УДК 004.89 doi: 10.15827/0236-235X.142.018-023 2024. Т. 37. № 1. С. 18–23

Анализ времени выполнения теста неравенства Белла для поиска информации

Альдарф Алаа [1]✉, Шакер Алаа [1], И.А. Бессмертный [1]

[1] Университет ИТМО, г. Санкт-Петербург, 197101, Россия

Ссылка для цитирования
Альдарф А., Шакер А., Бессмертный И.А. Анализ времени выполнения теста неравенства Белла для поиска информации // Программные продукты и системы. 2024. Т. 37. № 1. С. 18–23. doi: 10.15827/0236-235X.142.018-023
Информация о статье
Группа специальностей ВАК: 1.2.1
Поступила в редакцию: 08.07.2023 После доработки: 28.09.2023 Принята к публикации: 25.10.2023

Аннотация. Тест неравенства Белла повышает эффективность поиска информации и работы поисковых систем. Он упорядочивает полученные результаты на основе связей между словами, приоритезируя соответствующие ответы. Однако временные характеристики этого метода остаются неизученными, поскольку он работает медленнее метода TF-IDF. Методология исследования включает проведение экспериментов для анализа времени выполнения теста Белла и изучение различных аспектов самого теста и его компонентов. Эксперименты показывают, что вычисление матрицы HAL занимает значительную часть общего времени теста Белла, превышая 80 %. В работе также рассматривается использование библиотеки CuPy на графических процессорах для ускорения вычислений матрицы HAL, в результате выявлены лишь ограниченные преимущества ускорения на GPU из-за накладных расходов на передачу данных. В статье также представлен метод «сохранить и восстановить», который предполагает предварительное вычисление и сохранение матрицы HAL в БД с целью сокращения времени выполнения будущих запросов. Эффективность этого метода продемонстрирована на текстах с множеством повторяющихся слов, что приводит к более быстрому выполнению запросов по сравнению с повторным вычислением матрицы HAL для каждого запроса. Исследование имеет практическое значение для разработки эффективных систем поиска информации в реальном времени. Определяя основные компоненты, требующие много времени для выполнения теста Белла, особенно вычисление матрицы HAL, исследование выявляет потенциальные области для оптимизации и улучшения скорости и производительности поиска. Кроме того, внедренный метод «сохранить и восстановить» предлагает полезную стратегию для оптимизации систем поиска информации с текстами, содержащими повторяющийся контент.

Ключевые слова: тест неравенства Белла, поиск информации (IR), матрица HAL, библиотека CuPy (CUDA Python), библиотека NumPy, графический процессор (GPU), центральный процессор (CPU)

Введение. Огромный объем информационного потока, данных и веб-страниц усложняет процесс поиска (IR) из-за трудностей, связанных с восстановлением полезной и надежной информации [1]. Задача IR-систем может быть сведена к двум аспектам. Во-первых, как эффективно представить и ранжировать разнообразную неструктурированную информацию, создаваемую в каждый момент времени. Для этого необходимо решить такие задачи, как индексация и улучшение понимания контента с помощью передовых методов представления, а также ранжирование информационных элементов на основе представления. Во-вторых, как заставить IR-систему лучше понимать сложный запрос пользователя. Этот аспект включает в себя понимание контекста поиска пользователя, его намерений, а также способность измерять выполнение задачи и удовлетворенность пользователя посредством взаимодействия с ним [2]. Задачи IR-системы должны отвечать требованиям к скорости и времени, чтобы быть выполнимыми и применимыми в системах реального времени [3].

В последнее время появилось много исследовательских работ, посвященных квантовому подходу к поиску информации и анализу естественного языка [4]. При этом подходе используется тест Белла, который измеряет квантовую запутанность двух слов в контексте, что может вовлечь значения слов в процесс поиска и повысить его качество. Авторы работы [5] продемонстрировали применимость теста неравенства Белла к анализу текстов на естественном языке и поиску информации на русском языке. Результаты показали, что тест Белла сильно коррелирует с размером окна HAL (Hyperspace Analogue to Language) при моделировании семантического пространства. Более того, этот тест может быть использован для разделения текстовых документов на основе темы запроса. В [6] было предложено использовать тест Белла для оценки запутанности словарных пар в китайском языке. Экспе-

римент продемонстрировал валидность теста Белла для китайских текстов без сегментации слов с одним ограничением: тест Белла неприменим для узкого контекста. Авторы работы [7] представили новый метод поиска текста на арабском языке с помощью теста Белла и показали способность квантовой семантической модели определять, имеет ли изучаемый текст отношение к теме исследования или нет. Они продемонстрировали тесную взаимосвязь между размером окна HAL и качеством полученных результатов. Хотя время тестирования Белла относительно невелико, он ранее не анализировался с точки зрения времени.

Данная работа посвящена изучению времени тестирования Белла и экспериментам по ускорению этого теста посредством использования сохраненной матрицы HAL в БД и сравнения библиотеки CuPy на графическом процессоре (GPU) с библиотекой NumPy.

Анализ времени теста Белла и постановка задачи

Широко используемым методом построения текстового пространства и формализации документа в виде квадратной матрицы является HAL. Более того, HAL – это массив для связей между двумя терминами: векторы строк и столбцов записывают информацию о совместном появлении предыдущих и последующих слов отдельно [8]. На значение векторов HAL влияет размер окна. Более широкое окно означает большую вероятность ассоциаций между двумя терминами, но может пострадать из-за недостаточного соответствия. С другой стороны, небольшой размер окна означает сильную ассоциацию между двумя словами, но может пострадать от перенастройки. В данной работе использован фиксированный размер окна, равный 50. В N-мерной матрице HAL каждый документ будет иметь связанный с ним вектор. Векторное состояние документа – это сумма векторов всех слов, содержащихся в документе. Каждое слово векторного состояния извлекается из строк симметричной матрицы HAL [9].

В рамках настоящего исследования были проанализированы временные затраты на все фазы теста Белла, чтобы определить наиболее затратные. Количество уникальных слов в изучаемом тексте учитывается из-за его влияния на производительность. Все тесты в рамках данной работы проводились на графических процессорах NVIDIA (GTX 850m с 4 ГБ памяти) и Intel (64-разрядный i7-2,50 ГГц с 16 ГБ памяти). На рисунке 1 и в таблице 1 отражено время, необходимое на вычисления теста Белла для запроса из двух слов в трех текстовых файлах разного размера.

Время вычисления матрицы HAL занимает более 80 % от общего времени, поэтому ускорение ее вычисления значительно ускорит весь процесс. Сложность и время вычисления матрицы HAL зависят от объема изучаемого текста и размера окна HAL. Поскольку размер окна HAL напрямую влияет на точность результатов, для изучения влияния исследуемого объема текста при сохранении точности в рамках экспериментов использован фиксированный размер окна.

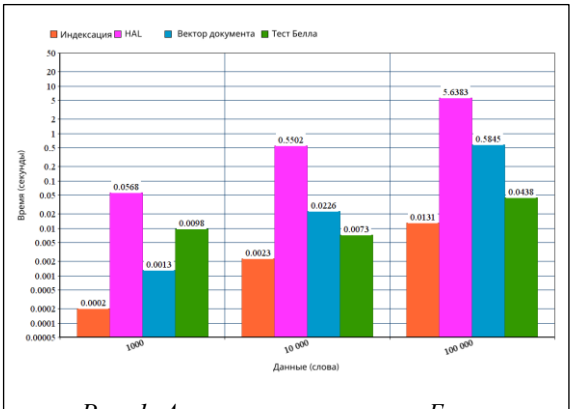

Рис. 1. Анализ времени теста Белла

Fig. 1. Bell test time analysis

Таблица 1

Анализ времени теста Белла

Table 1

Bell test time analysis

Общее количество слов	Количество уникальных слов	Время индексации	Время HAL	Время вектора документа	Время теста Белла	Общее время
10^3	581	0.0002674	0.0568687	0.0013880	0.0098938	0.0684181
10^4	3163	0.0023688	0.5502016	0.0226285	0.0073050	0.5825041
10^5	16641	0.0131049	5.6383774	0.5845391	0.0438911	6.2799127

Использование CuPy на графическом процессоре

Библиотека NumPy, как правило, ограничена одноузловой или многопоточной моделью выполнения только на центральном процессоре [10]. Поскольку наборы данных продолжают увеличиваться в размерах, а программы усложняются, растет потребность в решении этих проблем путем использования вычислительных ресурсов, намного превосходящих те, которые может предоставить один узел, работающий только на центральном процессоре [11]. Библиотека CuPy поддерживает большинство операций с массивами, которые есть в NumPy, включая индексацию, широковещательную передачу, вычисления с массивами и различные преобразования матриц [12]. Использование CuPy – способ многократно ускорить NumPy и матричные операции на графическом процессоре за счет использования библиотеки CUDA GPU [13]. Важно отметить, что ускорение сильно зависит от размера массива, с которым ведется работа. Время вычисления матрицы HAL с использованием CuPy на графическом процессоре по сравнению со временем вычисления с использованием Numpy отражено на рисунке 2.

Было обнаружено, что выполнение значительно замедлилось при использовании CuPy на графическом процессоре, поскольку скорость передачи данных в его память намного больше ускорения, возникающего в результате выполнения операций на графическом процессоре, из-за наличия небольшого количества вычислений по сравнению с размером данных.

Сохранение матрицы HAL

Метод, когда текст добавляется в БД и вычисляется матрица HAL, которая затем сохраняется в дополнение к исходному тексту в БД, назовем методом «сохранить и восстановить». Таким образом, при выполнении запроса матрицу HAL не нужно вычислять повторно: она просто извлекается из БД. Этот метод экономит время, поскольку не нужно повторно вычислять матрицу HAL, однако размер сохраняемой матрицы намного больше размера исходного текста, что является новым аспектом, который необходимо учитывать.

Успех этого метода зависит от нескольких факторов:

– общее количество слов в изучаемом тексте;

– количество уникальных слов в тексте;

– способы сохранения и восстановления матрицы HAL.

На рисунке 3 сравнивается время вычисления матрицы HAL со временем сохранения и восстановления для трех текстовых файлов разного размера, содержащих одинаковое количество уникальных слов.

Поскольку размер файла увеличивается, а количество уникальных слов остается постоянным, количество повторяющихся слов увеличивается, и метод сохранения и восстановления матрицы HAL работает быстрее, чем вычисление матрицы HAL.

На рисунке 4 сравнивается время вычисления матрицы HAL со временем сохранения и восстановления для трех текстовых файлов одинакового размера, содержащих разное количество уникальных слов.

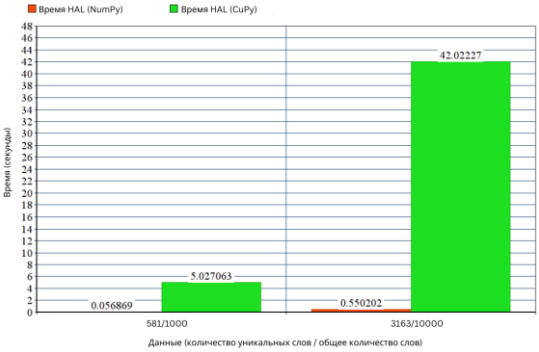

Рис. 2. Время вычисления матрицы HAL с использованием CuPy и NumPy

Fig. 2. HAL matrix computation time using CuPy and NumPy

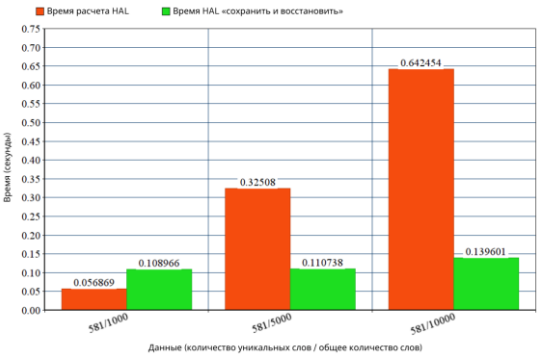

Рис. 3. Сравнение времени вычисления матрицы HAL со временем сохранения и восстановления для файлов разных размеров

Fig. 3. Comparing HAL matrix computation time with save and restore time for different file sizes

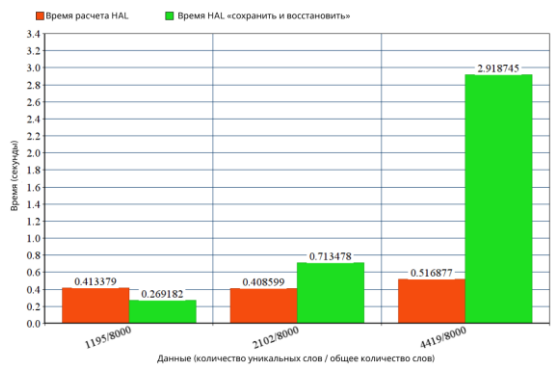

Рис. 4. Сравнение времени вычисления матрицы HAL со временем сохранения и восстановления для разного количества уникальных слов

Fig. 4. Comparing HAL matrix computation time with save and restore time for different numbers of unique words

Поскольку количество уникальных слов увеличивается, а размер файла остается постоянным, количество повторяющихся слов уменьшается, и метод сохранения и восстановления требует большего времени, чем вычисление матрицы HAL.

Экспериментальное исследование метода сохранения и восстановления показало, что он эффективен, когда изучаемый текст содержит много повторяющихся слов. Поскольку повторяющиеся слова в тексте уменьшают размер матрицы HAL и сохраняются только один раз, восстановление матрицы HAL меньшего размера будет быстрее и сэкономит время, необходимое для поиска по всему тексту для создания матрицы. Этот метод требует дополнительного места для хранения данных, но он приводит к более быстрому выполнению теста, если текст содержит много повторяющихся слов.

Заключение

Тест Белла дает хорошие результаты для извлечения информации из текстов на нескольких языках, но требует больше времени, чем другие используемые методы. Поэтому сокращение времени проведения этого теста имеет большое значение для определения возможности его применения в реальных приложениях. Данная работа показала, что вычисление матрицы HAL занимает более 80 % от общего времени проведения теста Белла, поэтому ускорение вычисления матрицы HAL окажет значительное положительное влияние на общий процесс. Было проведено сравнение времени вычисления матрицы HAL с использованием CuPy на GPU и NumPy. Результат показал, что использование CuPy значительно замедляет процесс вычислений, поскольку скорость передачи данных в память GPU намного больше, чем ускорение, вызванное выполнением операций на GPU. Метод сохранения и восстановления матрицы HAL был введен путем однократного вычисления матрицы HAL, сохранения ее в БД и восстановления при выполнении запроса. Практическое исследование доказало, что эффективность этого метода зависит от изучаемого текста и количества уникальных слов, которые он содержит: чем больше слов в изучаемом тексте повторяется, тем лучшие результаты дает метод.

В дальнейших исследованиях можно изучить влияние размера окна HAL на время вычисления матрицы и найти оптимальное значение, обеспечивающее достаточную точность при приемлемом времени выполнения. Кроме того, изучение влияния фрагментации текста и параллельной обработки может привести к хорошим результатам с точки зрения времени выполнения, и это делает актуальным изучение методов доступа к памяти и шаблонов хранения данных. Кроме того, авторы планируют запустить описанные тесты на более современных процессорах. Проблемой, не затронутой в данной работе, также является необходимость наличия места для хранения матрицы HAL в дополнение к тексту при использовании метода сохранения и восстановления.

Список литературы

1. Joby D.P. Expedient information retrieval system for web pages using the natural language modeling. AICN J., 2020, vol. 2, no. 2, pp. 100–110. doi: 10.36548/jaicn.2020.2.003.

2. Uprety S., Gkoumas D., Song D. A survey of quantum theory inspired approaches to information retrieval. ACM CSUR, 2020, vol. 53, no. 5, pp. 1–39. doi: 10.1145/3402179.

3. Stefani D., Turchet L. On the challenges of embedded real-time music information retrieval. Proc. Int. Conf. DAFx20in22, 2022, vol. 3, pp. 177–184.

4. Xain A., Goyal A., Singh B., Sharma S. Multilinguistic approach towards information retrieval system for Big Data. Proc. ICISS, 2020, pp. 159–164. doi: 10.1109/ICISS49785.2020.9315969.

5. Platonov A.V., Poleschuk E.A., Bessmertny I.A., Gafurov N.R. Using quantum mechanical framework for language modeling and information retrieval. Proc. Int Conf. AICT, 2018, pp. 1–4. doi: 10.1109/ICAICT.2018.8747051.

6. Bessmertny I.A., Huang X., Platonov A.V., Yu C., Koroleva J.A. Applying the Bell's test to Chinese texts. Entropy, 2020, vol. 22, no. 3, art. 275. doi: 10.3390/e22030275.

7. Shaker A., Aldarf A., Bessmertny I. The effectiveness of using bell inequality test for information retrieval in Arabic texts. Proc. MICSECS, 2020. URL: https://ceur-ws.org/Vol-2893/paper_7.pdf (дата обращения: 20.09.2023).

8. Ghavat A.K., Tekade B.G., Bhute V.S., Chikhalkar M.D., Vijaykar P. AI based symmetric answer evaluation system for descriptive answering. IRJMETS, 2020, vol. 2, no. 03, pp. 449–455.

9. Wu J.L., Xiao X., Yu L.C., Ye S.Z., Lai K.R. Using an analogical reasoning framework to infer language patterns for negative life events. BMC Medical Inform. and Decision Making, 2019, vol. 19, art. 173. doi: 10.1186/s12911-019-0895-8.

10. Wang Q., Pang J., Yue F., Yang S. SWPy: Python numerical computing library optimization for domestic many-core processors. Proc. ITAIC, 2022, pp. 1171–1178. doi: 10.1109/ITAIC54216.2022.9836688.

11. Okuta R., Unno Y., Nishino D., Hido S., Loomis C. Cupy: A NumPy-compatible library for NVIDIA GPU calculations. Proc. NIPS, 2017, vol. 151, no. 7, pp. 1–7.

12. Bauer M., Garland M. Legate NumPy: Accelerated and distributed array computing. Proc. Int. Conf. SC, 2019, art. 23, pp. 1–23. doi: 10.1145/3295500.3356175.

13. Morton J.M., Kaszyk K., Li L., Sun J., Dubach C., Steuwer M., Cole M., O'Boyle M.F. DelayRepay: Delayed execution for kernel fusion in Python. Proc. ACM SIGPLAN DLS, 2020, pp. 43–56. doi: 10.1145/3426422.3426980.

Software & Systems doi: 10.15827/0236-235X.142.018-023 2024, 37(1), pp. 18–23

Analyzing the time of Bell inequality test for information retrieval

Alaa Aldarf [1]✉, Alaa Shaker [1], Igor A. Bessmertny [1]

[1] ITMO University, St. Petersburg, 197101, Russian Federation

For citation
Aldarf, A., Shaker, A., Bessmertny, I.A. (2024) 'Analyzing the time of Bell inequality test for information retrieval', *Software & Systems*, 37(1), pp. 18–23 (in Russ.). doi: 10.15827/0236-235X.142.018-023
Article info
Received: 08.07.2023 After revision: 28.09.2023 Accepted: 25.10.2023

Abstract. The Bell inequality test enhances information retrieval and search engine efficiency. It orders retrieved results based on word relationships while prioritizing relevant outcomes. However, its time aspect remains unexplored since it is slower than the TF-IDF method. The research methodology of this work involves conducting experiments to analyze the time of the Bell test and exploring various aspects of the Bell test and its components. The experiments demonstrate that the HAL matrix computation constitutes a significant part of the total Bell test time exceeding 80%. The study also examines the use of the CuPy library on GPUs to accelerate HAL matrix calculations, which reveals that the benefits of GPU acceleration are limited due to data transfer overheads. Additionally, this work introduces the "save and restore" method, which involves precomputing and storing the HAL matrix in a database in order to reduce the time required for future queries. The effectiveness of this method is demonstrated for texts containing numerous repeated words that results in faster execution times compared to recalculating the HAL matrix for each query. The research holds practical significance for developing efficient and real-time IR systems. When identifying the major time-consuming components of the Bell test, particularly the the HAL matrix computation, the study points to potential areas for optimization and improvement in search speed and performance. Moreover, the introduced "save and restore" method provides a useful strategy for optimizing IR systems with texts containing repetitive content.

Keywords: Bell Inequality Test, Information Retrieval (IR), Hyperspace Analog Language (HAL), CuPy (CUDA Python) library, NumPy library, Graphics Processing Unit (GPU), Central Processing Unit (CPU)

References

1. Joby, D.P. (2020) 'Expedient information retrieval system for web pages using the natural language modeling', *AICN J.*, 2(2), pp. 100–110. doi: 10.36548/jaicn.2020.2.003.

2. Uprety, S., Gkoumas, D., Song, D. (2020) 'A survey of quantum theory inspired approaches to information retrieval', *ACM CSUR*, 53(5), pp. 1–39. doi: 10.1145/3402179.

3. Stefani, D., Turchet, L. (2022) 'On the challenges of embedded real-time music information retrieval', *Proc. Int. Conf. DAFx20in22*, 3, pp. 177–184.

4. Xain, A., Goyal, A., Singh, B., Sharma, S. (2020) 'Multilinguistic approach towards information retrieval system for Big Data', *Proc. ICISS*, pp. 159–164. doi: 10.1109/ICISS49785.2020.9315969.

5. Platonov, A.V., Poleschuk, E.A., Bessmertny, I.A., Gafurov, N.R. (2018) 'Using quantum mechanical framework for language modeling and information retrieval', *Proc. Int Conf. AICT*, pp. 1–4. doi: 10.1109/ICAICT.2018.8747051.

6. Bessmertny, I.A., Huang, X., Platonov, A.V., Yu, C., Koroleva, J.A. (2020) 'Applying the Bell's test to Chinese texts', *Entropy*, 22(3), art. 275. doi: 10.3390/e22030275.

7. Shaker, A., Aldarf, A., Bessmertny, I. (2020) 'The effectiveness of using bell inequality test for information retrieval in Arabic texts', *Proc. MICSECS*, available at: https://ceur-ws.org/Vol-2893/paper_7.pdf (accessed September 20, 2023).

8. Ghavat, A.K., Tekade, B.G., Bhute, V.S., Chikhalkar, M.D., Vijaykar, P. (2020) 'AI based symmetric answer evaluation system for descriptive answering', *IRJMETS*, 2(03), pp. 449–455.

9. Wu, J.L., Xiao, X., Yu, L.C., Ye, S.Z., Lai, K.R. (2019) 'Using an analogical reasoning framework to infer language patterns for negative life events', *BMC Medical Inform. and Decision Making*, 19, art. 173. doi: 10.1186/s12911-019-0895-8.

10. Wang, Q., Pang, J., Yue, F., Yang, S. (2022) 'SWPy: Python numerical computing library optimization for domestic many-core processors', *Proc. ITAIC*, pp. 1171–1178. doi: 10.1109/ITAIC54216.2022.9836688.

11. Okuta, R., Unno, Y., Nishino, D., Hido, S., Loomis C. (2017) 'Cupy: A NumPy-compatible library for NVIDIA GPU calculations', *Proc. NIPS*, 151(7), pp. 1–7.

12. Bauer, M., Garland, M. (2019) 'Legate NumPy: Accelerated and distributed array computing', *Proc. Int. Conf. SC*, art. 23, pp. 1–23. doi: 10.1145/3295500.3356175.

13. Morton, J.M., Kaszyk, K., Li, L., Sun, J., Dubach, C., Steuwer, M., Cole, M., O'Boyle, M.F. (2020) 'DelayRepay: Delayed execution for kernel fusion in Python', *Proc. ACM SIGPLAN DLS*, pp. 43–56. doi: 10.1145/3426422.3426980.

Авторы

Альдарф Алаа [1], аспирант,
aaldarf@itmo.ru
Шакер Алаа [1], аспирант,
alaashaker@itmo.ru
Бессмертный Игорь Александрович [1],
д.т.н., профессор, bessmertny@itmo.ru

[1] Университет ИТМО,
г. Санкт-Петербург, 197101, Россия

Authors

Alaa Aldarf [1], Postgraduate Student,
aaldarf@itmo.ru
Alaa Shaker [1], Postgraduate Student
alaashaker@itmo.ru
Igor A. Bessmertny [1], Dr.Sc. (Engineering),
Professor, bessmertny@itmo.ru

[1] ITMO University,
St. Petersburg, 197101, Russian Federation