Computationally Universal. *arXiv preprint arXiv:2410.03170 [cs.CL].* 2024.
2. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention Is All You Need. *arXiv preprint arXiv:1706.03762.* 2017.
3. Grubisic D., Cummins C., Seeker V., Leather H. Priority Sampling of Large Language Models for Compilers. *arXiv preprint arXiv:2402.18734.* 2024.

4. Schmidt R. M. Recurrent Neural Networks (RNNs): A Gentle Introduction and Overview. *arXiv preprint arXiv:1912.05911*. 2019.
5. Vennerod C. B., Kjorran A., Bugge E. S. Long Short-term Memory RNN. *arXiv preprint arXiv:2105.06756*. 2021.
6. Chung J., Gulcehre C., Cho K., Bengio Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555*. 2014. Presented at NIPS 2014 Deep Learning and Representation Learning Workshop. https://doi.org/10.48550/arXiv.1412.3555.
7. Yenduri G. et al. Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions. *arXiv preprint arXiv:2305.10435*. 2023.
8. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*. 2018. https://doi.org/10.48550/arXiv.1810.04805.
9. Sinha K. et al. Masked Language Modeling and the Distributional Hypothesis: Order Word Matters Pre-training for Little. *arXiv preprint arXiv:2104.06644*. 2021. https://doi.org/10.48550/arXiv.2104.06644.
10. Batsuren K. et al. Evaluating Subword Tokenization: Alien Subword Composition and OOV Generalization Challenge. *arXiv preprint arXiv:2404.13292*. 2024.
11. Kozma L., Voderholzer J. Theoretical Analysis of Byte-Pair Encoding. *arXiv preprint arXiv:2411.08671*. 2024. https://doi.org/10.48550/arXiv.2411.08671.
12. Ma H., Chen J., Wang G., Zhang C. Estimating LLM Uncertainty with Logits. *arXiv preprint arXiv:2502.00290*. 2025. https://arxiv.org/abs/2502.00290.
13. PEDAL: Prompts based on Exemplar Diversity Aggregated using LLMs. *arXiv preprint arXiv:2408.08869*. 2024. https://doi.org/10.48550/arXiv.2408.08869.
14. Yao Y. et al. Determine-Then-Ensemble: Necessity of Top-k Union for Large Language Model Ensembling. *arXiv preprint arXiv:2410.03777*. 2024.
15. Ravfogel S., Goldberg Y., Goldberger J. Conformal Nucleus Sampling. *arXiv preprint arXiv:2305.02633*. 2023. (Findings of ACL 2023). https://doi.org/10.48550/arXiv.2305.02633.
16. Renze M., Guven E. The Effect of Sampling Temperature on Problem Solving in Large Language Models. *Findings of the Association for Computational Linguistics: EMNLP*. 2024. P. 7346–7356. https://doi.org/10.18653/v1/2024.findings-emnlp.432.
17. Lou C., Jia Z., Zheng Z., Tu K. Sparser is Faster and Less is More: Efficient Sparse Attention for Long-Range Transformers. *arXiv preprint arXiv:2406.16747*. 2024.
18. Beltagy I., Peters M. E., Cohan A. Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150*. 2020.
19. Choromanski K. et al. Rethinking Attention with Performers. *arXiv preprint arXiv:2009.14794*. 2020. https://doi.org/10.48550/arXiv.2009.14794.
20. Bulatov A., Kuratov Y., Burtsev M. Recurrent Memory Transformer. *arXiv preprint arXiv:2207.06881*. 2022. Version 2. https://doi.org/10.48550/arXiv.2207.06881.
21. Lewis P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv preprint arXiv:2005.11401*. 2020.

Надійшла: 27.01.2025 / Прийнята: 27.02.2025