СБОРНИК **НАУЧНЫХ ТРУДОВ НГТУ. – 2018. – № 3–4 (93).** – 59–69

УДК 681.513 DOI: 10.17212/2307-6879-2018-3-4-59-69

ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ АЛГОРИТМОВ НЕЧЕТКОГО ПОИСКА*

А.В. ЛЕЩЕНКО

630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, студент факультета автоматики и вычислительной техники. E-mail: siberianhunger@gmail.com

В статье обозреваются некоторые популярные алгоритмы нечеткого поиска (approximate string matching, fuzzy search). Объясняются основные понятия, на которых основаны алгоритмы. С помощью псевдокода и схем отображается их формальная сторона. Рассматривается реализация одного из приведенных алгоритмов для объединенного решения агрегации digital-каналов, маршрутизации клиентов и геймификации рабочего пространства оператора сотовой связи. Программа, которая использует алгоритм нечеткого поиска и организует рабочее пространство оператора, была написана на хакатоне (соревнование по программированию), который состоялся на базе научной библиотеки НГТУ по заказу компании «Мегафон» 5–7 октября 2018 года командой Infinite Capacity. Программа создана на основе обширного стека технологий: вся серверная часть написана на JavaScript, алгоритм нечеткого поиска также представлен фреймворком elasticsearch для JavaScript(NoteJS), базы данных реализованы на PostgreSQL, сообщение с клиентами сотового оператора организовано с использованием ботов для социальных сетей(VK BOT api, TG BOT api). Ссылка на открытый код программы: https://github.com/JackMor/HKTgit. Программа с помощью объединения каналов связи из социальных сетей предоставляла информацию о клиентах оператору сотовой связи, и на основе имеющейся базы данных ответов для клиента алгоритм (fuzzy search) подбирал наиболее подходящие ответы, из которых оператор выбирал соответствующий его требованиям. Основное внимание статьи уделено алгоритму fuzzy search. Для лучшего понимания алгоритма предоставлены его формальная и схематическая схемы. Также в литературных источниках можно найти руководство к фреймворку elasticsearch, который основан на исходном алгоритме.

Ключевые слова: расстояние Левенштейна, редакционное расстояние, дистанция редактирования, алгоритм нечеткой логики, алгоритм нечеткого поиска, fuzzy search, алгоритм Нидлмана–Вунша, расстояние Дамерау–Левенштейна, approximate string matching

* Статья получена 24 октября 2018 г.

ВВЕДЕНИЕ

Рассмотрим определение алгоритма нечеткого поиска. Нечеткий поиск – это поиск информации, при котором выполняется сопоставление информации заданному образцу поиска или близкому к этому образцу значению. Алгоритмы нечеткого поиска используются в большинстве современных поисковых систем (например, для проверки орфографии) [1].

Также не помешает дать формальное определение алгоритма. Пусть $\sum$ – конечное множество (алфавит) размера $|\sum| = \sigma$.

Пусть $T \in \sum^*$ – текст длиной $n = |T|$.

Пусть $P \in \sum^*$ – образец длиной $m = |P|$.

Пусть $k \in \mathbb{R}$ – максимально разрешенное количество ошибок.

Пусть $d : \sum^* \times \sum^* \to \mathbb{R}$ – функция расстояния.

Тогда

Задача: даны T,P,k и d(.), которые возвращают множества всех позиций текста j таких, что существует i такое, что $d(P,T_{i..j}) \le k$ [2].

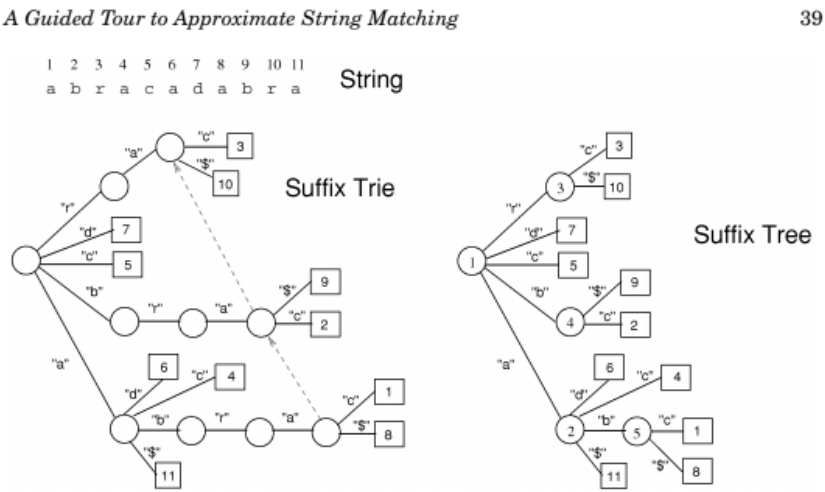

Рис. 1. Дерево окончаний для примера строки, $ – специальный символ для обозначения конца текста. Внутренние узлы дерева окончаний показывают позицию символа для нахождения в строке [3]

В упрощенном виде задачу алгоритма можно описать так: по заданному слову найти в тексте с размером n все слова, совпадающие с этим словом (или начинающиеся с этого слова), учитывая k возможных различий (неточностей).

После рассмотрения определения алгоритма нечеткого поиска рассмотрим основные необходимые понятия и несколько видов алгоритма с их характеристиками, далее будут приведены примеры практической реализации алгоритма.

1. РАССТОЯНИЕ ЛЕВЕНШТЕЙНА (LEVENSHTEIN DISTANCE)

Расстояние Левенштейна (Levenshtein distance) между двумя строками в теории информации и компьютерной лингвистике – это минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

Рекуррентная формула, описывающая расстояние Левенштейна:

$$\mathrm{d}(\mathrm{S}_1, \mathrm{S}_2) = \mathrm{D}(\mathrm{M}, \mathrm{N}) \text{ , где}$$

$$D(i,j) = \begin{cases} 0 & ;\ i = 0,\ j = 0 \\ i & ;\ j = 0,\ i > 0 \\ j & ;\ i = 0,\ j > 0 \\ D(i-1, j-1) & ;\ S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ;\ j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

$\min(a, b, c)$ возвращает наименьший из аргументов.

Рис. 2. Описание расстояния Левенштейна

Элементы строк нумеруются с первого, как принято в математике, а не нулевого; S1 и S2 – две строки (длиной M и N соответственно) в некотором алфавите, где редакционное расстояние d(S1,S2).

Свойства расстояния Левенштейна:

1) $d(S1,S2) \geqslant ||S1| - |S2||$

2) $d(S1,S2) \leqslant \max(|S1|,|S2|)$

3) $d(S1,S2) = 0 \Leftrightarrow S1 = S2$,

где d(S1,S2) – расстояние Левенштейна между строками S1 и S2, а |S| – длина строки S.

Также расстояние Левенштейна является метрикой.

2. ПРИМЕР 1: Алгоритм Вагнера–Фишера

Для нахождения кратчайшего расстояния необходимо вычислить матрицу D, используя вышеприведенную формулу. Ее можно вычислять как по строкам, так и по столбцам. Псевдокод алгоритма написан при произвольных ценах замен, вставок и удалений (важно помнить, что элементы нумеруются с первого). Псевдокод ниже решает простой частный случай, когда вставка символа, удаление символа и замена одного символа на другой стоят одинаково для любых символов [4, 5].

```
int levensteinInstruction(String s1, String s2, int In-
sertCost, int DeleteCost, int ReplaceCost):
  D[0][0] = 0
  for j = 1 to N
    D[0][j] = D[0][j - 1] + InsertCost
  for i = 1 to M
    D[i][0] = D[i - 1][0] + DeleteCost
    for j = 1 to N
      if S1[i] != S2[j]
         D[i][j] = min(D[i - 1][j] + DeleteCost,
         D[i][j - 1] + InsertCost,
         D[i - 1][j - 1] + ReplaceCost)
      else
         D[i][j] = D[i - 1][j - 1]
  return D[M][N]
```

3. ПРИМЕР 2: Метод динамического программирования Вагнера и Фишера

В методе динамического программирования последовательно, по предыдущим значениям, вычисляются расстояния между более длинными префиксами двух строк до получения окончательного результата. Опишем этот процесс более подробно.

Пусть $d_{i,j}$ есть расстояние между префиксами строк x и y, длины которых равны соответственно i и j, то есть

$$d_{i,j} = d(x(1, i), y(1, j))$$

Цену преобразования символа a в символ b обозначим через w(a,b). Таким образом, w(a,b) – это цена замены одного символа на другой, когда $a \neq b$, w(a,ε) – цена удаления a, а w(ε, b) – цена вставки b. Заметим, что в случае, когда выполнены нижеследующие условия, d является расстоянием Левенштейна:

$w(a, \varepsilon) = 1$

$w(\varepsilon, b) = 1$

$w(a, b) = 1$, если $a \neq b$,

$w(a, b) = 0$, если $a = b$

В процессе вычислений значения $d_{i,j}$ записываются в массив $(m+1) \times (n+1)$, а вычисляются они с помощью следующего рекуррентного соотношения:

$$d_{i,j} = \min\{d_{i-1,j} + w(x_i, \varepsilon), d_{i,j-1} + w(\varepsilon, y_j), d_{i-1,j-1} + w(x_i, y_i)\}$$

Оно выводится следующим образом. Если предположить, что известна цена преобразования x(1, i–1) в y(1, j), то цену преобразования x(1, i) в y(1, j) мы получим, добавив к ней цену удаления x_i. Аналогично цену преобразования x(1, i) в y(1, j) можно получить, прибавив цену вставки y_j к цене преобразования x(1, i) в y(1, j–1). Наконец, зная цену преобразования x(1, i–1) в y(1, j–1), цену преобразования x(1, i) в y(1, j) мы получим, прибавив к ней цену замены x_i на y_j. Вспомним, что расстояние $d_{i,j}$ является минимальной ценой преобразования x(1, j) в y(1, j), поэтому из трех указанных выше операций надо выбрать самую дешевую.

Перед тем как начать вычислять $d_{i,j}$, надо установить граничные значения массива. Что касается первого столбца массива, то значение $d_{i,0}$ равно сумме цен удаления первых i символов x. Аналогично значения $d_{0,j}$ первой строки задаются суммой цен вставки первых j символов y. Итак, имеем следующее:

$$d_{0,0} = 0$$

$$d_{i,o} = \sum_{k=1}^{i} w(x_i, \varepsilon) \quad \text{для } 1 < i < m$$

$$d_{o,j} = \sum_{k=1}^{j} w(\varepsilon, j_k) \quad \text{для } 1 < j < n$$

Рис. 3. Префиксное расстояние

Для расстояния Левенштейна $d_{i,0} = i$ и $d_{0,j} = j$. Ниже приведен массив, полученный при вычислении расстояния Левенштейна между строками *preterit* и *zeitgeist*. Из него видно, что расстояние между этими строками, т. е. $d_{8,9}$, равно 6 [6].

Расстояние Левенштейна между строками preterit и zeitgeist

	j	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
i			z	e	i	t	g	e	i	s	t
0		0	1	2	3	4	5	6	7	8	9
1	p	1	1	2	3	4	5	6	7	8	9
2	r	2	2	2	3	4	5	6	7	8	9
3	e	3	3	2	3	4	5	5	6	7	8
4	t	4	4	3	3	3	4	5	6	7	7
5	e	5	5	4	4	4	4	4	5	6	7
6	r	6	6	5	5	5	5	5	5	6	7
7	i	7	7	6	5	6	6	6	5	6	7
8	t	8	8	7	6	5	6	7	6	6	6

$X_з$	$X_у$
$X_в$	$D_{i,j}$

$$D_{ij} = min\,(X_в + 1, X_у + 1, X_з + C_{замены})$$

$$C_{замены} = \begin{cases} 1, & если\ S_1[i] \neq S_2[j] \\ 0, & иначе \end{cases}$$

Рис. 4. Итоговый вывод формулы расчета расстояния Левенштейна

Конец примера.

4. РЕАЛИЗАЦИЯ АЛГОРИТМА НЕЧЕТКОГО ПОИСКА В ELASTICSEARCH

Фреймворк, который был использован для создания программы, организующей рабочее пространство оператора сотовой связи – elasticsearch, основывается на такой метрике, как расстояние Дамерау–Левенштейна. Описание этой метрики представлено ниже.

$X_т$		
	$X_з$	$X_у$
	$X_в$	$D_{i,j}$

$$D_{ij} = min(X_в + 1, X_у + 1, X_з + C_{замены}, X_т + C_{транспозиции})$$

$$C_{замены} = \begin{cases} 1, & если\ S_1[i] \neq S_2[j] \\ 0, & иначе \end{cases}$$

$$C_{транспозиции} = \begin{cases} 1, & если\ \ S_1[i] = S_2[j-1]\ \ и\ \ S_1[i-1] = S_2[j] \\ \infty, & иначе \end{cases}$$

Рис. 5. Описание метрики Дамерау–Левенштейна

Чтобы вычислять такое расстояние, достаточно немного модифицировать алгоритм нахождения обычного расстояния Левенштейна следующим образом: хранить не две, а три последние строки матрицы, а также добавить соответствующее дополнительное условие: в случае обнаружения транспозиции при расчете расстояния также учитывать и ее стоимость.

Далее показана схема, представляющая работу алгоритма нечеткого поиска уже конкретно в elasticksearch [7, 10].

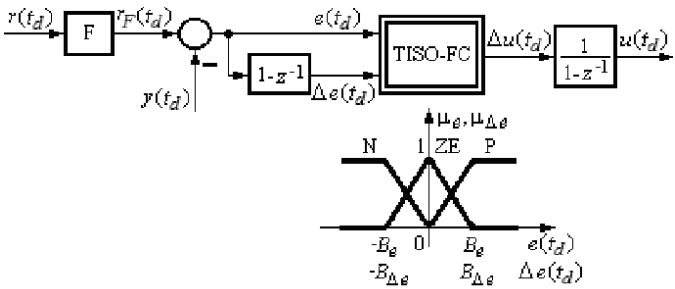

Рис. 6. Формальное схематическое отображение алгоритма нечеткого поиска

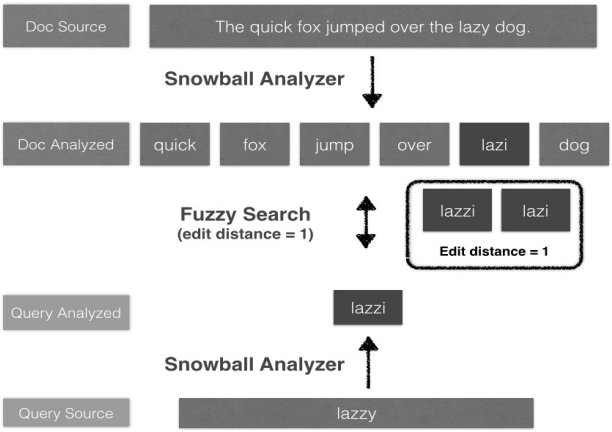

Рис. 7. Схематическое изображения алгоритма Fuzzy search

Из двух схем, указанных выше, можно сделать вывод о том, что эффективность алгоритмов нечеткого поиска напрямую зависит от редакционного расстояния (edit distance). Чем выше этот параметр, тем менее эффективен алгоритм [9].

ЗАКЛЮЧЕНИЕ

Алгоритмы нечеткого поиска применяются в обширном спектре современных технологий. Практическое использование алгоритмов нечеткого поиска в реальных поисковых системах тесно связано с фонетическими алгоритмами, алгоритмами лексического стемминга – выделения базовой части у различных словоформ одного и того же слова (например, такую функциональность предоставляют Snowball и Яндекс mystem [8]), а также с ранжированием на основе статистической информации либо же с использованием сложных изощренных метрик.

Также существует множество неэффективных алгоритмов нечеткого поиска, которые лучше обходить стороной.

СПИСОК ЛИТЕРАТУРЫ

1. *Мосалев П.М.* Обзор методов нечеткого поиска текстовой информации // Вестник МГУП имени Ивана Федорова. – 2013. – № 2. – С. 87–91.

2. Indexing methods for approximate string matching / G. Navarro, R. Baeza-Yates, E. Sutinen, J. Tarhio // IEEE Data Engineering Bulletin. – 2001. – Vol. 24, N 4. – P. 19–27.

3. *Navarro G.* A guided tour to approximate string matching // ACM Computing Surveys. – 2001. – Vol. 33, N 1. – P. 31–88.

4. Задача о редакционном расстоянии, алгоритм Вагнера–Фишера [Электронный ресурс]. – URL: http://neerc.ifmo.ru/wiki/index.php?title=Задача_о_редакционном_расстоянии,_алгоритм_Вагнера–Фишера (дата обращения: 13.03.2019).

5. *Lowrance R., Wagner R.A.* An extension of the string-to-string correction problem // Journal of the ACM. – 1975. – Vol. 22, N 2. – P. 177–183.

6. Метод динамического программирования Вагнера и Фишера [Электронный ресурс]. – URL: http://algolist.manual.ru/search/lcs/vagner.php (дата обращения: 13.03.2019).

7. *Cholakian A.* How to use fuzzy searches in elasticsearch [Electronic resource]. – URL: www.elastic.co/blog/found-fuzzy-search (accessed: 13.03.2019).

8. Нечеткий поиск в тексте и словаре [Электронный ресурс]. – URL: https://habr.com/post/114997/ (дата обращения: 13.03.2019).

9. *Vernica R., Li C.* Efficient top-k algorithms for fuzzy search in string collections // Proceedings of the First International Workshop on keyword search on structured data, 28 June 2009. – New York: ASM, 2009. – P. 9–14.

10. *Cholakian A.* A human-friendly tutorial for elasticsearch [Electronic resource]. – URL: http://exploringelasticsearch.com (accessed: 13.03.2019).

Лещенко Андрей Владимирович, студент факультета автоматики и вычислительной техники Новосибирского государственного технического университета. E-mail: siberianhunger@gmail.com

DOI: 10.17212/2307-6879-2018-3-4-59-69

Overview and practical implementation of fuzzy search algorithms*

A.V. Leshchenko

Novosibirsk State Technical University, 20 K. Marx Prospekt, Novosibirsk, 630073, Russian Federation, D. Sc. (Eng.). E-mail: ucit@ucit.ru

Fuzzy search and approximate search algorithms were reviewed in this article. All the fundamental definitions, that helping to better understand the concept of fuzzy search are noted below. Also was shown how to practically implement them to set up a work flow for mobile network operator. Program that have been shown in the implementation, now available via github linkhttps://github.com/JackMor/HKTgit. This program have been written by computer programmers team called “Infinite Capacity” on 2018 hackathon for Megafon, that have been placed in The library of Novosibirsk State Technical University . Wide spectrum of technology was used to create this program: JavaScript(NodeJS) for server and also fuzzy search was implemented with elasticsearch(JS framework), PostgreSQL for database management, VK BOT API and TG BOT API for the social media interaction. For better understanding of approximate search provided formal and schematic representations of the algorithm. Also you can find pseudocode explanation of Needleman–Wunsch algorithm below. Damerau–Levenshtein distance presented with the exact example of string approximate coparison. References are containing full guide to elasticsearch framework user, that might help to understand the conception of fuzzy search too. In the end of the article you can find a conclusion with the analysis of the future for approximate search algorithms.

Keywords: Approximate string matching, fuzzy search, Damerau–Levenshtein distance, Needleman–Wunsch algorithm, elasticsearch, Levenshtein distance, edit distance

REFERENCES

1. Mosalev P.M. Obzor metodov nechetkogo poiska tekstovoi informatsii [An overview of fuzzy text search methods]. *Vestnik MGUP imeni Ivana Fedorova – Vestnik MGUP by Ivan Fedorov*, 2013, no. 2, pp. 87–91.

2. Navarro G., Baeza-Yates R., Sutinen E., Tarhio J. Indexing methods for approximate string matching. *IEEE Data Engineering Bulletin*, 2001, vol. 24, no. 4, pp. 19–27.

3. Navarro G. A guided tour to approximate string matching. *ACM Computing Surveys*, 2001, vol. 33, no. 1, pp. 31–88.

4. *Zadacha o redaktsionnom rasstoyanii, algoritm Vagnera-Fishera* [The editorial distance problem, Wagner-Fisher algorithm]. Available at: http://neerc.ifmo.ru/wiki/index.php?title=Задача_о_редакционном_расстоянии,_алгоритм_Вагнера-Фишера (accessed 13.03.2019).

* *Received 24 October 2018.*

5. Lowrance R., Wagner R.A. An extension of the string-to-string correction problem. *Journal of the ACM*, 1975, vol. 22, no. 2, pp. 177–183.

6. *Metod dinamicheskogo programmirovaniya Vagnera i Fishera* [Wagner and Fisher dynamic programming method]. Available at: http://algolist.manual.ru/search/lcs/vagner.php (accessed 13.03.2019).

7. Cholakian A. *How to use fuzzy searches in elasticsearch*. Available at: www.elastic.co/blog/found-fuzzy-search (accessed 13.03.2019).

8. *Nechetkii poisk v tekste i slovare* [Fuzzy text and dictionary search]. Available at: https://habr.com/post/114997/ (accessed 13.03.2019).

9. Vernica R., Li C. Efficient top-k algorithms for fuzzy search in string collections. *Proceedings of the First International Workshop on keyword search on structured data*, 28 June 2009. New York, ASM, 2009, pp. 9–14.

10. Cholakian A. *A human-friendly tutorial for elasticsearch*. Available at: http://exploringelasticsearch.com (accessed 13.03.2019).

Для цитирования:

Лещенко А.В. Обзор и практическое применение алгоритмов нечеткого поиска // Сборник научных трудов НГТУ. – 2018. – № 3–4 (93). – С. 59–69. – DOI: 10.17212/2307-6879-2018-3-4-59-69.

For citation:

Leshchenko A.V. Obzor i prakticheskoe primenenie algoritmov nechetkogo poiska [Overview and practical implementation of fuzzy search algorithms]. *Sbornik nauchnykh trudov Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Transaction of scientific papers of the Novosibirsk state technical university*, 2018, no. 3–4 (93), pp. 59–69. DOI: 10.17212/2307-6879-2018-3-4-59-69.