УДК 007:681.512.2

ОТЛИЧИТЕЛЬНЫЕ ОСОБЕННОСТИ ДЕСЯТИЧНЫХ ИЕРАРХИЧЕСКИХ ЧИСЕЛ В ИНТЕРПРЕТАЦИИ ЭМБЕДДИНГОВ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

И. Ю. Каширин, д.т.н., профессор кафедры ВПМ РГРТУ Рязань, Россия;
orcid.org/0000-0003-1694-7410, e-mail: igor-kashirin@mail.ru

Рассматривается новый метод анализа входных естественно-языковых предложений для языковых LLM моделей. Основой нового метода является алгебра десятичных иерархических чисел, используемая в алгоритмах вычисления семантической близости слов, словосочетаний и предложений. Метод пригоден для локальных предметных областей и был апробирован в предметной области «политические новости». Для этой локальной области были разработаны OWL онтология и соответствующее графическое представление в форме семантической сети с разметкой базовых сущностей десятичными иерархическими числами. Семантическая сеть включает общий и прикладной уровни. Фрагмент общей онтологии представлен отношениями, дающими возможность существенно понизить вычислительную сложность алгебраических операций на графах знаний и, как следствие, уменьшить время вычисления семантического сходства естественно-языковых конструкций.

В программной реализации рассмотренного метода используется известная технология языковых нейронных сетей с концентрацией внимания DistilBERT. Обогащение знаний предобученной нейронной сети осуществляется с помощью генерации новых семантических эмбеддингов для слов (сущностей) естественно-языковых предложений и их встраивания в новую нейронную сеть перед дообучением (finetuning) в локальной предметной области.

В качестве обучающих корпусов для получения новой нейросетевой модели mIYu-bert v.2.0 взяты общий корпус из репозитория Hugging Face Datasets и локальный корпус материалов, извлеченных автором из англоязычных политических статей международных электронных средств массовой информации, в частности RT, Meduza, cnn, TASS, NYTimes, bloomberg, WSJ.

Экспериментальная часть материала основана на применении программного инструментария языка программирования Python v.3 (Anaconda 3), LLM DistilBERT и пакета программ mIYu-bert v.3.1. Последний инструментарий реализован автором материала.

Выполненная серия экспериментов позволяет квалифицировать новый метод применения десятичных иерархических чисел в дообучении моделей LLM для вычисления семантического сходства как основу технологии, не уступающей по эффективности имеющимся на сегодняшний день международным аналогам и не превосходящей их по вычислительной сложности.

***Целью работы** является описание нового метода вычисления семантического сходства в языковых LLM моделях с использованием десятичных иерархических чисел на основе OWL онтологий, а также универсальной алгебры при формировании графов знаний в локальных предметных областях.*

***Ключевые слова**: десятичные иерархические числа, универсальные алгебры, языковые DistilBERT-модели, анализ естественного языка, онтологические таксономии, семантическое сходство.*

DOI: 10.21667/1995-4565-2026-95-117-129

Введение

Большие языковые нейросетевые модели (LLM, Large Language Models) [1] могут продолжить свое развитие в метаанализе знаний, заключающемся в исследовании моделью своей собственной организации с целью изменения ее функциональных основ для обретения нового свойства оригинальных форм творческой деятельности. В этом случае произойдет еще один крутой подъем в области новых технологий искусственного интеллекта (ИИ), который превзойдет человека в его творческих способностях. Существенной проблемой станет и кибербезопасность для эпохи ИИ. Эта проблема наиболее хорошо исследована в Китае, где

Технический комитет по стандартизации информационной безопасности (ТС260) опубликовал подробное руководство по реагированию на инциденты с генеративным ИИ [2]. В то же время развитие этих технологий остановить столь же бессмысленно, сколько и невозможно в силу развития информационных войн, которые запретить нельзя.

Одним из шагов в сторону развития отечественных российских технологий ИИ является применение алгебраической математики иерархических чисел, впервые сформулированной в [3]. Перспективность работ в этой области подтверждается многочисленными научными исследованиями иерархических эмбеддингов в LLM [4-7].

Иерархические числа дают возможность использовать компактную векторизацию семантических пространств с предварительным обучением LLM для совершенствования их внутренней структуры на основе метаанализа знаний. Рассмотрение результатов в этой области и является **целью** настоящей статьи.

Алгебраическая система десятичных иерархических чисел

Кратко идея иерархических чисел выглядит так. Пусть N – множество целых чисел, включающих 0 и отрицательные элементы. N = {…, -3, -2, -1, 0, 1, 2, 3 , …}. Пусть также есть выделенный символ «.».

Множество A = N ∪ «.» определяется как алфавит с целыми числами n ∈ N, «∪» –операция объединения множеств, «.» – символ-разделитель уровней, «∈» – теоретико-множественное отношение принадлежности элемента множеству. Тогда грамматика

$$h \rightarrow < n >,\ h \rightarrow < n > . <h>$$

описывает множество бинарных иерархических чисел H с элементами h.

Примеры иерархических чисел: -42.0.1.0.4, 4, -12, 0.1.1.0.

Иерархические числа имеют графическую интерпретацию, в которой соответствуют численным индексам вершин деревьев с одной корневой вершиной n ∈ N. Количество таких деревьев равномощно множеству N, поскольку корневой вершиной любого такого дерева является вершина n, не имеющая в своем написании символа «.».

Спуск в вершину n_1 на один уровень вниз от вершины n_x производится бинарной операцией «+» ($n_x + n_y = n_x . n_y$), подъем от вершины n_1. n_2 вверх – унарной операцией «--»($n_1 . n_2$ -- = n_1, 0-- = 0). Таким образом, любое иерархическое число либо является целым числом, либо начинается с целого числа и содержит в своей структуре целые числа, разделенные точками.

Приведем несколько примеров.

Пусть { n_1 , n_2 , n_3, n_4, n_x } ⊂ **N,** где «⊂» – отношение «подмножество множества».

$$n_1 . n_2 + n_3 . n_4 = n_1 . n_2 . n_3 . n_4 ,\ n_1 . n_2 . n_3 \text{ -- } = n_1 . n_2,$$

$$0.n_x + 0 = 0.n_x.0,\ 0.0 + 0 = 0.0.0,\ 0 + -4 = 0.-4,\ n_x \text{ -- } = n_x,$$

$$41\text{ -- } = 41,\ -33 + 0 = -33.0,\ -33\text{ -- } = -33.$$

Более сложными являются операции {º , ^, | |}.

Операция «º» вычисляет общего предка двух вершин, например:

$$45.-1.0.0.12\ º\ 45.-1.1.0.12 = 45.-1,\quad 1.-2.14\ º\ -5.3.3.3 = 0.$$

То есть при отсутствии общего предка результатом операции является 0.

Операция «^» вычисляет путь из одной вершины в другую через общего предка. При этом иерархическое число, идентифицирующее общего предка, опускается, например:

$$45.-1.0.0.14\ \ ^\ 45.-1.1.0.12 = [(45.-1.0.0.14 \rightarrow 45.-1.0.0 \rightarrow 45.-1.0 \rightarrow 45.-1) \rightarrow$$

$$(45.-1.1 \rightarrow 45.-1.1.0 \rightarrow 45.-1.1.0.12)] = 14.0.0.1.0.12,$$

где символом «→» обозначен один шаг перехода по дереву от вершины к вершине.

Наконец, унарная операция «| |» вычисляет длину иерархического числа, например:

$$14.0.0.1.0.12| = 6.$$

Здесь используется свойство алгебры иерархических чисел H, определяемое как алгебра, объемлющая формальную арифметику целых чисел. Действительно, алгебра целых чисел

(арифметика) является частным случаем алгебры H, если доопределить на ней арифметические операции сложения, вычитания и умножения чисел.

После рассмотрения семантики приведенных операций можно задать универсальную арифметическую алгебру иерархических чисел H:

$$H = < H, \Omega >, \Omega = \{+ (2), --(1), º(2) , ^(2), | |(1)\},$$

где H – множество-носитель, а Ω – сигнатура алгебры, т.е. множество операций. Числа в круглых скобках обозначают местность операций, т.е. количество аргументов.

Операции {+, º , ^} являются бинарными, операции {--, | |} определены как унарные.

Рассмотренную алгебру можно дополнить до алгебраической системы Hs = < H, Ω, R >, введя множество отношений R = {< , >, ∼, = }, где отношения «a > b» и « b < a » соответственно «число a сложней числа b» и «число b короче числа a». Символами «∼» (a∼b) и « = » (a = b) обозначены отношения соответственно «равенство длин чисел a и b» и полное совпадение чисел.

Идея использования десятичных иерархических чисел

Применение иерархических чисел при проектировании онтологических таксономий для LLM основано на вычислении иерархических эмбеддингов (embeddings). Семантические многомерные пространства задаются тысячами измерений, каждое из которых соответствует одной словарной единице (лексеме). Каждое слово предложения на естественном языке представляет собой вектор в семантическом пространстве, который можно записать как соответствующий многомерный кортеж. В первом приближении каждый элемент такого кортежа – число, вычисленное как семантическая близость слова к другому слову, задающему соответствующее измерение пространства. Такое представление дает возможность рассчитать главенство каждого из слов в предложении. Механизм этого расчета назван «концентрацией внимания» [8].

В то же время LLM, спроектированные на этой основе [9] (GPT 5, RoBERTa-transformers 4.3.0, Claude 3.2, LLaMA 3.2, Yandex/YaLM-100B, Gemini 3, GigaChat, BrainBot и т.п.), имеют существенные недостатки:

- потребность в гигантских вычислительных ресурсах;
- невозможность полноценной творческой генерации;
- отсутствие механизмов самосовершенствования (метаанализа);
- отсутствие знаний в узкоспециализированных предметных областях;
- наличие ошибок в ответах на вопросы клиентов;
- риск нарушения функциональной целостности весов нейронных сетей при дообучении модели;
- отсутствие знаний о современной оперативной информации, появившейся после выпуска LLM.

Функционал творчества и метаанализа требует применения дополнительных математических формализмов, учитывающих внутреннюю структуру знаний, которая включает использование осознанных онтологических таксономий. Парадокс функциональных принципов LLM заключается в том, что они могут по соответствующему запросу пользователя вполне достоверно сгенерировать отдельные фрагменты таких таксономий и выдать результат в качестве ответа, но пока не могут ввести их в базовые схемы своей внутренней организации.

Таким образом, можно сформулировать задачу «башни генеративных LLM», аналогичную задаче «башни алгоритмических языков»: «построить внутреннюю структуру знаний более высокого уровня, используя для этого генеративные возможности существующих LLM». Дополнительным положительным эффектом от такого высокоуровневого построения является упрощение генеративных языковых моделей. Упрощение дает возможность не только использовать метауровневые архитектуры LLM, но и позволяет сократить трудоемкость (вычислительную сложность) применения RAG (Retrieval-Augmented Generation) [10, 11], решающих проблему использования оперативной информации.

На этом этапе автором настоящей статьи предлагается использовать математический формализм десятичных иерархических чисел. Он может быть использован при вычислении иерархических эмбеддингов и существенно упрощает расчет семантической близости словарных конструкций различного уровня в естественном языке.

Оптимизация процессов векторизации слов в семантическом пространстве для получения улучшенных эмбеддингов достигается за счет следующих факторов.

Сокращение используемой оперативной памяти

Иерархические числа являются семантическими индексами понятий и отношений, которые составляют смысл естественно-языкового предложения (словарной конструкции, текста, словосочетания). Эти индексы уже получены в базовых LLM на основе применения принципа «башни генеративных LLM». Индексы отражают место понятия или отношения в родовидовой, меронимической, причинно-следственной и других семантических таксономиях. Вследствие этих фактов вектора семантического пространства (эмбеддинги) становятся существенно компактней.

Так, например, «нападение (1.25.3.1)» → «атака (1.25.3.1.1)» → «бомбардировка (1.25.3.1.1.1), «стрельба (1.25.3.1.1.2)» соответствуют понятиям, связанным транзитивным родовидовым отношением «→» и, следовательно, имеют все смысловые связи, принадлежащие понятию «нападение». Нетрудно заметить, что соответствующие понятиям иерархические индексы это качество наглядно отображают.

То же самое можно наблюдать в причинно-следственной таксономии событий: «война (2.12)» => «атака (2.25.3.1.1)» => «убийство (2.25.3.1.1.3)».

Сказанное свидетельствует о том, что иерархические числа позволяют сократить семантическое пространство, разделив эмбеддинги понятий по уровням, в которых количество измерений существенно меньше, чем в существующих LLM. Становится возможным использование меньших векторов или даже их частей.

Ускорение вычислений

Меньшее количество чисел в векторах делает матричные умножения и вычисления сходства значительно меньше по их вычислительной сложности.

Регуляризация/Интерпретируемость

Явное включение информации об иерархии может помочь модели лучше обобщать и делать более предсказуемые выводы, особенно если задача потребует понимания родовидовых отношений.

Вычисление семантической близости на основе десятичных иерархических чисел

Главным свойством десятичных иерархических чисел является то, что их частным случаем (подалгеброй) является формальная арифметика целых чисел. Для того чтобы это свойство можно было эффективно использовать для вычисления семантической близости, алгебру десятичных иерархических чисел следует расширить следующим образом.

Сигнатуру Ω нужно дополнить арифметическими операциями сложения «$\oplus$», вычитания «$\ominus$», умножения «$\otimes$» и деления нацело «/»:

$$\Omega_A = \{\oplus(2), \ominus(2), \otimes(2), /(2)\},\ \Omega_0 = \{\{+ (2), --(1), {}^{o}(2) , \hat{}(2), |\,|(1)\},$$
$$\Omega = \Omega_A \cup \Omega_0 = \{+ (2), --(1), {}^{o}(2) , \hat{}(2), |\,|(1), \oplus(2), \ominus(2), \otimes(2), /(2)\},$$

где «$\cup$» – теоретико-множественная операция объединения множеств.

Фактически при дополнении сигнатуры ранее рассмотренной алгебры формальной арифметикой Ω_A алгебру Ω пришлось разбить на два подмножества.

Алгебраическая семантика арифметических операций Ω_A описывается простыми алгоритмами.

$a \oplus b = b \oplus a$, пусть a содержит n разрядов (целых чисел, разделенных в иерархическом числе точками) и b содержит m разрядов. Пусть также m больше или равно n. Тогда результатом операции является поразрядное сложение первых n разрядов иерархических чисел a и b.

Оставшиеся m-n разряды копируются из соответствующих разрядов большего по количеству разрядов числа.

Например, для a = 12.-1.0.2.4, b = 3.5.-1.0.0.115 выполняются следующие равенства:

$$12.\text{-}1.0.2.4 \oplus 3.5.\text{-}1.0.0.115 = 3.5.\text{-}1.0.0.115 \oplus 12.\text{-}1.0.2.4 = 15.4.\text{-}1.2.4.115$$

Полностью аналогичен смысл других арифметических операций, с той лишь разницей, что над соответствующими разрядами иерархических чисел-аргументов выполняются соответствующие арифметические операции вычитания, умножения и деления нацело (пример деления нацело классической арифметики: 3/5= 0; 5/2 = 2).

В сделанных определениях метод вычисления семантического сходства для словарных конструкций естественного языка разных уровней выглядит следующим образом.

Пусть есть две словарные конструкции a и b. Это могут быть тексты, предложения, словосочетания и отдельные словоформы. Каждая из конструкций состоит из последовательности слов (словоформ):

$$a = a_1\ a_2\ a_3\ \dots\ a_i \dots\ a_n\ ;\ b = b_1\ b_2\ b_3\ \dots\ b_i\ \dots\ b_m.$$

Каждое из слов $\{ a_1\ a_2\ a_3\ \dots\ a_i \dots\ a_n\ b_1\ b_2\ b_3\ \dots\ b_j\ \dots\ b_m \}$ может быть представлено в родовидовой таксономии своей части речи или лексической категории десятичным иерархическим числом. При этом первый разряд числа является индексом лексической категории.

Примеры

<прилагательное> => <жестокий> => <сокрушительный> => <смертельный>
3 3.5 3.5.1 3.5.1.0

<прилагательное> => <окрашенный> => <черно-белый> => <черный>
3 3.12 3.12.0 3.12.0.0

<прилагательное> => <окрашенный> => <цветной> => <зеленый>
3 3.12 3.12.1 3.12.1.1

<глагол> => <действовать> => <нападать>=><атаковать> => <бомбить>
1 1.3 1.3.7 1.3.7.0 1.3.7.0.0

Для лексических единиц, не имеющих выраженной родовидовой таксономии, могут использоваться иерархические числа с малым числом разрядов.

Примеры

<вопрос> => [качественный] => <как>
8 8.1 8.1.0

<вопрос> => [объектный] => <кто>
8 8.2 8.2.0

<вопрос> => [объектный] => <что>
8 8.2 8.2.1

Для сложных отглагольных словарных конструкций можно закодировать причинно-следственную таксономию.

Примеры

<событие>=><конфликтовать>=><атаковать>=>
10 10.5 10.5.1
<получить жертвы>=><потерпеть поражение>
10.5.1.1 10.5.1.4

<событие>=><конфликтовать>=><атаковать>=>
10 10.5 10.5.1
<получить жертвы>=><получить отпор>
10.5.1.1 10.5.1.2

В программной реализации сложные словарные конструкции идентифицируются с помощью механизма паттернов (шаблонов) или сложных регулярных выражений [12]. В этом случае *дополнительный* иерархический индекс может быть приписан только глаголу словарной конструкции.

Дополнительные индексы – это иерархические числа, приписываемые элементам словарных конструкций (словам), кроме уже имеющихся индексов, например родовидовых.

Таким образом, каждый из элементов $\{a_1\ a_2\ a_3\ \dots\ a_i \dots\ a_n\ b_1\ b_2\ b_3\ \dots\ b_j\ \dots\ b_m\}$ входных словарных конструкций **a** и **b** соответствует списку собственных иерархических индексов I, J.

Впрочем, очень часто это одноэлементный список. Можно этот факт записать следующим формальным выражением.

$\{ a_1 (i_{11}, i_{12}, i_{13}, \ldots) a_2 (i_{21}, i_{22}, i_{23}, \ldots) \ldots a_i \ldots a_n(i_{n1}, i_{n2}, i_{n3}, \ldots) b_1 (j_{11}, j_{12} \ldots) \ldots b_m (b_{m1}, b_{m2}, \ldots)\}$

Далее суть вычисления семантического сходства **a** и **b** заключается в попарном сопоставлении всех индексов всех слов из **a** со всеми индексами всех слов из **b**, т.е. каждого с каждым. После этого сравнения для каждой сопоставленной пары слов выбирается их наибольшее общее иерархическое число, для которого вычисляется его длина (количество разрядов). Это одноразрядное число по определению десятичных иерархических чисел тоже относится к иерархическим числам.

Введем операции $ind(a_x)$, $ind(b_y)$, имеющие следующий смысл.

$$ind(a_x) = ind(a_x (i_{x1}, i_{x2}, \ldots, i_{x\text{-}end})) = \{ i_{x1}, i_{x2}, \ldots, i_{x\text{-}end} \},$$
$$ind(b_y) = ind(b_y(j_{y1}, j_{y2}, \ldots, j_{y\text{-}end})) = \{ j_{y1}, j_{y2}, \ldots, j_{y\text{-}end} \}.$$

Вычисление множества всех пар индексов для слов a_x и b_y конструкций **a** и **b** вычисляется следующим выражением.

$$D(a_x, b_y) = ind(a_x) \times ind(b_y) = \bigcup_{i_x = i_{x1}, j_y = j_{y1}}^{i_{x\text{-}end}, j_{y\text{-}end}} i_x) i_i \ ,$$

где «×» – декартово произведение множеств, «∪» – объединение множеств.

Далее введем функцию O.

$$O:(i_x, j_j) \rightarrow^{o} (i_x, j_j), \ Max(|O(D(a_x, b_y))|),$$

где «O» – функция вычисления общей части иерархических индексов для каждой пары индексов из множества пар $D(a_x, b_y)$. Ее основой является операция «^{O}» из алгебры иерархических чисел (для одной пары).

Функция O в качестве результата выдает множество общих частей пар иерархических чисел $D(a_x, b_y)$, т.е. множество индексов, но не пар. Эти индексы являются результатами сопоставления всех иерархических индексов слова a_x со словом b_y для словарных конструкций **a** и **b**.

Унарная операция «| |» алгебры иерархических чисел превращает множество индексов, вычисленных функцией O, в одноразрядные иерархические числа, совпадающие с длиной индексов как количеством разрядов.

Наконец, функция Max выбирает наибольшее число из всего множества чисел после операции «| |».

Для упрощения выражение $Max(|O(D(a_x, b_y))|)$ обозначим как $L(a_x, b_y)$:

$$L(a_x, b_y) = Max(\left|O(D(a_x, b_y))\right|.$$

Иными словами, $L(a_x, b_y)$ – функция вычисления длины максимального общего индекса для всех пар индексов слов a_x и b_y. Чем больше эта величина, тем более близкими по смыслу являются сопоставленные слова.

Для вычисления семантического сходства S двух словарных конструкций **a** и **b** в целом используется следующее выражение.

$$S = 2 * \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} 2 * L(a_i, b_j) / (\left|a_i\right| + \left|b_j\right|)}{\left|a\right| + \left|b\right|}.$$

Это выражение уже использовалось ранее для более простых вычислений [9], в которых слова в словарных конструкциях использовали единственный двоичный иерархический индекс. В настоящей статье этот случай расширен на полиморфические (множественные) таксономии с индексацией десятичными иерархическими числами. В программной реализации рассмотренного метода для вычисления семантического сходства после этапа вычисления множества $O(D(a_x, b_y))$ можно использовать функцию нормализации десятичных чисел soft-max [13,14].

Проектные решения применения десятичных иерархических чисел в больших языковых моделях (экспериментальное исследование)

При проектировании программы, применяющей для вычисления семантического сходства десятичные иерархические числа, используется механизм проектора [15]. На поздних этапах кроме оценки с десятичными иерархическими индексами используется также косинусное сходство. В качестве предметной области выбран военный конфликт на Ближнем Востоке. Фрагмент семантической сети (граф знаний) для этой предметной области представлен на рисунке 1. Для наименований сущностей здесь использован английский язык, поскольку в практических разработках автора анализу на токсичность и фейк подвергаются зарубежные электронные издания (msnbc, bloomberg, cnn, springer, nbcnew, thrguardian, nytimes, france24) и отечественные издания на английском языке (RT, Meduza, Kremlin, Globalaffairs, Themoscowtimes, RussiaBeyond, Rossiyasegodnya, Interfax, SputnikInternational).

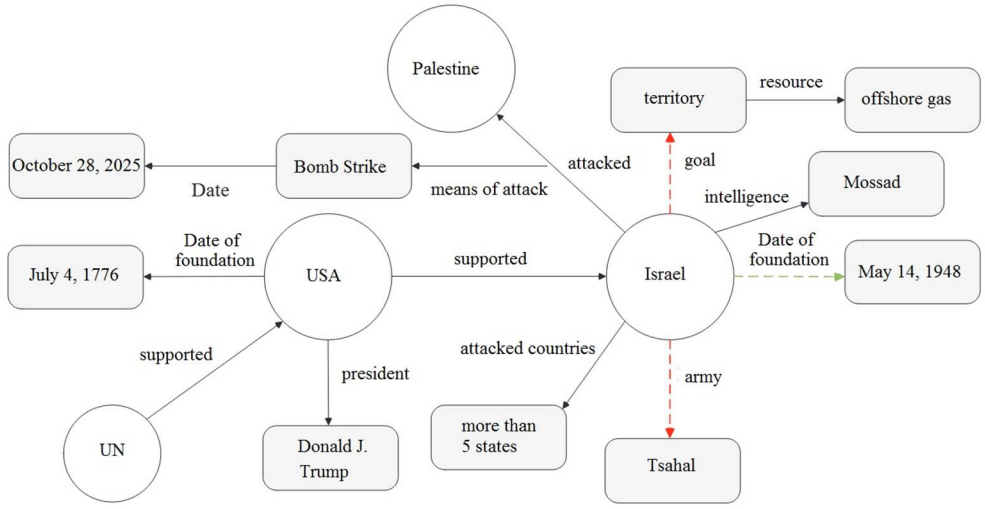

Рисунок 1 – Семантическая сеть в LLM
Figure 1 – Semantic network in LLM

Эту семантическую сеть можно использовать как основу для проектирования конкретных контекстных графов знаний с идентификацией основных сущностей десятичными иерархическими числами. Примерный фрагмент такого графа знаний с использованием родовидовой таксономии приведен на рисунке 2. Далее для удобства понимания отечественными специалистами в упрощенных примерах использован русский язык.

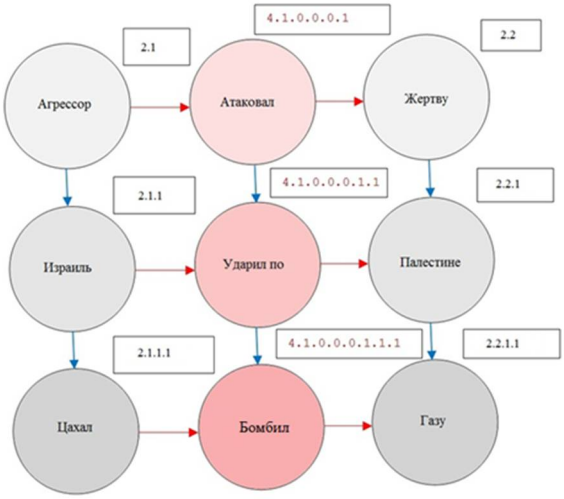

Рисунок 2 – Фрагмент контекстного графа знаний
Figure 2 – Fragment of contextual knowledge graph

Главный модуль примерной программы вычисления семантического сходства естественноязыковых конструкций на языке программирования Python v.3.0 представлен далее.

```
# Подготовка данных с новыми иерархическими ID и полными предложениями
data_items = [
    # Концептуальные сущности с их иерархией
    {"text": "Событие", "id": "4.1.0.0.0"},
    {"text": "атака", "id": "4.1.0.0.0.1"},
    {"text": "удар", "id": "4.1.0.0.0.1.1"},
    {"text": "бомбардировка", "id": "4.1.0.0.0.1.1.1"},
    {"text": "Агрессор", "id": "2.1"},
    {"text": "Израиль", "id": "2.1.1"},
    {"text": "Цахал", "id": "2.1.1.1"},
    {"text": "Жертва", "id": "2.2"},
    {"text": "Палестина", "id": "2.2.1"},
    {"text": "Газа", "id": "2.2.1.1"},
    # Полные предложения из датасета
    {"text": "Агрессор атаковал жертву.", "id": "full_sentence_A"},
    {"text": "Израиль нанес удар по Палестине.", "id": "full_sentence_B"},
    {"text": "Цахал разбомбил Газу.", "id": "full_sentence_C"},
]
# Создание иерархического проектора
class HierarchyProjector(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(HierarchyProjector, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(input_dim, output_dim),
            nn.ReLU(),
            nn.Linear(output_dim, output_dim)
        )
    def forward(self, x):
        return self.net(x)
input_dim = embeddings.shape[1]  # Размерность MiniLM - 384
output_dim = 128 # Проекция в 128-мерное пространство
projector = HierarchyProjector(input_dim, output_dim)
optimizer = optim.Adam(projector.parameters(), lr=0.005)
# Определение пар родитель-потомок для функции потерь(для иерархических ID)
parent_child_pairs_indices = []
for child_item in data_items:
    if not child_item['id'].startswith("full_sentence_"):
        child_id_parts = child_item['id'].split('.')
        if len(child_id_parts) > 1: # Если ID более чем из одной части
            parent_id = '.'.join(child_id_parts[:-1])
            if parent_id in id_to_idx: # Существует ли такой родитель
                child_idx = id_to_idx[child_item['id']]
                parent_idx = id_to_idx[parent_id]
                parent_child_pairs_indices.append((child_idx, parent_idx))
print(f"Найденные пары родитель-потомок для обучения
                  ({len(parent_child_pairs_indices)}):")
for child_idx, parent_idx in parent_child_pairs_indices:
    print(f"  Родитель: '{sentences[parent_idx]}' (ID: {da-
ta_items[parent_idx]['id']}) "
          f"-> Потомок: '{sentences[child_idx]}' (ID: {da-
ta_items[child_idx]['id']})")
# Функция потерь для иерархии
def hierarchical_loss(projected_embeddings, pairs):
    total_loss = 0.0
    for child_idx, parent_idx in pairs:
        child_embedding = projected_embeddings[child_idx]
        parent_embedding = projected_embeddings[parent_idx]
        # MSE loss для минимизации расстояния между родителями и детьми
        total_loss += torch.nn.functional.mse_loss(child_embedding,
             parent_embedding)
```

```
    return total_loss
# Цикл обучения
num_epochs = 200
for epoch in range(num_epochs):
    optimizer.zero_grad()
    projected = projector(embeddings)
    loss = hierarchical_loss(projected, parent_child_pairs_indices)
    loss.backward()
    optimizer.step()
#Получение финальных проецированных эмбеддингов
final_embeddings = projector(embeddings).detach().cpu().numpy()
#Функция для ответа на вопрос
def answer_question(question_text, projector_model, base_sbert_model,
                    all_projected_embeddings, all_texts, text_to_idx_map):
    # Векторизация и проецирование вопроса
    question_base_embedding = base_sbert_model.encode(question_text, con-
vert_to_tensor=True).detach().clone()
    projected_question_embedding = projec-
tor_model(question_base_embedding).squeeze().detach().cpu().numpy()
    # Определение целевых событийных предложений для сравнения
    event_sentences_texts = [
        "Агрессор атаковал жертву.",
        "Израиль нанес удар по Палестине.",
        "Цахал разбомбил Газу."
    ]
    # Получение индексов в общем списке 'all_texts'
    event_indices = [text_to_idx_map[txt] for txt in event_sentences_texts]
    # Получение эмбеддингов
    projected_event_embeddings = all_projected_embeddings[event_indices]
    # Вычисление сходства между вопросом и событиями
    similarities = []
    for i, event_emb in enumerate(projected_event_embeddings):
        sim =  dhn_similarity(data_items)
              *sk_cosine_similarity(projected_question_embedding.reshape(1,
           -1), event_emb.reshape(1, -1))[0][0]
        similarities.append((sim, event_sentences_texts[i]))
    similarities.sort(key=lambda x: x[0], reverse=True)
    most_similar_event_sentence = similarities[0][1]
# Основная часть для ответа на вопрос
question = "Кто был агрессором при атаке на Палестину?"
answer = answer_question(question, projector, base_model, final_embeddings, sen-
tences, text_to_idx)
print(f"\nВопрос: '{question}'")
print(f"Ответ: {answer}")
#  Дополнительные тесты
print("\n Дополнительные вопросы")
question_2 = "Кто бомбил Газу?"
answer_2 = answer_question(question_2, projector, base_model, final_embeddings,
sentences, text_to_idx)
print(f"\nВопрос: '{question_2}'")
print(f"Ответ: {answer_2}")
# Other question …
```

Результат вычисления

```
Найденные пары родитель-потомок для обучения (7):
  Родитель: 'Событие' (ID: 4.1.0.0.0) -> Потомок: 'атака' (ID: 4.1.0.0.0.1)
  Родитель: 'атака' (ID: 4.1.0.0.0.1) -> Потомок: 'удар' (ID: 4.1.0.0.0.1.1)
  Родитель: 'удар' (ID: 4.1.0.0.0.1.1) -> Потомок: 'бомбардировка' (ID:
4.1.0.0.0.1.1.1)
  Родитель: 'Агрессор' (ID: 2.1) -> Потомок: 'Израиль' (ID: 2.1.1)
  Родитель: 'Израиль' (ID: 2.1.1) -> Потомок: 'Цахал'  (ID: 2.1.1.1)
  Родитель: 'Жертва' (ID: 2.0) -> Потомок: 'Палестина' (ID: 2.0.1)
```

```
  Родитель: 'Палестина' (ID: 2.0.1) -> Потомок: 'Газа' (ID: 2.0.1.1)

Начинаем обучение проектора...
Epoch 0, Loss: 0.240950
Epoch 20, Loss: 0.120300
Epoch 40, Loss: 0.120021
Epoch 60, Loss: 0.092208
Epoch 80, Loss: 0.083552
Epoch 100, Loss: 0.059947
Epoch 120, Loss: 0.031580
Epoch 140, Loss: 0.010020
Epoch 160, Loss: 0.000000
Epoch 180, Loss: 0.000000

Наиболее релевантное предложение для вопроса 'Кто был агрессором при атаке на
Палестину?':
  'Израиль нанес удар по Палестине.' (Сходство: 0.9977)
Вопрос: 'Кто был агрессором при атаке на Палестину?'
Ответ: Израиль
Дополнительные вопросы
Наиболее релевантное предложение для вопроса 'Кто бомбил Газу?':
  'Израиль нанес удар по Палестине.' (Сходство: 1.0000)
Вопрос: 'Кто бомбил Газу?'
Ответ: Израиль
Наиболее релевантное предложение для вопроса 'Кто атаковал жертву?':
  'Израиль нанес удар по Палестине.' (Сходство: 0.9897)
Вопрос: 'Кто атаковал жертву?'
Ответ: Израиль
```

Заключение

Произведенные эксперименты позволяют сделать вывод о выигрыше программы (mIYubert v.3), основанной на применении десятичных иерархических чисел в сравнении с вычислением классического косинусного сходства и известными моделями DistilBERT на корпусе для обучения Hugging Face Datasets [16]. Выигрыш по точности составляет примерно 4 %, выигрыш во времени – 60 – 80 %. При этом корректировка результатов осуществлялась в пользу DistilBERT как в сторону увеличения, так и в сторону уменьшения семантического сходства там, где это могло быть оправдано в рамках экспертного мнения.

Библиографический список

1. **Каширин И.Ю.** Формирование LLM-ориентированных ресурсов знаний на основе генерации дополненной информации // Вестник Рязанского государственного радиотехнического университета. 2025. № 94. С. 85-97. DOI: 10.21667/1995-4565-2025-94-85-97.

2. Технический комитет по стандартизации информационной безопасности (TC260). Руководство по реагированию на инциденты с генеративным ИИ (на китайском языке). [Electronic resource]. Update date: February 2025.URL:https://www.tc260.org.cn/upload/2025-09-15/1757913511246034883.pdf (date of application: 02.01.2026).

3. **Каширин И.Ю.** Иерархические числа для проектирования ICF-таксономий искусственного интеллекта // Вестник Рязанского государственного радиотехнического университета. 2020. № 71. С. 71-82.

4. Mohannad Elhamod, Kelly M. Diamond, A. Murat Maga et al. Hierarchy-guided neural network for species classification. British Ecological Society. Methods Ecol Evol. 2022, 642-652. DOI: 10.1111/2041-210X.13768.

5. Hierarchical Embeddings for Text Search. [Electronic resource]. Update date: 10.12.2024. URL: https://gwern.net/tree-embedding , (date of application: 02.01.2026).

6. **Gabor Petnehazi, Bernadett Aradi**. HERCULES: Hierarchical Embedding-based Recursive Clustering Using LLMs for Efficient Summarization. License: arXiv.org perpetual non-exclusive license arXiv: 2506.19992v1 [cs.LG] 24 Jun 2025.

7. Yu J., Zhang C., Hu Z., Ji Y. Embedding Hierarchical Tree Structure of Concepts in Knowledge Graph Embedding. Electronics, 2024, 13(22), 4486. doi.org/10.3390/electronics13224486.

8. **Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Łukasz, Polosukhin Illia** (December 2017). «Attention is All you Need» (PDF). In I. Guyon and U. Von Luxburg and S. Bengio and H. Wallach and R. Fergus and S. Vishwanathan and R. Garnett (ed.). 31st Conference on Neural Information Processing Systems (NIPS). Advances in Neural Information Processing Systems. Vol. 30. Curran Associates, Inc. arXiv:1706.03762.

9. **Каширин И.Ю**. Эмбеддинги иерархических чисел для обогащения трансформерных языковых моделей внешними онтологическими знаниями // Вестник Рязанского государственного радиотехнического университета. 2025. № 93. С. 72-82. DOI: 10.21667/1995-4565-2025-93-72-82.

10. **An B., Zhang S., Dredze M.** RAG LLMs are Not Safer: A Safety Analysis of Retrieval-Augmented Generation for Large Language Models. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (v.1: Long Papers, Albuquerque, New Mexico. Association for Computational Linguistics). 2025, pp. 5444-5474.

11. **M. Hu, X. Zhao, J. Wei, J. Wu**, et al. «rT5: A Retrieval-Augmented Pre-trained Model for Ancient Chinese Entity Description Generation» In International Conference on Natural Language Processing and Chinese Computing, Cham: Springer, 2023, pp. 736-748.

12. **Каширин И.Ю**. Извлечение фактов из естественно-языковых текстов методом унификации семантических паттернов // Вестник Рязанского государственного радиотехнического университета. 2025. № 91.С. 36-49. DOI: 10.21667/1995-4565-2025-91-36-49.

13. **Yoshua Bengio, Jean-Sébastien Senécal.** Quick training of probabilistic neural nets by importance sampling. In Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, 2003, pp. 17-24.

14. **Guy Blanc, Steffen Rendle**. Adaptive sampled softmax with kernel based sampling. In International Conference on Machine Learning, pages, 2018, pp. 590-599.

15. **Junbum Cha, Wooyoung Kang, Jonghwan Mun, Byungseok RohHoneybee**. Locality-Enhanced Projector for Multimodal LLM. 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024, pp. 13817-13827. DOI: 10.1109/CVPR52733.2024.01311.

16. Hugging Face Datasets. [Electronic resource]. Update date: 20.12.2025, URL: https://huggingface.co/datasets (date of application: 06.01.2026).

UDC 007:681.512.2

DISTINCTIVE FEATURES OF DECIMAL HIERARCHICAL NUMBERS IN LARGE LANGUAGE MODEL EMBEDDINGS INTERPRETATION

I. Yu. Kashirin, Doctor in technical sciences, Professor, Department of computing and applied mathematics, RSREU, Ryazan, Russia;
orcid.org/0000-0003-1694-7410, e-mail: igor-kashirin@mail.ru

This paper discusses a new method for analyzing input natural language sentences for LLM language models. A new method is based on the algebra of decimal hierarchical numbers used in algorithms for calculating the semantic similarity of words, phrases, and sentences. The method is suitable for local subject areas and has been tested in «political news» subject area. For this local domain, an OWL ontology and a corresponding graphical representation in the form of a semantic network were develope, with basic entities marked up with decimal hierarchical numbers. A semantic network includes general and application level. A fragment of general ontology is represented by relation, which significantly reduce computational complexity of algebraic operations on knowledge graphs and, consequently, reduce the time required to calculate the semantic similarity of natural language constructs.

Software implementation of the method under consideration uses well-known DistilBERT technology for language neural networks with attentional focus. Knowledge enrichment of pre-trained neural network is achieved by generating new semantic embeddings for words (entities) of natural

language sentences and integrating them into a new neural network before fine-tuning in local domain. Training corpora for a new neural network model mIYu-bert v.2.0 were a general corpus from Hugging Face Datasets repository and a local corpus of materials extracted by the author from English-language political articles from international electronic media outlets, including RT, Meduza, CNN, TASS, NYTimes, Bloomberg, and WSJ.

The experimental portion of the material is based on Python v.3 (Anaconda 3) programming language toolkit, LLM DistilBERT, and mIYu-bert v.3.1 software package. The latter toolkit was implemented by the author.

The completed series of experiments allows us to qualify a new method of using decimal hierarchical numbers in the retraining of LLM models for calculating semantic similarity as the basis for the technology that is equal in efficiency to currently available international analogues and does not exceed them in computational complexity.

***The aim of this paper** is to describe a new method to calculate semantic similarity in LLM language models using decimal hierarchical numbers based on OWL ontologies, as well as universal algebras for generating knowledge graphs in local subject areas.*

***Keywords**: decimal hierarchical numbers, universal algebras, DistilBERT language models, natural language analysis, ontological taxonomies, semantic similarity.*

DOI: 10.21667/1995-4565-2026-95-117-129

References

1. **Kashirin I.Yu.** Formirovanie LLM-orientirovannykh resursov znanij na osnove gene-racii dopolnennoj informacii. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2025, no. 94, pp. 85-97. DOI: 10.21667/1995-4565-2025-94-85-97.

2. Tekhnicheskij komitet po standartizacii informacionnoj bezopasnosti (TC260). Rukovodstvo po reagirovaniyu na incidenty s generativnym II (na kitajskom yazyke). [*Electronic resource*]. Update date: February 2025.URL:https://www.tc260.org.cn/upload/2025-09-15/1757913511246034883.pdf (date of application: 02.01.2026).

3. **Kashirin I.Yu**. Ierarkhicheskie chisla dlya proektirovaniya ICF-taksonomij iskusstvennogo intellekta. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2020, no. 71. pp. 71-82.

4. **Mohannad Elhamod, Kelly M. Diamond, A. Murat Maga** et al. Hierarchy-guided neural network for species classification. *British Ecological Society*. Methods Ecol Evol. 2022, 642-652. DOI: 10.1111/2041-210X.13768.

5. Hierarchical Embeddings for Text Search. [*Electronic resource*]. Update date: 10.12.2024. URL: https://gwern.net/tree-embedding , (date of application: 02.01.2026).

6. **Gabor Petnehazi, Bernadett Aradi**. HERCULES: Hierarchical Embedding-based Recursive Clustering Using LLMs for Efficient Summarization. License: *arXiv.org perpetual non-exclusive license arXiv*:2506.19992v1 [cs.LG] 24 Jun 2025.

7. **Yu J., Zhang C., Hu Z., Ji Y.** Embedding Hierarchical Tree Structure of Concepts in Knowledge Graph Embedding. *Electronics*. 2024, 13(22), 4486. doi.org/10.3390/electronics13224486.

8. **Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Łukasz, Polosukhin Illia** (December 2017). «Attention is All you Need» (PDF). In I. Guyon and U. Von Luxburg and S. Bengio and H. Wallach and R. Fergus and S. Vishwanathan and R. Garnett (ed.). *31st Conference on Neural Information Processing Systems (NIPS)*. Advances in Neural Information Processing Systems. Vol. 30. Curran Associates, Inc. arXiv:1706.03762.

9. **Kashirin I.Yu.** Ehmbeddingi ierarkhicheskikh chisel dlya obogashcheniya transformernykh yazykovykh modelej vneshnimi ontologicheskimi znaniyami. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta.* 2025, no. 93, pp. 72-82. DOI: 10.21667/1995-4565-2025-93-72-82.

10. **An B., Zhang S., Dredze M**. RAG LLMs are Not Safer: A Safety Analysis of Retrieval-Augmented Generation for Large Language Models. *In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies* (v.1: Long Papers, Albuquerque, New Mexico. Association for Computational Linguistics), 2025, pp. 5444-5474.

11. **Hu M., Zhao X., Wei J., Wu J.**, et al. «rT5: A Retrieval-Augmented Pre-trained Model for Ancient Chinese Entity Description Generation» *In International Conference on Natural Language Processing and Chinese Computing*, Cham: Springer, 2023, pp. 736-748.

12. **Kashirin I.Yu**. Izvlechenie faktov iz estestvenno-yazykovykh tekstov metodom unifikacii semanticheskikh patternov. *Vestnik Ryazanskogo gosudarstvennogo radiotekh-nicheskogo universiteta*. 2025, no. 91, pp. 36-49. DOI: 10.21667/1995-4565-2025-91-36-49.

13. **Yoshua Bengio and Jean-Sébastien Senécal**. Quick training of probabilistic neural nets by importance sampling. *In Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*. 2003, pp. 17-24.

14. **Guy Blanc and Steffen Rendle**. Adaptive sampled softmax with kernel based sampling. *In International Conference on Machine Learning*, pages, 2018, pp. 590-599.

15. **Junbum Cha, Wooyoung Kang, Jonghwan Mun, Byungseok RohHoneybee**. Locality-Enhanced Projector for Multimodal LLM. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 13817-13827. DOI: 10.1109/CVPR52733.2024.01311.

16. Hugging Face Datasets. [*Electronic resource*]. Update date: 20.12.2025, URL: https://huggingface.co/datasets (date of application: 06.01.2026).