Усовершенствования алгоритма автоматизированного определения рифмы

В.Б. Барахнин[1], О.Ю. Кожемякина[1], И.С. Пастушков[1], И.В. Кузнецова[1], Ю.С. Борзилова[1]

[1]Институт вычислительных технологий СО РАН, просп. акад. Лаврентьева 6, Новосибирск, Россия, 630090

Аннотация. В работе рассматриваются подходы к усовершенствованию одного из шагов алгоритма, используемого для автоматизированного определения рифмы в поэтических текстах. Инструмент автоматизированного определения рифмы является одним из модулей системы комплексного анализа поэтических текстов. В текущей реализации модуля подзадача поиска и определения рифмы решается с помощью нахождения слов с созвучными окончаниями с использованием «Грамматического словаря русского языка» А. А. Зализняка и базовых правил фонетического анализа. Предложены альтернативные варианты решения задачи поиска в словаре слов с созвучными окончаниями. Полученные результаты позволят сделать вывод об оптимальности текущей реализации и возможной доработки используемых методов для решения задач определения рифмовки поэтического текста.

1. Введение

Одной из задач автоматизации комплексного анализа поэтических текстов [1] является определение характеристик, связанных с метроритмикой произведения. Среди работ, в которых статистическая информация, извлеченная из поэтического текста, применялась для задач филологии, можно отметить исследование [2], которое, несмотря на составление его вручную, представляет достаточно исчерпывающую статистическую картину метроритмики пушкинских произведений, что позволяет авторам находить закономерности, присущие пушкинской рифме. Современные информационные технологии позволяют проводить такие исследования если не полностью автоматически, то с минимальным использованием труда экспертов-филологов.

Решение задачи автоматизированного анализа поэтических текстов требует адаптации к различным языкам. Разные подходы обусловлены как спецификой языка (в частности, особенностями построения поэтических текстов), так и инструментарием, используемым исследователями. Инструментарий, в свою очередь, зависит от целей, поставленных исследователями (например, получить подтверждение какой-либо закономерности в структуре поэтического текста), и, в некоторой степени, от технологий, актуальных на момент исследования. Что касается языковой универсальности инструментов, то невозможно разработать систему автоматического анализа метра и ритма, предназначенную для широкого спектра языков. Более того, неразрешимой задачей является разработка системы анализа метроритмики, подходящей хотя бы под группу близких языков – каждый язык требует выработки своих подходов, учитывающих его структуру. Авторами исследования [3]

проводился эксперимент для схожих по структуре языков, который закончился безуспешно из-за специфики каждого из рассматриваемых авторами языков.

В работе [4], рассматривались автоматические процедуры спецификации поэтического текста - метроритмическая разметка и идентификация стихового метра. Автоматизация метроритмической разметки достигается с помощью следующих процедур [5]:

- нумерация строк произведения;
- токенизация слов;
- акцентуация произведения;
- выделение рифмованных строк;
- выделение слоговой схемы.

Авторы разработали открытый сетевой ресурс [6], который представлен компонентами: проблемно-ориентированным «Тезаурусом по поэтологии» и «Блоком анализа и спецификации» текстовых объектов. В блоке «анализа и спецификации» выделяются два комплекса задач [4]: спецификация терминологических статей тезауруса и спецификация поэтического произведения. Структура комплекса включает в себя группы решений задач:

- метроритмическая разметка текста;
- заполнение полей спецификации произведения;
- идентификация метра.

Среди инструментов, выполняющих метроритмический анализ, представляют интерес веб-ресурсы [7] и [8]. Первый из них, Рифмовед.ру, позиционируется как вспомогательный инструмент для анализа поэтического текста, который выделяет строфику и определяет формы (сонет, секстина). Алгоритмы были разработаны на основе авторской концепции программного стихоанализа В. Онуфриева, однако в научных источниках не найдено теоретического описания этих алгоритмов. Авторы ресурса указывают, что работа алгоритмов рассчитана на анализ поэтических текстов, написанных в традиционных формах, классическими строфами и размерами. Этот факт сильно ограничивает применение инструмента для больших корпусов поэтических текстов для поэтов, не относящихся к классической литературе. Второй ресурс, Стихоробот, очевидно, представляет собой любительский веб-портал для определения стихотворного размера, генерации новых стихотворений и подбор рифмы. При попытке определения стихотворного размера сервис уточняет все спорные ситуации (варианты акцентуации), что занимает много времени у пользователя. Теоретическое описание используемых алгоритмов в доступных источниках также отсутствует.

В ИВТ СО РАН разработано веб-приложение [9], которое используется для анализа структурного уровня русскоязычных поэтических текстов. Алгоритмы, лежащие в основе приложения, описаны в [10]. Приведенный алгоритм не предполагает работу со сложными случаями анализа поэтических текстов, поэтому в [11] предложена реализация алгоритмов из [4], который включает более строгую классификацию стихотворений по метру. Но в алгоритме определения рифмы из [4] авторы используют веб-приложение «Большой словарь рифм» [12]: на вход приложению поступает слово, на выходе возвращается полное множество слов, рифмующееся с ним (в отрыве от контекста). Однако подход занимает много времени и ресурсов, поэтому разрабатываемый нами алгоритм поиска рифм реализован из соображений возможности образования рифмы: строки рифмуются, если у последних слов в строке одинаковая позиция ударного слога и фонетически совпадают окончания.

Для выявления фонетически совпадающих окончаний мы используем данные об окончаниях из статьи [13]. Алгоритм обращается за словом в таблицу со словами, агрегированными из словаря А.А. Зализняка [14], реализованный стандартным образом.

Цель исследования заключается в поиске альтернативных подходов поиска рифмующихся строк в базе данных и проведения серии экспериментов для выяснения наиболее эффективного способа для использования в алгоритме. Предлагаемое решение:

1. Построить таблицу с инвертированными строками, отсортированных лексикографически.

2. Разбить все слова на секции по диапазонам после окончаний. Другими словами, хранить окончания (инвертированные) отдельно от слова (в качестве метаданных).
3. Добавить триграммные символьные индексы в оригинальной таблице со всеми словами.
4. Выполнить эксперимент по поиску рифмующихся слов с помощью секций (искать только окончания) и триграммных индексов.
5. Сравнить производительность варианта поиска по секциям с применением индексов или с помощью комбинации этих вариантов.

2. Подготовка данных

Одним из вариантов реализации поиска является разбиение исходной таблицы со словами по секциям. Версия БД PostgreSQL, развернутой на сервере ИВТ СО РАН, поддерживает простое секционирование: разбиение одной большой логической таблицы на несколько небольших физических секций [15]. Преимущества использования секций:

– Когда в выборке или изменении данных задействована бóльшая часть одной секции, последовательное сканирование этой секции может выполняться гораздо быстрее, чем случайный доступ по индексу к данным, разбросанным по всей таблице.

– Массовую загрузку и удаление данных можно осуществлять, добавляя и удаляя секции, если это было предусмотрено при проектировании секций. Команды *ALTER TABLE NO INHERIT* и *DROP TABLE* работают гораздо быстрее, чем массовая загрузка. Эти команды также полностью исключают накладные расходы, связанные с выполнением операции *VACUUM* после команды *DELETE*.

– Редко используемые данные можно перенести на более медленные, а, следовательно, более дешевые носители.

В PostgreSQL возможно реализовать секционирование по диапазонам (например по диапазонам дат) или по списку – явно указывается, какие значения ключа должны относиться к каждой секции. В рамках данного исследования применяется секционирование по списку, где в качестве значений ключа указываются окончания слов словаря.

Для создания списка секций, представляющих собой таблицы, используется python-скрипт, действующий по алгоритму:

1. Обращение к таблице со словами.
2. Выделение последних *N* символов из слова.
3. Формирование массива всех окончаний словаря.
4. Подсчет и сортировка использования каждого окончания в порядке убывания.
5. Выделение *M* первых окончаний из массива, на основе которых будет выполнено секционирование.

В качестве *N* последних символов взяты 4 символа, это значение возможно изменить в будущем. Для построения отсортированного словаря используется модуль collections библиотеки Counter [16]. В результате получен словарь следующей структуры:

Counter({'НОГО': 86077, 'ЕЙСЯ': 76978, 'ВШЕЙ': 76400, 'ИМСЯ': 62934, 'ШЕГО': 61719, 'ГОСЯ': 57630, 'ИХСЯ': 57617, 'НОМУ': 57354, 'ВШИМ': 57282 ...})

В полученном словаре ключ – искомое окончание (последние *N* символов), а значение – количество вхождений данного окончания. Из созданного словаря решено выделить значения окончаний с коэффициентами, входящими в 90-ю процентиль. Эти окончания использовались для создания секций.

Процесс создания секционированных таблиц включает в себя следующие этапы:

1. Создание родительской таблицы, свойства которой наследуют все дочерние таблицы (секции). Родительской таблицей является таблица со словами из словаря А. А. Зализняка [14] со структурой:
   а) идентификатор;
   б) слово;
   в) акцентуация (номер слога, на который падает ударение);
   г) тип слова;

В качестве дополнительной колонки добавлены окончания (последние *N* символов) каждого слова в инвертированном порядке.

2. Создание дочерних таблиц с наследованием структуры родительской. В этих таблицах не будет никаких дополнительных колонок, кроме унаследованных. Все дочерние таблицы будем называть секциями.

3. Добавление в таблицы-секции ограничений, определяющих допустимые значения ключей для каждой секции. Ограничения при этом не пересекаются, то есть никакие значения ключа не относятся сразу к нескольким секциям.

4. Создание индекса по ключевой колонке для каждой секции. В исследовании индексы создавались для значения «Слово».

5. Определение триггера для перенаправления данных, добавляемых в главную таблицу, в соответствующую секцию. Созданный триггер срабатывает при SQL-команде *INSERT*.

Созданный триггер запускает функцию, которая добавляет значения в соответствующую секцию (таблицу). Фрагмент функции:

```
CREATE OR REPLACE FUNCTION words_with_reversed_endings_insert_function()
RETURNS TRIGGER AS $$
BEGIN
    IF (NEW.ending = 'нии') THEN
            INSERT INTO words_with_endings_nii (id, word_form, ending, accent, word_type)
            VALUES (NEW.id, NEW.word_form, reverse(NEW.ending), NEW.accent,
    NEW.word_type);
    ELSIF (NEW.ending = 'ний') THEN
            INSERT INTO words_with_endings_niy (id, word_form, ending, accent, word_type)
            VALUES (NEW.id, NEW.word_form, reverse(NEW.ending), NEW.accent,
    NEW.word_type);
```

Создание таблиц, индексов, триггера и функции выполняется через python-скрипт в автоматизированном режиме. Требуется ручная корректировка названий таблиц и индексов, поскольку для именования использовались транслитерованные названия окончаний – в некоторых случаях возникали ситуации, которые потребовали ручного вмешательства (используется транслитератор transliterate [17]). К таким можно отнести, например, совпадение названий при транслитерации окончаний «ЕМСЯ» и «ЁМСЯ».

В контексте БД PostgreSQL триграмма – это группа из трех последовательных символов. Мы можем измерить схожесть двух строк, подсчитав количество совпадающих триграмм. Эта простая идея оказывается очень эффективной для измерения схожести слов на многих естественных языках, а также для решения смежных задач, таких как, например, нечеткий поиск (здесь подразумевается поиск по сходству, называемый в англоязычной литературе fuzzy search).

БД PostgreSQL поддерживает два типа текстовых индексов [18]: GIN (Generalized Inverted Index, обобщённый инвертированный индекс) и GiST (Generalized Search Tree, обобщенное дерево поиска), которые обеспечивают работу с символьными триграммами, что является необходимым условием для использования GIN, который по умолчанию оперирует лексемами. Несмотря на то, что GIN по описанию очень похож на эксперимент с инвертированными строками, GiST тоже имеет основание для применения: его древовидная структура увеличивает полноту поисковой выдачи за счет включения неточных попаданий, что вполне подходит для задачи поиска рифмы, так как в таблице из работы В.М. Жирмунского [13] содержатся, в том числе, пары окончаний, не совпадающих по написанию.

3. Эксперименты

Предполагается провести следующие эксперименты с поиском рифм на БД PostgreSQL:

1. Поиск нужного окончанию среди имен секций: *SELECT * from pg_catalog.pg_tables where %условия на имена секций%*. Стоит отметить, что только в этом эксперименте используются описанные ранее инвертированные строки.

2. Поиск окончаний по неполному совпадению *LIKE* на таблице без индексов.

3. Поиск окончаний по неполному совпадению *LIKE* на таблице с построенным индексом GIN по символьным триграммам: *CREATE INDEX trgm_idx ON test_trgm USING GIN (t gin_trgm_ops)*;
4. Поиск окончаний по неполному совпадению *LIKE* на таблице с построенным индексом GiST по символьным триграммам: *CREATE INDEX trgm_idx ON test_trgm USING GIST (t gist_trgm_ops)*.

Для проведения эксперимента взята минимально возможная выборка из 100 примеров окончаний; 80% выборки состояло из случайно подобранных наиболее часто употребляемых окончаний (первые 500 экземпляров), оставшиеся 20% — примеры из следующих 100 экземпляров употребляемых окончаний (также выбранных в случайном порядке). Измерялись затраты времени на проведение экспериментов по каждому из вариантов (1)–(4). При выполнении каждого эксперимента выведены характеристики (в скобках указаны сокращенные обозначения):

- среднее время выполнения SQL-запроса (avg);
- медиана (mediana);
- 90-я процентиль (90 perc);
- 95-я процентиль (95 perc);
- 98-я процентиль (98 perc).

Результаты приведены в таблице 1.

Таблица 1. Результаты проведения эксперимента.

	avg	*mediana*	*90 perc*	*95 perc*	*98 perc*
Поиск среди имен секций	0,798	0,831	1,704	1,942	2,381
Без индексов	2,497	2,364	3,450	4,109	4,193
GIN	2,258	2,130	2,974	3,343	3,645
GIST	2,407	2,392	3,179	3,330	3,607

По результатам возможно сделать ряд выводов:

– Наименее затратным по времени оказался вариант (1), предполагающий поиск среди имен секций. Такой показатель частично обусловлен экземплярами окончаний, по которым не создавались секции – в таких случаях затраты на выполнение SQL-запроса были ничтожно малыми.

– Результаты поиска без индексов и поиска с использованием индекса GIST отличаются друг от друга в незначительной степени, что говорит о нецелесообразности использования индекса GIST для решения задачи исследования.

– Удовлетворительные результаты показало применение индекса GIN для поиска по неполному совпадению (3).

4. Выводы

Применение встроенных инструментов БД PostgreSQL долгое время было ограничено поисковыми системами в современном их понимании, когда с помощью СУБД возвращались результаты по запросу на естественном языке. Для задачи поиска рифмы быстродействие программы не является определяющим фактором. В настоящей работе были показаны наиболее перспективные подходы, а также приведены примеры, как с помощью простых шагов существенно ускорить работу алгоритма, что дает возможность другим исследователям применить эти подходы в рамках своих исследований без требования экспертных знаний по БД PostgreSQL. Кроме того, интерфейс обращения к СУБД не меняется (за исключением построения таблицы с инвертированными строками вручную), что несомненно удобно для разработчиков, интегрированных с БД PostgreSQL систем анализа текстов.

5. Благодарности
Исследование выполнено при поддержке РНФ (№ 19-18-00466).

6. Литература

[1] Барахнин, В.Б. Об автоматизации комплексного анализа русского поэтического текста / В.Б. Барахнин, О.Ю. Кожемякина // CEUR Workshop Proceedings. – 2012. – Т. 934. – С. 167-171.
[2] Лапшина, Н.В. Метрический справочник к стихотворениям А.С.Пушкина / Н.В. Лапшина, И.К. Романович, Б.И. Ярхо – М., Л.: Academia, 1934. – 144 с.
[3] Mittmann, A. Escansão automático de versos em português // Universidade Federal de Santa Catarina, 2016.
[4] Бойков, В.Н. Об автоматической спецификации стиха в информационно-аналитической системе / В.Н. Бойков, М.С. Каряева, В.А. Соколов, А.И. Пильщиков // CEUR Workshop Proceedings. – 2015. – Т. 1536. – С. 144-151.
[5] Pilshchikov, I. Reconnaissance automatique des mètres des vers russes: Une approche statistique sur corpus / I. Pilshchikov, A. Starostin // Langages. – 2015. – Vol. 3(199). – P. 89-106.
[6] Wikipoetics [Электронный ресурс]. – Режим доступа: http://wikipoetics.ru/ (20.09.2019).
[7] Рифмовед.ру [Электронный ресурс]. – Режим доступа: http://rifmoved.ru/analiz_stihov.htm (20.10.2019).
[8] Размер стиха [Электронный ресурс]. – Режим доступа: http://neogranka.ru/razmer_stiha.html (21.10.2019).
[9] Анализ поэтических текстов онлайн [Электронный ресурс]. – Режим доступа: http://poem.ict.nsc.ru/ (21.10.2019).
[10] Барахнин, В.Б. Алгоритмы комплексного анализа русских поэтических текстов с целью автоматизации процесса создания метрических справочников и конкордансов / В.Б. Барахнин, О.Ю. Кожемякина, А.В. Забайкин // CEUR Workshop Proceedings. – 2015. – Т. 1536. – Р. 138-143.
[11] Barakhnin, V.B. Development and Implementation of the Algorithm for Automatic Analysis of Metrorhythmic Characteristics of Russian Poetic Texts / V.B. Barakhnin, O.Yu. Kozhemyakina, I.V. Kuznetsova // CEUR Workshop Proceedings. – 2019. – Vol. 2523. – P. 290-298.
[12] Большой словарь рифм [Электронный ресурс]. – Режим доступа: http://rifmovnik.ru/docs.htm (21.10.2019).
[13] Жирмунский, В.М. Рифма, ее история и теория / В.М. Жирмунский – М.: Книга по Требованию, 2014. – 342 с.
[14] Зализняк, А.А. Грамматический словарь русского языка: словоизменение: около 10000 слов – М.: Русский язык, 1980. – 880 с.
[15] PostgreSQL: Documentation 9.4: Partitioning [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/docs/9.4/ddl-partitioning.html (22.10.2019).
[16] Collections — Container datatypes [Электронный ресурс]. – Режим доступа: https://docs.python.org/3.7/library/collections.html (22.10.2019).
[17] Transliterate – PyPi [Электронный ресурс]. – Режим доступа: https://pypi.org/project/transliterate/ (22.10.2019).
[18] Postgres Pro Standard: Документация: Типы индексов GIN и GiST [Электронный ресурс]. – Режим доступа: https://postgrespro.ru/docs/postgrespro/9.5/textsearch-indexes (23.10.2019).

Improvements to automated the algorithm definition of rhyme

V.B. Barakhnin[1], O.Yu. Kozhemyakina[1], I.S. Pastushkov[1], I.V. Kuznetsova[1], Yu.S. Borzilova[1]

[1]Institute of Computational Technologies SB RAS, Academician Lavrentiev Ave. 6, Novosibirsk, Russia, 630090

Abstract. The paper considers approaches to the improvement of one of the steps of the algorithm used for the automated determination of rhyme in poetic texts. The automated rhyme detection tool is one of the modules of the system of complex analysis of poetic texts. In the current module implementation, the rhyme search and definition subtask is solved by finding words with consonant endings using the A. A. Zaliznyak Grammar Dictionary of the Russian Language and the basic rules of phonetic analysis. Alternative solutions to the search problem in the dictionary of words with consonant endings are proposed. The results obtained will allow us to conclude that the current implementation is optimistic and the methods used can be finalized to solve the problems of determining the rhyme of a poetic text.