Методы, технологии и приложения искусственного интеллекта

УДК 004.912
DOI:10.25729/ESI.2025.38.2.001

Объединение глубоких моделей и разреженных представлений в информационном поиске: обзор и анализ современных подходов

Абрамович Роман Константинович, Добрынин Вячеслав Юрьевич, Платонов Алексей Владимирович

Университет ИТМО,
Россия, Санкт-Петербург, *asmetliness24237@gmail.com*

Аннотация. Традиционные методы поиска, основанные на разреженных представлениях, характеризуются высокой производительностью, но ограниченным качеством в связи с неспособностью учитывать семантические связи в данных. С другой стороны, плотные векторные представления позволяют улучшить качество благодаря захвату семантических отношений в данных. Однако, эти методы сталкиваются с проблемами масштабируемости и требуют значительных вычислительных ресурсов. С развитием глубоких нейронных сетей, в том числе архитектур на основе трансформеров, появляется все больше работ, целью которых является объединить два подхода. Цель представленной обзорно-аналитической статьи – сравнить существующие работы, использующие глубокие модели для формирования разреженных представлений.

Ключевые слова: глубокие нейронные сети, семантический поиск, вычислительные ресурсы, разреженные представления, инвертированный индекс

Цитирование: Абрамович Р.К. Объединение глубоких моделей и разреженных представлений в информационном поиске: обзор и анализ современных подходов / Р.К. Абрамович, В.Ю. Добрынин, А.В. Платонов // Информационные и математические технологии в науке и управлении, 2025. – № 2(38). – С. 5-17. – DOI:10.25729/ESI.2025.38.2.001.

Введение. Современные поисковые системы функционируют по двухэтапной схеме (рис. 1). Задачей первого этапа является сканирование всего пространства документов и сокращение количества потенциальных кандидатов до десятков тысяч с помощью легковесных структур данных, таких, как обратный индекс. Обратным индексом называют структуру, в которой для каждого термина (токена) указывается список документов, где встречается данный термин, что позволяет быстро производить поиск. Он эффективен для быстрого поиска на огромных массивах данных, но обладает рядом недостатков. Поскольку индекс строится только на основе встречаемости отдельных терминов или n-грамм (последовательность из n элементов, в данном случае – терминов), он не способен уловить контекст и семантические отношения между словами, что снижает качество первичного ранжирования и приводит к проблеме несоответствия словарей запроса и документа. На втором и последующих этапах применяются более сложные и ресурсоемкие ре-ранжирующие модели, например, глубокие нейронные сети, основанные на архитектуре “Трансформер” [1], поскольку они позволяют учитывать глубокие контекстуальные связи в текстах, значительно улучшая качество ранжирования. Такие модели генерируют плотные векторные представления (dense embeddings), также называемые контекстуализированными представлениями/векторами. В отличие от традиционных разреженных векторов (где большинство координат равны нулю), плотные векторы содержат ненулевые значения во всех измерениях; кроме того, важным отличием является их размерность, которая на порядок ниже размерности разреженных векторов и позволяет компактно кодировать семантику. Эти векторы генерируются с использованием нейронных сетей, предварительно обученных на больших корпусах, и способны улавливать скрытые (контекстуальные) связи между словами, благодаря чему их применяют, чтобы преодолеть проблему несоответствия словарей между

запросом и документом, а также улучшить качество поиска за счет более глубокого учета смысла текста. Однако, разреженные представления и структуры данных, основанные на них (такие как обратный индекс), значительно превосходят плотные представления в скорости поиска, а также в требованиях к памяти.

При построении современных поисковых систем возникают несколько ключевых проблем. Во-первых, стремительный рост объемов данных вынуждает разработчиков заботиться о масштабировании и быстром доступе к информации, иначе сканирование десятков или сотен миллионов документов становится узким местом. Во-вторых, традиционные методики часто не учитывают контекст и могут отбрасывать синонимичные или грамматически изменённые формы слов, что приводит к неполному охвату релевантных результатов. В-третьих, точное сопоставление текстов усложняется из-за использования разных языков, терминологий и неоднозначностей в формулировках. Все эти сложности подталкивают к созданию гибридных решений, объединяющих преимущества разреженных индексов и контекстуализированных векторных моделей, где каждое слово представлено с учетом окружения.

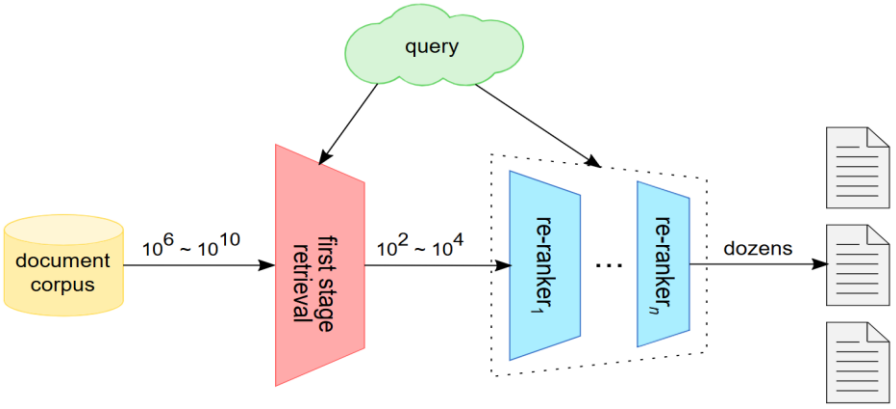

Рис. 1. Двухэтапная архитектура из легковесного первого этапа поиска (англ. first state retrieval) и последующего ре-ранжирования (англ. re-ranker), заимствовано из [2]

Недавние работы рассматривают возможность интеграции разреженных и плотных моделей в одной системе поиска, что может включать обучение глубоких моделей на генерацию разреженных векторов, трансформацию существующих плотных векторов в разреженные или использование плотных векторов для улучшения структуры обратного индекса уже на первом этапе. При таком подходе авторы пытаются объединить производительность традиционных разреженных моделей и глубокое понимание семантических связей плотных моделей, улучшая качество или производительность поиска.

Эффективность тех или иных методов оценки релевантности часто проверяется на крупных наборах данных. Одним из наиболее популярных является датасет MS MARCO, созданный компанией Microsoft и представляющий собой коллекцию пар «запрос – параграф», используемых при обучении и тестировании ранжирующих алгоритмов. На нем, как правило, рассчитывают метрику MRR@10 (Mean Reciprocal Rank). Данная метрика для каждого запроса находит первый релевантный документ в топ-10 результатах итоговой выборки, считает для данного документа его обратный ранг, после чего усредняет полученные значения для всех запросов. Таким образом, оценивается, насколько высоко система поднимает корректный ответ при ранжировании. Формула вычисления MRR@10 и обратного ранга представлена ниже:

$$MRR@10 = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{r_i}$$

где Q - множество всех запросов, r_i - позиция первого релевантного документа для запроса i в итоговой топ 10 выборке (так, если релевантный документ был 3 по счету из 10, то его позиция будет равна 3), $\frac{1}{r_i}$ - обратный ранг первого релевантного документа для запроса i. Следует отметить, что если в топ-10 кандидатах не встретится ни одного релевантного документа, значение обратного ранга выставляется в 0.

В данной работе мы представляем обзор основных публикаций и методов, направленных на интеграцию этих подходов, рассматривая принципы их функционирования и результаты. Особое внимание уделяется рассмотрению возникающих при этом проблем: корректному учету контекста, оптимизации хранения и извлечения информации, что позволит глубже понять текущее состояние исследований и потенциальные векторы развития гибридных моделей поиска.

1. SNRM. Работа "From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing" [3], представленная Хамедом Замани и соавторами в 2018 году, была одной из первых работ, предложивших модель ранжирования, объединяющую преимущества разреженных и плотных векторных представлений с сохранением эффективности поиска.

Модель SNRM (Standalone neural ranking model), представленная авторами, работает по принципу обучения модели генерировать разреженные векторные представления для запроса и документа, которые в дальнейшем могут быть использованы для построения и поиска по обратному индексу.

Представления запросов и документов генерируются с использованием глубоких нейронных сетей, при этом для кодирования как запроса, так и документа используется одна и та же архитектура с общими параметрами.

Для генерации разреженного представления документа эта модель сначала отдельно генерирует разреженное представления для каждой последовательной n-граммы токенов, после чего агрегирует их вместе, используя average pooling (усреднение значений в векторе по выбранным осям). В математическом виде процесс получения итогового вектора может быть выражен следующей формулой:

$$\psi(d) = \frac{1}{|d|-n+1}\sum_{i=1}^{|d|-n+1}\psi_{ngram}(t_i, t_{i+1}, \dots, t_{i+n-1}),$$

где d - документ, для которого генерируется вектор, t_1, t_2, …, $t_{|d|}$ – токены данного документа с сохранением порядка, ψ_{ngram} – полносвязная нейронная сеть, работающая по принципу, похожему на работу автокодировщика (autoencoder) [4], способного сжимать данные в компактное представление, а затем восстанавливать их. Однако, в отличии от класического автокодировщика, во время обучения модели авторы используют дополнительную функцию потерь в виде L1 регуляризации для разреживания итоговых векторов:

$$L_1(x) = \sum_{i=1}^{|x|}|x_i|$$

где x - вектор, подающийся на вход функции регуляризации, i - индекс элемента вектора x.

Использование такого подхода позволяет учесть длину входного документа и генерировать более разреженные представления для коротких документов и запросов, а также сохранить локальный контекст, в котором встречается каждый отдельный токен t_i.

За счет разреженности полученных векторов эта модель может быть эффективно использована для построения обратного индекса, что позволяет перейти к одноступенчатой архитектуре поиска.

Процесс построения обратного индекса данной модели строится на основе “скрытых” (англ. latent) токенов. Скрытые токены в контексте модели SNRM – это индексы компонентов итогового разреженного вектора, из чего следует, что количество скрытых токенов ограничено размерностью итогового вектора, т.е. если итоговый вектор будет размерности 5000, в обратном индексе будет 5000 скрытых токенов. Из этого также следует, что скрытые токены SNRM не совпадают напрямую с реальными токенами оригинального текста, что затрудняет их интерпретацию.

Сам же процесс индексации состоит в том, что каждый документ проходит через модель, получая разреженное представление данного документа. Затем, для каждого индекса данного векторного представления определяется: если *i*-ый элемент не равен нулю, значит, данный документ содержит скрытый токен *i* и должен быть добавлен в обратный индекс для этого токена. В качестве оценки релевантности при этом будет взято значение вектора по этому индексу.

Процесс поиска заключается в получении разреженного вектора для запроса, определения всех скрытых токенов (ненулевых компонент полученного вектора), находящихся в данном векторе, использовании обратного индекса для извлечения всех документов, содержащих скрытые токены запроса, и последующего вычисления релевантности. В качестве функции для оценки релевантности используется скалярное произведение между векторами запроса и документа.

Используя данный подход, авторам удалось добиться значительного улучшения качества поиска по сравнению с классическими методами ранжирования на обратном индексе, такими, как BM25 [5].

Однако, данная модель обладает рядом недостатков:

1. Потеря интерпретируемости токенов – поскольку для построения обратного индекса мы используем скрытые токены, полученные после генерации разреженного представления, мы не имеем возможности сопоставить их с оригинальным словарем и интерпретировать.
2. Потребность вычисления модели во время поиска – поскольку для обращения к обратному индексу в данной модели нам требуются скрытые токены запроса, запрос требуется пропускать через модель, что значительно повышает количество используемых ресурсов на этапе поиска по сравнению с BM25.
3. Статически заданное количество скрытых токенов – размерность результирующего разреженного вектора, генерируемого моделью, и, как следствие, количество скрытых токенов, доступных для использования в обратном индексе, жестко задается на этапе обучения модели, делая процесс его изменения труднореализуемым.

2. SparTerm. Еще одной значимой работой, имевшей значительное влияние на последующие исследования, стала работа "SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval" (SparTerm) [6], представленная Яном Байем и соавторами в 2020 году.

В своей работе авторы исследуют проблему улучшения традиционных разреженных представлений на основе терминов, используя глубокие языковые модели, такие, как BERT [7]. Модель SparTerm, представленная в работе, направлена на улучшение семантического соответствия оценок тематике документа в обратном индексе, сохраняя при этом преимущества эффективности, интерпретируемости и точного соответствия терминов.

Данная модель состоит из двух компонентов (рис. 2):

1. Предиктор важности (англ. importance predictor) – генерирует плотное векторное представление, отражающее семантическую важность для каждого термина в словаре,

основанное на контекстуальных представлениях, полученных с помощью предобученной языковой модели.

2. Контроллер активации (англ. gating controller) – создает бинарный массив той же размерности, что и векторное представление, определяющий, какие из элементов векторного представления попадут в итоговый разреженный вектор.

Контроллер активации при этом может иметь 2 режима работы:

1. Literal-only gating – в итоговое разреженное представление включаются только токены, которые изначально присутствовали в документе.
2. Expansion-enhanced gating – контроллер пытается дополнительно выполнить расширение запроса и документа, добавляя в них новые термины, близкие по смыслу тем, которые уже есть в документе.

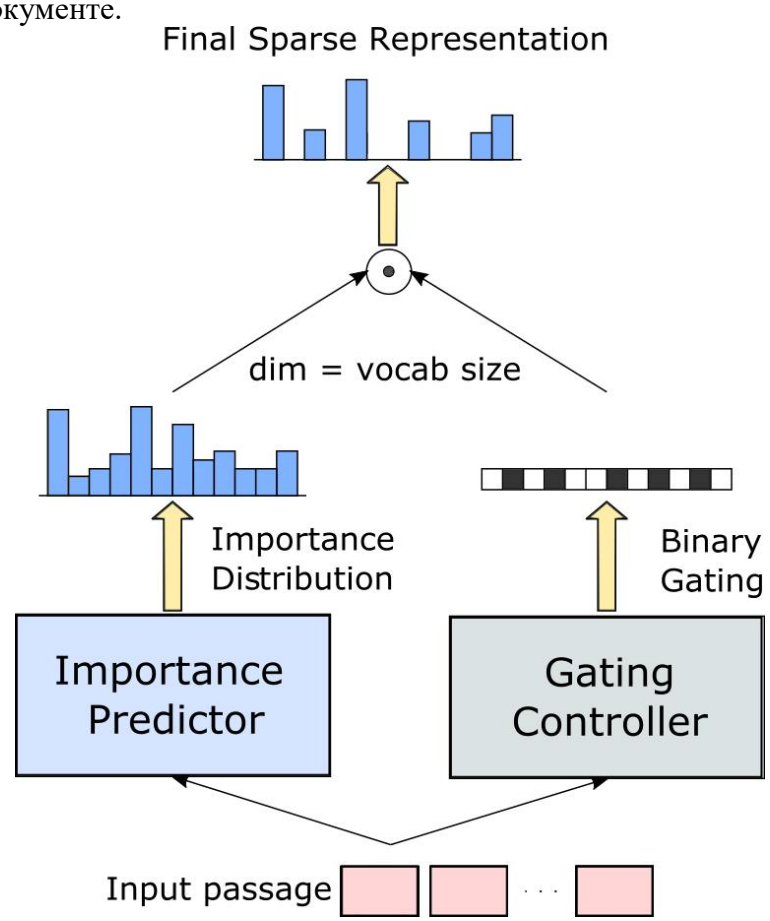

Рис. 2. Взаимодействие Предиктора Важности (англ. Importance Predictor) и Контроллера Активации (англ. Gating Controller) в архитектуре SparTern (заимствовано из [5])

Таким образом, помимо проблемы с потерей учета контекста, данная модель может решить проблему несовпадения словарей запроса и ответа.

Предсказатель важности использует BERT для получения контекстуализированных векторных представлений для каждого термина документа. Затем каждое векторное представление конкретного токена трансформируется в представление относительно всего словаря с использованием матричных трансформаций.

Так, для конкретного документа или последовательности токенов $t_1, t_2 \dots t_n$ и соответствующих им векторных представлений $h_1, h_2 \dots h_n$, полученных с помощью BERT, оценка важности w_{ij} каждого входного токена i относительно каждого токена в словаре j вычисляется по формуле:

$$w_{ij} = Transform(h_i)^T E_j + b,$$

где E_j – плотное представление токена j в словаре, полученное с помощью BERT, h_i – плотное векторное представление конкретного токена во входящем документе или последовательности, b – настраиваемый коэффициент смещения, Transform – линейный слой с активацией Gaussian Error Linear Unit [8] и слоем нормализации (англ. Layer Norm) [9].

Итоговое векторное представление $w = [w_0, w_1, \dots w_n]$, где n – количество токенов в словаре, получается путем суммирования оценок важности каждого токена из словаря относительно токенов входящей последовательности после применения функции активации ReLU, обеспечивающей положительность весов терминов:

$$w_j = \sum_{i=0}^{L} ReLU(w_{ij}),$$

где $ReLU(x) = max(0, x)$, w_j – итоговая оценка важности словарного токена j относительно входного документа, L - количество токенов входного документа.

Итоговое разреженное векторное представление, используемое для построения обратного индекса, определяется формулой:

$$p' = F(p) \odot G(p),$$

где $F(p)$ – плотный вектор, полученный предиктором важности, $G(p)$ – бинарный вектор, полученный контроллером активации, а символ $\odot$ означает покоординатное (поэлементное) умножение соответствующих компонентов векторов.

На тестовых данных авторы получили значительный прирост качества по сравнению с классическими моделями: MRR@10=0.29 на датасете MS MARCO, по сравнению с MRR@10=0.184 у BM25.

Основным преимуществом данной модели по сравнению с похожими работами является полное сохранение исходных терминов, позволяющее сохранить свойство интерпретируемости результатов модели.

Ограничением же является то, что SparTerm с использованием expansion-enhanced gating не может быть обучена от начала до конца (end-to-end), то есть все её компоненты не оптимизируются совместно в рамках одного процесса обучения. Вместо этого один из её ключевых элементов — контроллер активации — обучается заранее и фиксируется во время обучения предиктора важности, что ограничивает возможности модели в адаптации к конкретной задаче. Это не позволяет модели изучить оптимальную стратегию разреженности — то есть самостоятельно определить, какие термины и в каком количестве должны участвовать в поиске, чтобы добиться наилучших результатов ранжирования в зависимости от конкретной задачи.

3. SPLADE. Развитием SparTerm является модель SPLADE [10], представленная Тибо Формалем и соавторами в 2021 году. Эта модель упрощает архитектуру SparTerm, убирая необходимость в использовании контроллера активации для разреживания итогового вектора.

Относительно модели SparTerm, основное изменение заключается в изменении логики расчета итоговой важности каждого токена в словаре относительно входного документа. За счет добавления эффекта логарифмической насыщенности авторам удается предотвратить доминирование отдельных терминов и обеспечить разреженность итогового вектора без использования отдельного контроллера активации.

Итоговая формула расчета важности каждого токена в словаре при новом подходе выглядит следующим образом:

$$w_j = \sum_{i \in t} log\left(1 + ReLU(w_{ij})\right),$$

где $ReLU(x) = max(0, x)$, w_{ij} – оценка важности токена словаря j относительно токена входного документа i, w_j – итоговая оценка важности токена словаря j относительно всего документа.

Помимо эффекта логарифмической насыщенности, разреженность итоговых векторов обеспечивается дополнительной функцией потерь при обучении модели, по аналогии с работой SNRM. Однако, в качестве функции потерь авторы SPLADE использовали FLOPS [11] регуляризатор, понижающий количество вычислений с плавающей точкой, необходимое для вычисления вектора документа. По результатам тестов, использование FLOPS позволяет строить более сбалансированный, и, как следствие, более эффективный поисковый индекс.

Благодаря пересмотру архитектуры, авторам удалось добиться MRR@10=0.322 на датасете MS MARCO, улучшив итоговое качество поиска по сравнению со SparTerm (MRR@10=0.29 на тестовых данных).

4. ColBERT, ColBERTv2. В отличии от ранее рассмотренных работ авторы ColBERT [12] не пытаются генерировать разреженные представления для построения индекса и ускорения поиска. Вместо этого авторы используют ряд алгоритмических оптимизаций для сокращения количества ресурсов, необходимых для осуществления процесса поиска.

Основными подходами к оптимизации при этом являются: использование методов приближенного поиска соседей (ANN) с использованием библиотеки faiss [13], позволяющей значительно ускорить поиск схожих векторов на больших объемах данных; PQ-преобразования [14] для сжатия векторных представлений и сокращения потребляемой памяти; использование введенной авторами концепции «позднее взаимодействие» (англ. late interaction), позволяющей отделить процесс получения контекстных векторов запросов и документов от расчета оценки релевантности между ними.

PQ-преобразование (product quantization) - метод сжатия с потерями, используемый для сжатия плотных векторных представлений. Данный метод разбивает исходный вектор на заданное количество под-векторов как представлено на рисунке 3. Затем, под-векторы, находящиеся на одинаковых позициях в оригинальных векторах (например, под-вектора на позиции 3) кластеризуются друг с другом, в результате чего, для каждой позиции подвектора получается набор цетроидов - векторов, находящихся ровно в середине рассматриваемых кластеров. Далее, каждый под-вектор в оригинальном векторе заменяется на индекс ближайшего к нему центроида. Полученные сжатые векторы, в которых под-векторы заменены на индексы центроидов под-векторных кластеров, занимают значительно меньше памяти и могут быть использованы чтобы приблизительно восстановить оригинальный вектор путем обратной замены индекса на центроид.

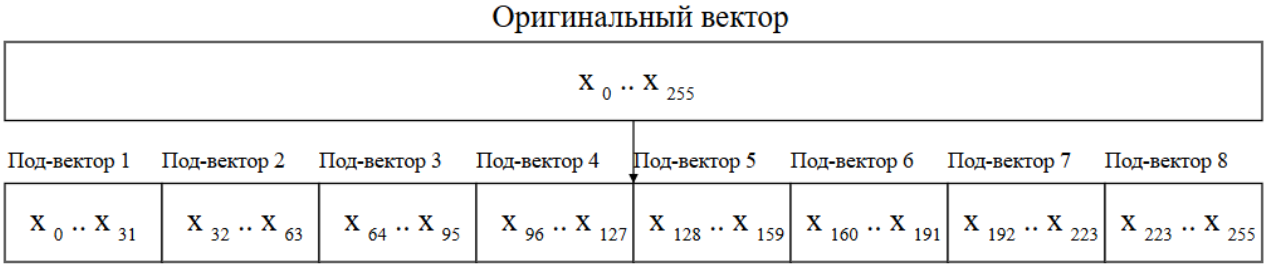

Рис. 3. Разбиение на подвектора для PQ

Использование позднего взаимодействия позволяет вынести в оффлайн процесс индексирования документов путем получения для них векторного представления, его сжатия с использованием PQ и сохранения в векторный индекс faiss для последующего поиска.

Процесс обработки запроса состоит из нескольких этапов. Сначала, для каждого входного токена запроса вычисляется множество его контекстуализированных векторов E_q. Далее, для каждого полученного вектора параллельно отправляется запрос в векторный индекс faiss для извлечения векторных представлений заранее проиндексированных документов кандидатов.

Финальным этапом является расчет итоговой оценки релевантности всех документов, найденных в векторном индексе, с запросом пользователя. Итоговая оценка при этом рассчитывается с использованием определенной авторами функцией MaxSim, которая выполняет попарное сравнение схожести каждого векторного представления запроса и документа, и выбирает максимум:

$$S_{q,d} \coloneqq \sum_{i \in [|E_q|], j \in [|E_d|]} max(E_{q_i} \cdot E_{d_j}^T),$$

где E_q – множество контекстуализированных векторов запроса, E_d – множество контекстуализированных векторов документа, q_i - контекстуализированный вектор запроса под индексом i из множества E_q, d_j - контекстуализированный вектор документа под индексом j из множества E_d, $S_{q,d}$ – итоговая оценка релевантности между запросом q и документом d. Данный процесс представлен на рис. 4:

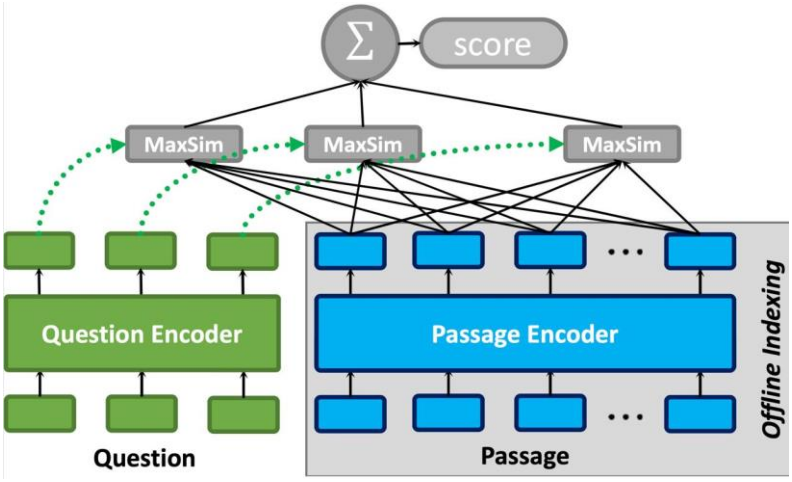

Рис. 4. Оценка релевантности (англ. score) между запросом (англ. question) и документом (англ. passage) с использованием функции MaxSim (заимствовано из [11])

Эта модель показывает высокое качества поиска (MRR@10=0.36 по сравнению с 0.184 у BM25 на датасете MS MARCO), однако требует намного больше ресурсов для вычисления оценки релевантности между документом и запросом, а также требует большего дискового пространства для хранения векторных представлений документов.

В дальнейшем авторы этого подхода развивают свою идею в модели ColBERTv2 [15], внедряя новую, улучшенную схему кодирования векторов, работающую в несколько этапов. На первом этапе происходит кластеризация индексируемых векторов, в результате которой получается заданное количество векторных кластеров. Далее, для каждого кластера вычисляется центроид - вектор, находящийся ровно в середине рассматриваемого кластера, при этом каждому центроиду присваивается порядковый индекс. Следующим шагом, для каждого индексируемого вектора вычисляется остаточный (англ. residual) вектор r, представляющий из себя разность вектора и ближайшего к нему центроиду:

$$r = v - \mathrm{C}_t$$

где v - индексируемый вектор, C_t - ближайший центроид к этому вектору.

Далее, остаточные вектора r сжимаются с использованием PQ-преобразования для сокращения занимаемой памяти, группируются по индексу центроида C, использованного для получения остаточного вектора, и сохраняются на диск вместе с самими центроидами.

Благодаря такому подходу размер хранимых векторов сокращается с 256 байт до 20 или 36 байт в зависимости от настроек PQ, что значительно снижает количество потребляемой памяти. Более того, хранение векторов в разрезе центроидов, к которым они принадлежат, позволяет дополнительно сократить время поиска, поскольку позволяет сразу отбросить

вектора и центроиды далекие от векторов входного запроса. При этом, для оценки релевантности во время поиска, оригинальные вектора восстанавливаются по формуле:

$$\underline{v} = C_t + \underline{r}$$

где $\underline{v}$ - приближенное восстановленное значение оригинального вектора, $\underline{r}$ - сжатый через PQ остаточный вектор, C_t - центроид, использованный для получения остаточного вектора.

Новая схема кодирования остатков позволяет эффективно сжать векторы без изменения архитектуры самой модели или ее обучения, сохраняя при этом высокую точность поиска: MRR@10 = 0.397 на датасете MS MARCO.

Однако, несмотря на сокращение требуемого объема памяти для хранения векторов и оптимизации в виде поиска только в ближайших центроидов, вторая версия модели все еще страдает от необходимости проведения тяжелых вычислительных операций на этапе поиска для получения векторного представления запроса и поиска по векторному индексу.

5. SparseEmbed. Модель SparseEmbed [16], представленная в 2023 году, является комбинацией идей моделей SPLADE и ColBERT. Эта модель работает в 2 этапа. На первом этапе, с использованием в качестве основы модели SPLADE, для документа генерируется разреженный вектор размерностью словаря, а также сохраняются плотные векторы для каждого токена документа или запроса.

На втором этапе, используя плотные векторы активированных токенов в качестве весов внимания (attention scores) - параметров глубокой нейронной сети, которые отражают важность каждого токена и позволяют более точно учитывать контекст и значимость отдельных терминов при генерации векторных представлений для поиска релевантных документов, на базе запроса или документа генерируются плотные векторные представления для всех активированных токенов разреженного вектора (результата первого этапа).

Таким образом, данная модель не использует разреженный вектор напрямую для поиска или для построения обратного индекса. Вместо этого разреженный вектор служит индикатором, для каких терминов из словаря следует сгенерировать плотный вектор для оценки релевантности. Данный подход представлен на рис. 5:

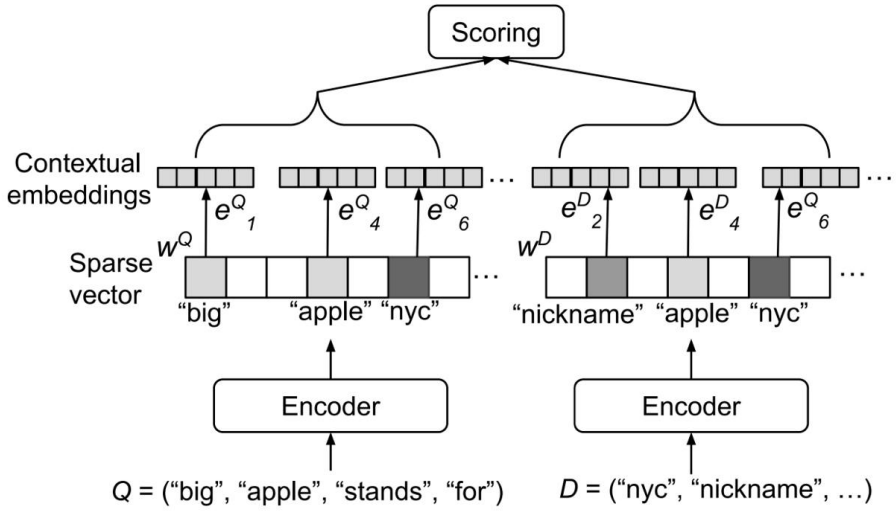

Рис. 5. Использование разреженных векторов (англ. sparse vector) и контекстуализированных векторов (англ. contextualized embeddings) для расчета релевантности в модели SparseEmbed (заимствовано из [15])

В качестве оценки релевантности используется сумма векторных произведений плотных векторов терминов, совпадающих в запросе и документе:

$$s(q,d) = \sum_{(i,j)\in I^q \times I^d, i=j} \left(e_i^q\right)^T e_j^d,$$

где индексы q и d – сравниваемые запрос и документ, e_i^q и e_j^d – плотные представления пересекающихся активированных терминов, для которых выполняется скалярное произведение, I^q и I^d – множество активированных (присутствующих как в запросе, так и в документе) терминов запроса и документа.

Преимуществом данной модели по сравнению с SPLADE является более точный захват контекста за счет использования плотных векторных представлений. По сравнению с ColBERT, оценка релевантности SparseEmbed является более производительной, поскольку требует линейного времени относительно количества активированных терминов, тогда как ColBERT требует квадратичного времени для сравнения всех векторов запроса и документа между собой.

Авторы работы получили оценку качества в 0.392 для датасета MS MARCO, лишь незначительно уступая ColBERTv2 на тех же данных (0.397).

Однако, данная модель также страдает от необходимости вычислять контекстуализированные векторные представления на этапе поиска, значительно увеличивая количество необходимых ресурсов.

Заключение. Исследования в области информационного поиска, направленные на объединение разреженных и плотных представлений, продолжают оставаться актуальными. Современные глубокие модели, такие, как трансформеры, значительно улучшили качество поиска за счет возможности захвата семантических отношений между терминами. Тем не менее, эти модели сталкиваются с проблемами масштабируемости и требуют значительных вычислительных ресурсов, что ограничивает их применение в реальных системах, а модели на основе разреженных представлений страдают от отсутствия учета контекста и, как следствие, низкого качества ранжирования. Рассмотренные в статье модели, такие, как SNRM, SparTerm, SPLADE и другие, демонстрируют различные подходы к интеграции разреженных и плотных представлений. SNRM предлагает генерировать разреженные представления с использованием глубоких нейронных сетей, позволяя использовать обратные индексы для эффективного поиска. SparTerm и SPLADE улучшают этот подход, вводя более сложные механизмы оценки важности терминов и расширения запросов, что позволяет лучше учитывать семантические связи. Модель SPLADE, в частности, представляет значительное улучшение по сравнению с предыдущими подходами, используя логарифмическое насыщение и разреженную регуляризацию для достижения высокой эффективности, и качества поиска.

В статье авторы выполнили аналитический обзор основных исследований в области гибридных (разреженно-плотных) моделей, рассмотрев их ключевые принципы, проблемы и результаты на тестовых наборах данных. Особое внимание уделялось учету контекста, оптимизации хранения и извлечения информации, что позволило глубже проанализировать текущее состояние исследований. Анализ показал, что все рассмотренные подходы (SNRM, SparTerm, SPLADE и др.) способны сочетать высокое качество ранжирования с частичной сохранностью быстроты поиска, свойственной разреженным структурам. Однако остаются нерешенными задачи по дальнейшему повышению масштабируемости, а также по преодолению барьера вычислительной сложности, вызванного необходимостью обучения и применения плотных моделей.

Среди перспективных векторов развития гибридных моделей можно выделить:

1) разреживание уже обученных плотных моделей с целью их более эффективной интеграции в обратный индекс;
2) выделение кластеров векторных представлений токенов для более точного обучения словаря и оптимизации хранения векторных представлений;

3) динамическое изменение (адаптивный выбор) размерности разреженных векторов в зависимости от задач и ресурсов;
4) развитие методов гибридной индексации, учитывающих контекст на всех этапах поиска.

Таким образом, главным результатом выполненного аналитического обзора является систематизация существующих направлений в интеграции разреженных и плотных представлений, и демонстрация того, что при грамотном сочетании этих подходов можно достичь высокой точности поиска без критичного снижения производительности. Интеграция разреженных и плотных представлений в информационном поиске представляет собой перспективное направление исследований, которое потенциально может значительно улучшить качество и эффективность поисковых систем.

Список источников

1. Vaswani A., Shazeer N., Parmar N., et al. Attention is all you need, arXiv (Cornell University), 2017, vol. 30, pp. 5998-6008, DOI:10.48550/arXiv.1706.03762.
2. Guo J., Cai Y., Fan Y., Sun F., et al. Semantic models for the first-stage retrieval: A comprehensive review. ACM Transactions on Information Systems, 2022, 40(4), pp. 1-42, DOI:10.1145/3486250
3. Zamani H., Dehghani M., Croft W.B., et al. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing, The 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 497-506.
4. Bank D., Koenigstein N., Giryes R. Autoencoders. Deep learning in science, 2021.
5. Robertson S. E., Zaragoza H. The probabilistic relevance framework: BM25 and beyond, Found. Trends Inf. Retr., 2009, vol. 3, pp. 333-389.
6. Bai Y., Li X., Wang G., Zhang C. et al. SparTerm: Learning term-based sparse representation for fast text retrieval, arXiv, 2020.
7. Devlin J., Chang M. W., Lee K., Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv, 2018, DOI:10.48550/arXiv.1810.04805.
8. Hendrycks D., Gimpel K. Gaussian Error Linear Units (GELUs). arXiv, 2016, DOI:10.48550/arXiv.1606.08415.
9. Ba Jimmy L., Kiros Jamie Ryan, Hinton Geoffrey E. Layer normalization. arXiv, 2016, DOI:10.48550/arXiv.1607.06450.
10. Formal T., Piwowarski B., Clinchant S. SPLADE: Sparse lexical and expansion model for first stage ranking, Proceedings of the 44th International ACM SIGIR Conference on research and development in information retrieval, 2021.
11. Biswajit Paria, Chih-Kuan Yeh, Ian E. H. Yen, et al. Minimizing FLOPs to Learn Efficient Sparse Representations. arXiv, 2020, DOI:10.48550/arXiv.2004.05665.
12. Khattab O., Zaharia M.A. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT, Proceedings of the 43rd International ACM SIGIR Conference on research and development in information retrieval, 2020.
13. Johnson J., Douze M., Jégou H. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 2017, vol. 7, pp. 535-547, DOI:10.48550/arXiv.1702.08734.
14. Jégou H., Douze M., Schmid C. Product quantization for nearest neighbor search. IEEE Transactions on pattern analysis and machine intelligence, 2011, vol. 33, pp. 117-128, DOI:10.1109/TPAMI.2010.57.
15. Santhanam K., Khattab O., Saad-Falcon J., et al. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. North American chapter of the association for computational linguistics, 2021, DOI:10.48550/arXiv.2112.01488.
16. Kong W., Dudek J.M., Li C., et al. SparseEmbed: Learning sparse lexical representations with contextual embeddings for retrieval. Proceedings of the 46th International ACM SIGIR conference on research and development in information retrieval, 2023.

Абрамович Роман Константинович. *Аспирант, факультет программной инженерии и компьютерной техники университета ИТМО. AuthorID: 1253076, SPIN: 8710-8245, Scopus AuthorID: 58759320100, ORCID: 0009-0005-5397-2772, asmetliness24237@gmail.com, 197101, Россия, Санкт-Петербург, Кронверкский пр. 49.*

Добрынин Вячеслав Юрьевич. *Аспирант, факультет программной инженерии и компьютерной техники университета ИТМО. AuthorID: 1014269, SPIN: 9792-5868, Scopus AuthorID: 57223099701, ORCID: 0009-0004-3056-8403, vidobrynin@itmo.ru, 197101, Россия, Санкт-Петербург, Кронверкский пр. 49.*

Платонов Алексей Владимирович. *Доцент, к.т.н, факультет программной инженерии и компьютерной техники университета ИТМО. AuthorID: 1048089, SPIN: 1973-7334, Scopus AuthorID: 57197736275, ORCID: 0000-0002-8485-1296, avplatonov@itmo.ru, 197101, Россия, Санкт-Петербург, Кронверкский пр. 49.*

UDC 004.912
DOI:10.25729/ESI.2025.38.2.001

Combining deep language models and sparse vector representations in information retrieval: a review and analysis of modern approaches

Roman K. Abramovich, Viacheslav Yu. Dobrynin, Alexey V. Platonov

ITMO university,
Russia, St. Petersburg, *asmetliness24237@gmail.com*

Abstract. Traditional search methods based on sparse vector representations are characterized by high efficiency but limited quality due to their inability to capture semantic relationships in the data. On the other hand, dense vector representations can improve quality by capturing semantic relationships. However, these methods face scalability issues and require significant computational resources. With the development of deep neural networks, including transformer-based architectures, there is a growing interest in combining these two approaches. The purpose of this review paper is to review existing works that use deep models to generate sparse representations.
Keywords: deep neural networks, semantic search, computational resources, sparse representations, inverted index

References

1. Vaswani A., Shazeer N., Parmar N., et al. Attention is all you need, arXiv (Cornell University), 2017, vol. 30, pp. 5998-6008, DOI:10.48550/arXiv.1706.03762.
2. Guo J., Cai Y., Fan Y., Sun F. et al. Semantic models for the first-stage retrieval: A comprehensive review. ACM Transactions on Information Systems, 2022, 40(4), pp. 1-42, DOI:10.1145/3486250
3. Zamani H., Dehghani M., Croft W.B., et al. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing, The 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 497-506.
4. Bank D., Koenigstein N., Giryes R. Autoencoders. Deep learning in science, 2021.
5. Robertson S. E., Zaragoza H. The probabilistic relevance framework: BM25 and beyond, Found. Trends Inf. Retr., 2009, vol. 3, pp. 333-389.
6. Bai Y., Li X., Wang G., Zhang C. et al. SparTerm: Learning term-based sparse representation for fast text retrieval, arXiv:, 2020.
7. Devlin J., Chang M. W., Lee K., Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv, 2018, DOI:10.48550/arXiv.1810.04805.
8. Hendrycks D., Gimpel K. Gaussian Error Linear Units (GELUs). arXiv, 2016, DOI:10.48550/arXiv.1606.08415.
9. Ba Jimmy L., Kiros Jamie Ryan, Hinton Geoffrey E. Layer normalization. arXiv, 2016, DOI:10.48550/arXiv.1607.06450.
10. Formal T., Piwowarski B., Clinchant S. SPLADE: Sparse lexical and expansion model for first stage ranking, Proceedings of the 44th International ACM SIGIR Conference on research and development in information retrieval, 2021.
11. Biswajit Paria, Chih-Kuan Yeh, Ian E. H. Yen, et al. Minimizing FLOPs to Learn Efficient Sparse Representations. arXiv, 2020, DOI:10.48550/arXiv.2004.05665.
12. Khattab O., Zaharia M.A. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT, Proceedings of the 43rd International ACM SIGIR Conference on research and development in information retrieval, 2020.
13. Johnson J., Douze M., Jégou H. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 2017, vol. 7, pp. 535-547, DOI:10.48550/arXiv.1702.08734.
14. Jégou H., Douze M., Schmid C. Product quantization for nearest neighbor search. IEEE Transactions on pattern analysis and machine intelligence, 2011, vol. 33, pp. 117-128, DOI:10.1109/TPAMI.2010.57.

15. Santhanam K., Khattab O., Saad-Falcon J., et al. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. North American chapter of the association for computational linguistics, 2021, DOI:10.48550/arXiv.2112.01488.
16. Kong W., Dudek J.M., Li C., et al. SparseEmbed: Learning sparse lexical representations with contextual embeddings for retrieval. Proceedings of the 46th International ACM SIGIR conference on research and development in information retrieval, 2023.

Abramovich Roman Konstantinovich. *PhD student, Faculty of software engineering and computer science, ITMO university. SPIN: 8710-8245, AuthorID: 1253076, Scopus AuthorID: 58759320100, ORCID: 0009-0005-5397-2772,, asmetliness24237@gmail.com, 197101, Russia, Saint Petersburg, Kronverksky pr. 49.*

Dobrynin Viacheslav Yurievich. *PhD student, Faculty of software engineering and computer science, ITMO university. SPIN: 9792-5868, AuthorID: 1014269, Scopus AuthorID: 57223099701, ORCID: 0009-0004-3056-8403, vidobrynin@itmo.ru, 197101, Russia, Saint Petersburg, Kronverksky pr. 49.*

Platonov Alexey Vladimirovich. *Associate Professor, Ph.D., Faculty of software engineering and computer science, ITMO university. SPIN: 1973-7334, AuthorID: 1048089, Scopus AuthorID: 57197736275, ORCID: 0000-0002-8485-1296, avplatonov@itmo.ru, 197101, Russia, Saint Petersburg, Kronverksky pr. 49.*

Статья поступила в редакцию 09.07.2024; одобрена после рецензирования 20.03.2025; принята к публикации 11.04.2025.

The article was submitted 07/09/2024; approved after reviewing 03/20/2025; accepted for publication 04/11/2025.