УДК 519.6+004.9

СРАВНЕНИЕ АЛГОРИТМОВ НЕЧЕТКОГО ПОИСКА ПРИ ОЦЕНКЕ СФОРМИРОВАННЫХ СЛОВ ПО СЛОГОВОЙ МОДЕЛИ В СИСТЕМЕ РАСПОЗНОВАНИЯ СЛИТНОЙ РУССКОЙ РЕЧИ

Спирина А.В.,
научный руководитель д-р техн. наук Семенкин Е.С., науч. сотр. Института информационных технологий Ульмского университета, Заблоцкий С. Г.
Институт информатики и телекоммуникаций
Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева

На сегодняшний день направление автоматизированного распознавания речи является одним из самых быстро развивающихся направлений. Системы автоматизированного распознавания речи находят свое применение в различных сферах человеческой деятельности: компьютерные приложения, web-приложения, применение голосовых команд для управления средствами передвижения, замена ручного набора текста голосовым.

Словарь для распознавания слитной русской речи значительно больше, чем для распознавания слитной английской речи, так как русский язык обладает большим количеством различных словоформ, в связи с этим традиционные пословные модели языка оказываются неприменимыми.

При распознавании акустический сигнал преобразуется в последовательность слогов, но границы слов не известны. Задача распознавания слитной речи состоит в правильном объединении слогов в слова.

Существуют подходы для решения этой задачи, но, тем не менее, требуется дальнейшее улучшение используемых алгоритмов с целью ускорения их работы и повышения точности распознавания.

Для объединения слогов в слова используется генетический алгоритм. Прежде всего, необходимо оценить полученные предложения. Чтобы ускорить этот этап, нужно ускорить поиск сформированного генетическим алгоритмом слова по имеющемуся словарю.

Для сравнения были выбраны два алгоритма нечеткого поиска: расстояние Левенштейна и функция релевантности.

Была реализована функция релевантности, рассмотренная в диссертации “Разработка математического и программного обеспечения идентификации объектов в базе данных на основе нестрогого соответствия” Карахтанова Д.С. 2011 года.

Расчет значения функции релевантности производится по формуле (1), (2):

$$R = \frac{\sum_{i=1}^{N} r(i)}{N}; \quad (1)$$

$$r(i) = \frac{Match(Str1, Str2, i) + Match(Str2, Str1, i)}{Count(Str1, i) + Count(Str2, i)}, \quad (2)$$

где $Count(Str,i) = (len(Str)-i+1)$;

$len(S)$ – длина строки S;

$Match(S_1, S_2, i)$ – сумма совпадений всех подстрок длиной i из S_1 в строке S_2.

Расстоянием Левенштейна называется минимальное количество операций вставки, удаления и замены символа, необходимых для преобразования одной строки в другую.

Пусть S_1 и S_2 - строки, тогда расстояние Левенштейна будет вычисляться по формуле (3).

$$D_{i,j} = \begin{cases} 0, & i = 0, j = 0 \\ i, & i > 0, j = 0 \\ j, & i = 0, j > 0 \\ \min(D_{i,j-1} + 1, D_{i-1,j} + 1, X_{i-1,j-1} + C_{замены}), & i > 0, j > 0 \end{cases} \quad (3)$$

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

Тестирование алгоритмов проводилось на словаре, состоящем из 100.000 слов, хранящихся в программе в виде массива слов. Кроме того, для сравнения с исходным словом из словаря выбирались слова, длина которых отличалась не более, чем на 2 от длины исходного слова. Для словаря, состоящего из 100.000 слов, распределение слов по длине представлено на рисунке 1:

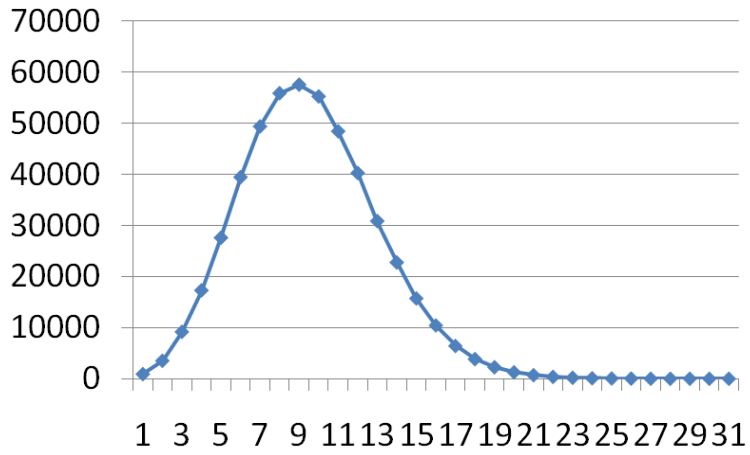

Рисунок 1- График распределение количества слов из словаря в зависимости от длины исходного слова

На рисунке 1 по горизонтальной оси расположены длины исходного слова, на вертикальной оси количество слов из словаря с длиной, отличающейся не более чем на 2 от длины исходного слова.

При тестировании скорости нечеткого поиска с помощью расстояния Левенштейна и функции релевантности с параметрами n=1, n=2, n=3 b n=4 на словаре 100.000 слов, хранящихся в виде вектора в программе, были получены результаты, представленные на рисунке 2.

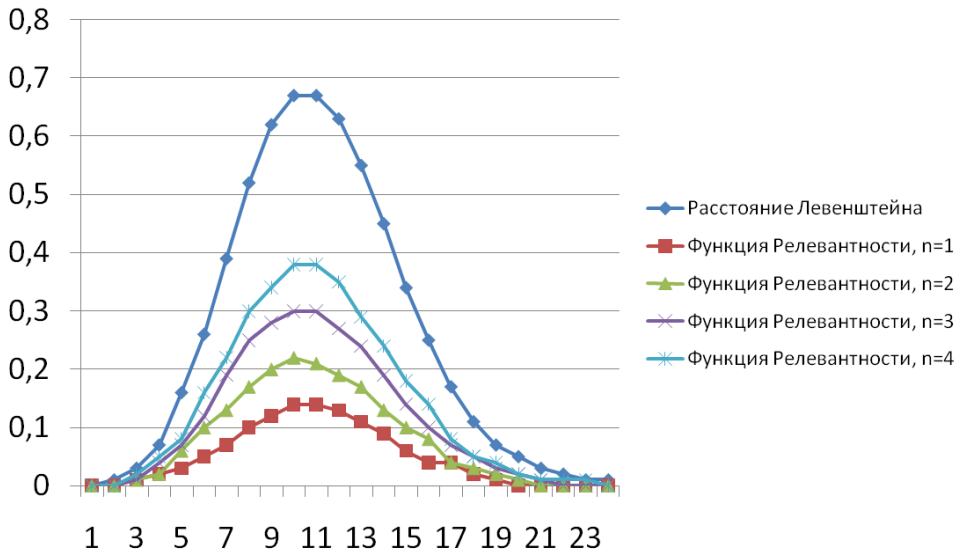

Рисунок 2 – График скорости поиска исходного слова по усеченному словарю

На рисунке 2 по горизонтальной оси расположена длина исходного слова, по вертикальной оси – время поиска в секундах.

Все расчеты проводились на компьютерах Ульмского университета (на базе процессора Intel(R) Core(TM)2 CPU 6700 @ 2.66GHz, в системе Linux).

В дальнейшем весь словарь хранится в префиксном дереве trie. В связи с этим была предложена модификация функции релевантности. Идея модификации заключается в том, что при получении очередного слова не надо вычислять функцию релевантности заново. На каждом шаге, при добавлении очередной буквы к слову, необходимо вычислить только коэффициенты схожести на этом шаге.

Таким образом, на каждом шаге мы используем ранее посчитанные значения, а не вычисляем заново. Кроме того, при поиске по дереву, высчитываем значение функции релевантности только тогда, когда длина слова отличается от длины исходного слова не более чем на 2. Причем до этого момента (|длина слова из словаря – длина исходного слова| ≤ 2), коэффициенты похожести вычисляются только в одну сторону, то есть на подстроки разбивается только слово из словаря и ищется в исходном слове. При достижении определенной длины слова из словаря высчитываются коэффициенты похожести в обратную сторону, то есть разбивается на подстроки исходное слово, после чего высчитывается значение функции релевантности.

Так же для разных длин слов лучше использовать определенный параметр функции релевантности. Например, если длина слова меньше 6, можно использовать параметр n=2, так для относительно коротких слов применение функции релевантности с параметром n=4 увеличивает время работы алгоритма при практически одинаковом результате.

При тестировании не на дереве, а просто на определенном слове время работы простой функции релевантности и усеченной (то есть без поиска вхождений подстрок исходного слова в слове из словаря) рекурсивной функции релевантности получились следующие: усеченная рекурсивная функция релевантности работает в 5-6 раз быстрее, чем исходная функция релевантности.

Исходя из полученных результатов, можно сделать вывод, что функция релевантности, а также ее модификация, возможно, позволят улучшить скорость нечеткого поиска по словарю для слова, сформированного с помощью генетического алгоритма.