Сравнение одновекторного и многовекторного подходов при выполнении запросов к реляционным базам данных

Чан Дык Мань
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
duc.manh.tran@mail.ru

Юлия А. Шичкина
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
strange.y@mail.ru

Ха Муон
Факультет информационных технологий Телекоммуникационный университет Нячанг, Вьетнам
muon.ha@mail.ru

Аннотация. **Развитие методов векторизации данных способствует расширению возможностей выполнения более сложных и нечетких запросов к реляционным базам данных. В предлагаемом в статье подходе текстовые данные преобразуются в многомерные векторы, что позволяет оценить сходство между вектором запроса и вектором данных. При этом качество выполнения нечеткого запроса с применением технологии векторизации зависит от способа построения вектора, порядка и количества данных, представленных вектором. В данной статье сравниваются два подхода к выполнению запросов к реляционным базам данных с помощью технологии векторизации: одновекторный и многовекторный подход. Эффективность подходов оценивается на основе времени выполнения запросов и качества результатов поиска записей. Анализ результатов тестирования подходов показывает, что каждый из рассмотренных подходов обладает своими преимуществами и ограничениями. Подходы, обеспечивающие более высокую скорость выполнения запросов за счет меньших временных затрат на реализацию векторизации данных, как правило, демонстрируют более низкое качество результатов поиска записей. Напротив, подходы, требующие большего времени для выполнения векторизации данных, обычно обеспечивают более высокое качество результатов поиска записей.**

Ключевые слова — векторизация, SQL, поиск информации, одновекторный подход, многовекторный подход, TF-IDF

I. Введение

В последние годы технологии векторизации стали прорывным решением в области обработки естественного языка и поиска информации. Технология векторизации лежит в основе работы многих современных систем — от семантических поисковых систем до систем рекомендаций контента и чат-ботов [1].

Многие исследования применяют технологию векторизации для повышения эффективности поиска информации в различных областях. Например, технология векторизации TF-IDF применяется для улучшения возможности поиска похожих статей в интернете [2], а также для поддержки поиска информации в медицинской сфере [3]. Некоторые другие исследования показывают, что применение технологий векторизации, таких как Word2Vec, fasttext и Glove, улучшит эффективность поиска информации по сравнению с традиционными методами, основанными на ключевых словах [4], [5], [6]. Кроме того, применение технологий векторизации текста для выполнения запросов возможно для улучшения возможностей поиска информации в реляционных системах баз данных [7], [8].

Результаты анализа существующих исследований в этой области показывают, что, несмотря на широкое применение технологий векторизации, ни одно из исследований не было посвящено сравнению различных подходов к представлению данных в виде векторов при выполнении запросов к реляционным базам данных, особенно различий между представлением тестовых полей в форме отдельных векторов и в форме единого вектора, между различными последовательностями текстовых полей при их объединение в единый вектор. Данное исследование направлено на анализ и сравнение эффективности различных подходов к представлению текстовых полей в виде векторов, с целью выработки рекомендаций по применению технологий векторизации при реализации запросов к реляционным базам данных.

Цель данного исследования – сравнить эффективность двух подходов к векторизации значений текстовых полей, участвующих в запросах к реляционным базам данных:

- одновекторный подход, при котором данные полей, относящиеся к запросу, представляются одним вектором.
- многовекторный подход, при котором данные полей, относящиеся к запросу, представляются отдельными векторами.

Оценка эффективности проводится на основе таких критериев, как время выполнения запроса и качество результатов поиска записей.

II. Подходы

В этом разделе представлены подходы к выполнению запросов к реляционным базам данных с использованием технологии векторизации. Предполагается, что реляционная база данных содержит одну таблицу.

A. Одновекторный подход

В данном подходе поля данных, относящиеся к запросу, представляются в форме единого вектора для каждой записи. Данный подход может быть реализован двумя способами.

Способ 1. Все значения полей последовательно объединяются в единый текст перед выполнением векторизации.

Пусть входными данными являются:

Records – набор всех записей таблицы.

q – запрос.

Fq – набор полей, участвующих в запросе q.

Vectorization() – функция для преобразования текста в вектор.

Distance() – функция для вычисления расстояния между векторами.

k – количество возвращаемых результатов.

Выходные данные:

k записей, наиболее близких к точному ответу на запрос.

Тогда данный способ может быть реализован в несколько этапов, псевдокод которых приведен ниже.

Этап 1. Преобразовать значения полей в вектор для каждой записи.

```
vector_list  empty ← empty
for each record r in Records do
        doc_r ← empty string
        for each field f in Fq do
                doc_r ← doc_r + value of r[f] + "."
        vec_r ← Vectorization(doc_r)
        vector_list.add((r, vec_r))
end for
```

Этап 2. Преобразовать значения полей запроса в вектор.

```
doc_q ← empty string
for each field f in F do
        doc_q ← doc_q + value of q[f] + "."
vec_q ← Vectorization(doc_q)
```

Этап 3. Вычислить расстояния между векторами записей и векторам запроса.

```
scores_list ← empty list
   for each (r, vec_r) in vector_list do
      score ← Distance(vec_q, vec_r)
      scores_list.add((r, score))
end for
```

Этап 4. Сортировка и поиск результатов.

Отсортировать «scores_list» по «score» в порядке возрастания и вернуть первые k элементов «scores_list».

Способ 2. Значение каждого поля сначала векторизуется, а затем векторы объединяются.

Пусть входными данными являются:

Records – набор всех записей таблицы A.

q – запрос.

Fq – набор полей, участвующих в запросе q.

Vectorization() – функция для преобразования текста в вектор.

Distance() – функция для вычисления расстояния между векторами.

k – количество возвращаемых результатов.

Выходные данные:

k записей, наиболее близких к точному ответу на запрос.

Тогда данный способ может быть реализован в несколько этапов, псевдокод которых приведен ниже.

Этап 1. Преобразовать значения полей в вектор для каждой записи.

```
record_vectors ← empty list
for each record r in Records do
        field_vectors ← empty list
        for each field f in F do
                vec_f ← Vectorization(r[f])
                field_vectors.append(vec_f)
        end for
        vec_r ← concatenate(field_vectors)
        record_vectors.add((r, vec_r))
        end for
```

Этап 2. Преобразовать значения полей запроса в вектор.

```
query_vectors ← empty list
for each field f in Fq do
        vec_qf ← Vectorization(q[f])
        query_vectors.append(vec_qf)
end for
vec_q ← concatenate(query_vectors)
```

Этап 3. Вычислить расстояния между векторами записей и векторам запроса.

```
scores_list ← empty list
for each (r, vec_r) in record_vectors do
        score ← Distance(vec_q, vec_r)
        scores_list.add(r, score)
end for
```

Этап 4. Сортировка и поиск результатов.

Отсортировать «scores_list» по «score» в порядке возрастания и вернуть первые k элементов «scores_list».

B. Многовекторный подход

В многовекторном подходе каждое поле данных рассматривается как независимый вектор.

Пусть входными данными являются:

Records – набор всех записей таблицы A.

q – запрос.

Fq – набор полей для, участвующих в запросе q.

Vectorization() – функция для преобразования текста в вектор.

Distance() – функция для вычисления расстояния между векторами.

k – количество возвращаемых результатов.

Выходные данные:

k записей, наиболее близких к точному ответу на запрос.

Тогда данный подход может быть реализован в несколько этапов, псевдокод которых приведен ниже.

Этап 1. Преобразовать значения полей в вектор для каждой записи.

```
record_vectors ← empty list
for each record r in Records do
    field_vectors ← empty list
    for each field f in F do
        vec_f ← Vectorization(r[f])
        field_vectors.append(vec_f)
    end for
    record_vectors.add((r, field_vectors))
end for
```

Этап 2. Преобразовать значения полей запроса в вектор.

```
query_vectors ← empty list
for each field f in F do
    vec_qf ← Vectorization(q[f])
    query_vectors.append(vec_qf)
end for
```

Этап 3. Вычислить расстояния между векторами записей и вектором запроса.

```
scores_list ← empty list
for each (r, field_vectors) in record_vectors do
    total_distance ← 0
    for i from 1 to length(Fq) do
        d←Distance(query_vectors[i], field_vectors[i])
        total_distance ← total_distance + d
    end for
scores_list.add((r, total_distance))
end for
```

Этап 4. Сортировка и поиск результатов.

Отсортировать «scores_list» по «total_distance» в порядке возрастания и вернуть первые k элементов «scores_list».

III. Результаты

A. *База данных и запросы*

Оценка времени выполнения запросов и качества результатов поиска записей с использованием двух способов одновекторного подхода и одного способа многовекторного подхода проводилась на базе данных: Fashion Clothing Products Dataset [9]. К базе данных были выполнены следующие 4 запроса: Q1, содержащий одно текстовое поле; Q2 содержащий 2 текстовых поля, Q3 с 3мя текстовыми полями, Q4 с 4мя текстовыми полями.

B. *Времени выполнения запросов и качество результатов поиска записей*

Программа для тестирования подходов была написана на Python в среде Anaconda, и в качестве системы управления базами данных использовалась MySQL. Для выполнения векторизации данных использовался технология векторизации TF-IDF.

На рис. 1 показаны графики изменения времени выполнения запросов Q1–Q4 с помощью трех способов.

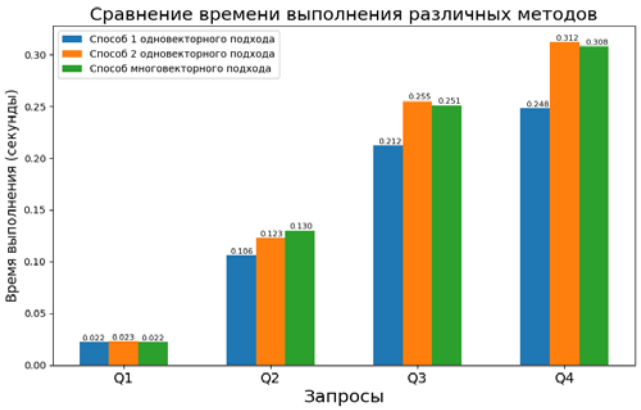

Рис. 1. Времени выполнения различных способов

Из рис. 1 видно, что способ 1 одновекторного подхода имеет наименьшее время выполнения запроса. При этом различие во времени выполнения запросов между способом 2 одновекторного подхода и многовекторным подходом является незначительным. Разница во времени выполнения запросов между способами объясняется тем, что способ 1 одновекторного подхода выполняет меньшее количество операций по векторизации данных, чем способ 2 одновекторного подхода и способ многовекторного подхода. Количество операций по векторизации, выполняемых способом 2 одновекторного подхода и многовекторным подходом, одинаково.

Для оценки качества результатов поиска записей тремя вышеописанными способами был проведён расчет следующих метрик: точность (precision), полнота (recall) и F1-мера (F1-score). На рис. 2 показаны тепловые карты, отображающие точность трех способов при выполнении запросов Q1–Q4 в зависимости от количества возвращаемых результатов k.

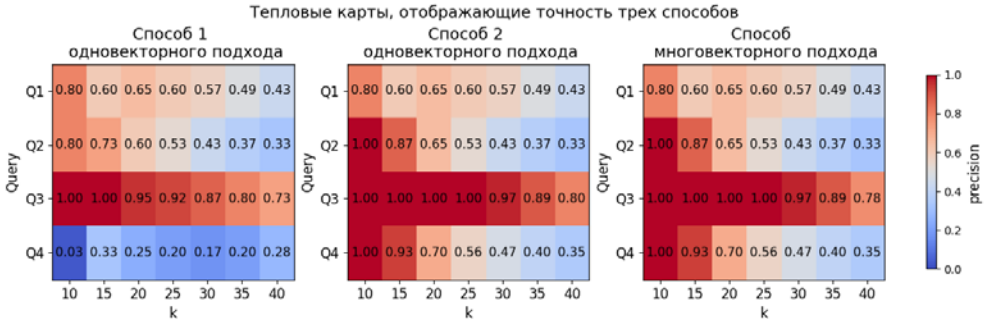

Рис. 2. Тепловые карты, отображающие точность выполнения запросов тремя способами

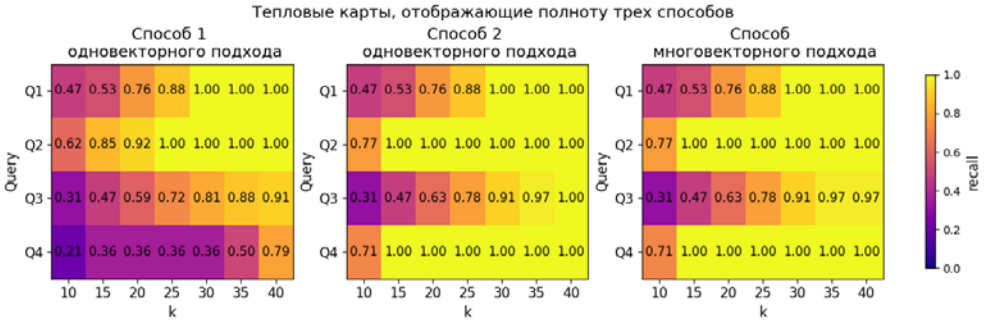

Рис. 3. Тепловые карты, отображающие полноту трех способов

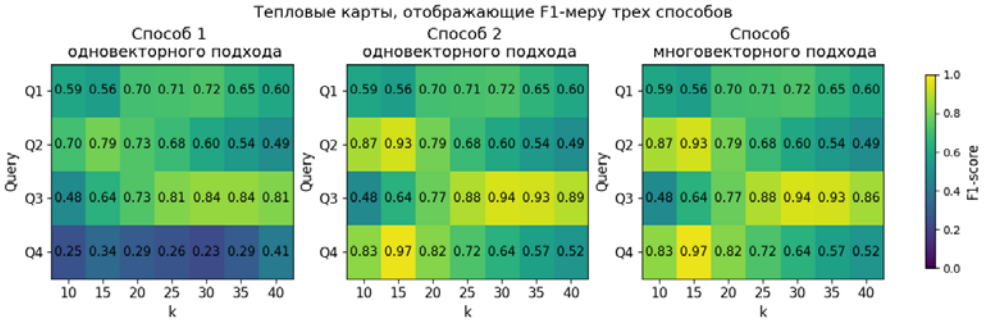

Рис. 4. Тепловые карты, отображающие F1-меру трех способов

Из рис. 2 видно, что точность трех способов обратно пропорциональна количеству возвращаемых результатов.

На рис. 3 показаны тепловые карты, отображающие полноту трех способов при выполнении запросов Q1–Q4 в зависимости от количества возвращаемых результатов k.

Из рис. 3 видно, что полнота трех способов пропорциональна количеству возвращаемых результатов.

На рис. 4 показаны тепловые карты, отображающие F1-меру трех способов при выполнении запросов Q1–Q4 в зависимости от количества возвращаемых результатов k.

Из рис. 4 видно, что по мере увеличения количества возвращаемых результатов показатель F1-меры сначала возрастает до своего максимального значения, а затем постепенно снижается.

Из результатов, представленных на рис. 2–4, видно, что способ 2 одновекторного подхода и многовекторный подход имеют практически одинаковое качество результатов поиска записей при выполнении различных запросов. В случае, когда запрос содержит одно текстовое поле (Q1), качество результатов поиска записей трех способов одинаково. Однако для запросов, содержащих несколько текстовых полей (Q2–Q4), способ 1 одновекторного подхода имеет самое низкое качество результатов поиска записей.

IV. Заключение

В данной работе сравнивались различные подходы к выполнению запросов с использованием технологии векторизации по времени выполнения запросов и качеству результатов поиска записей. Результаты показывают, что качество результатов поиска записей обратно пропорционально времени выполнения запроса. Качество результатов поиска записей для способа 2 одновекторного подхода и многовекторного подхода является эквивалентным и выше, чем для способа 1 одновекторного подхода. Качество результатов поиска записей способов не зависит от количества полей при простых запросах (одно поле), однако при увеличении числа полей качество способа 1 одновекторного подхода снижается, тогда как способ 2 одновекторного подхода и многовекторного подхода сохраняют высокое качество.

В дальнейшем планируется разработать метод выбора последовательности полей для повышения качества результатов поиска записей с использованием технологии векторизации.

Список источников

[1] J. Zhang, J. Zhang, S. Ma, J. Yang, and G. Gui, “Chatbot Design Method Using Hybrid Word Vector Expression Model Based on Real Telemarketing Data.,” *KSII Transactions on Internet & Information Systems*, vol. 14, p. 1400, Apr. 2020, doi: 10.3837/tiis.2020.04.001.

[2] A. N. Khusna and I. Agustina, “Implementation of Information Retrieval Using Tf-Idf Weighting Method On Detik.Com’s Website,” in 2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA), Yogyakarta, Indonesia: IEEE, Oct. 2018, pp. 1–4. doi: 10.1109/TSSA.2018.8708744.

[3] M. Zandigohar and Y. Dai, “Information retrieval in single cell chromatin analysis using TF-IDF transformation methods,” in 2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Las Vegas, NV, USA: IEEE, Dec. 2022, pp. 877–882. doi: 10.1109/BIBM55620.2022.9994949.

[4] N. Nagpal, “Query Expansion for Information Retrieval using Word Embeddings: A Comparative Study,” in 2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS), Tashkent, Uzbekistan: IEEE, Oct. 2022, pp. 289–293. doi: 10.1109/ICTACS56270.2022.9988667.

[5] D. Roy, D. Ganguly, S. Bhatia, S. Bedathur, and M. Mitra, “Using Word Embeddings for Information Retrieval: How Collection and Term Normalization Choices Affect Performance,” in Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino Italy: ACM, Oct. 2018, pp. 1835–1838. doi: 10.1145/3269206.3269277

[6] G. Zuccon, B. Koopman, P. Bruza, and L. Azzopardi, “Integrating and Evaluating Neural Word Embeddings in Information Retrieval,”

in Proceedings of the 20th Australasian Document Computing Symposium, Parramatta NSW Australia: ACM, Dec. 2015, pp. 1–8. doi: 10.1145/2838931.2838936

[7] F. Khalifeh, M. Taheri, E. Mansoori, and M. Fakhrahmad, “Enhancing Keyword Search in Relational Databases With Word Embeddings,” IEEE Access, vol. 13, pp. 105401–105416, 2025, doi: 10.1109/ACCESS.2025.3574428

[8] R. Bordawekar and O. Shmueli, “Using Word Embedding to Enable Semantic Queries in Relational Databases,” in Proceedings of the 1st Workshop on Data Management for End-to-End Machine Learning, Chicago IL USA: ACM, May 2017, pp. 1–4. doi: 10.1145/3076246.3076251

[9] “Fashion Clothing Products Dataset.” Accessed: Mar. 03, 2026. [Online]. Available: https://www.kaggle.com/datasets/shivamb/fashion-clothing-products-catalog