УДК 512.8, 681.55

О. В. Герман, кандидат технических наук, доцент (БГТУ);
Н. И. Гурин, кандидат физико-математических наук, доцент (БГТУ);
Ю. О. Герман, ассистент (БНТУ)

ЗАДАЧА КОРРЕКЦИИ РАСПОЗНАВАНИЯ РЕЧИ ДЛЯ ОБРАБОТКИ СЕМАНТИЧЕСКИМ АНАЛИЗАТОРОМ

В статье рассмотрена формальная задача коррекции ошибочно введенных слов речевого запроса к семантической базе знаний электронного интерактивного учебника. Предлагается алгоритм восстановления текста вопроса с выполнением поиска ответа на B-дереве, узлы которого представляют ключевые понятия предметной области электронного учебника. B-дерево проиндексировано по интервальным значениям средневзвешенных ASCII-кодов букв, образующих ключевые слова. Эти значения используются в качестве индексов, позволяющих организовать выбор следующих узлов B-дерева при неудачном сравнении слова запроса и слова в текущем узле дерева. Алгоритм доведен до программной реализации, позволяющей указать его вычислительные характеристики.

The paper considers a formal specification of a problem connected to restoration of incorrectly written words forming a verbal question to the knowledge base of interactive electronic tutorial. It is proposed an algorithm to recover the text in question with the performance of finding the answer to B-tree whose nodes represent key concepts in the field of e-textbook. B-tree is indexed by means of the ASCII-values intervals corresponding to the key words of the subject area. This ASCII-values intervals enable one to organize further searching in the case of unsuccessful comparison of the word from the question with the word associated to a currently observed tree node. The algorithm is realized programmatically what makes possible to estimate its behavior.

Введение. При реализации электронного интерактивного учебника предполагается возможность речевого взаимодействия. В частности, следует отметить пакет SAPI5.2, предназначенный для интеграции в FrameWork 3.5 NET и позволяющий переводить звуковую речь в предложения и наоборот. При этом иногда возникает проблема точности перевода, например, связанная с нечеткой дикцией говорящего. В связи с этим стоит задача коррекции вопросов, вводимых голосом. Таким образом, стоит двуединая задача: восстановить текст вопроса и найти релевантный вопросу ответ. Обе эти задачи решаются одновременно. Из известных алгоритмов можно отметить [1, 2]. В настоящей работе предлагается оригинальный метод, соединяющий механизм ассоциативного поиска и поиска на B-дереве. Использование ассоциативного механизма позволяет сделать поиск устойчивым к ошибкам в записи слов. B-дерево дает возможность выполнить поиск быстро. Указанные моменты отличают предлагаемый алгоритм от указанных выше.

Идентификация слов с ошибками. Для идентификации слов нам необходима метрика (функция) для измерения степени схожести слов. Хорошо известны метрики Левенштейна, Левенштейна – Дамерау, Жаккарда и др. (например, метрика Евклида или Махаланобиса). Эти метрики не принимают во внимание смысловую основу слова, которая передается «каркасом» слова, составленным из согласных букв. Для учета указанного обстоятельства рассматриваем два слова – искомое (оригинальное), а также «*скелетон*» искомого слова, из которого выброшены гласные буквы. Для обоих вариантов слов производим разбиение на биграммы и вычисляем оценку сходства с эталоном по формуле

$$\alpha = n_{1,2} / n_{\text{search\_pattern}}, \qquad (1)$$

где $n_{1,2}$ – число совпадающих биграмм в списках биграмм искомого и эталонного слов, $n_{\text{search\_pattern}}$ – размер списка биграмм искомого слова. Результирующая оценка степени сходства слов выполняется по формуле

$$\begin{gathered} \alpha = \lambda_1 \cdot \alpha_1 + \lambda_2 \cdot \alpha_2, \\ \lambda_1 + \lambda_2 = 1, \quad \lambda_1, \quad \lambda_2 \geq 0, \end{gathered} \qquad (2)$$

где λ_1, λ_2 определяют приоритеты (веса) для критериев совпадения исходного слова с эталоном (α_1) и скелетона искомого слова со скелетоном эталона (α_2).

Приведем пример. Пусть искомое слово (оригинал) записано как *cupp*. Эталон есть *cup*. Список биграмм слова *cupp* есть [*cu*, *up*, *pp*]. Список биграмм слова *cup* есть [*cu*, *up*]. Оценка $\alpha_1 = 0.667$. Для скелетонов слов (*cpp* и *cp*) имеем: $\alpha_2 = 0.5$. Для практического применения экспериментально установлено, что наиболее эффективное распознавание достигается для $\lambda_1 = 0.4$, $\lambda_2 = 0.6$. Слово считается распознанным, если оценка (2) не ниже 0.57 (как это принято в теории принятия решений при использовании функций полезности). Таким образом,

мы располагаем формальным механизмом проверки идентичности слов в узлах B-дерева. Следует далее определить, в каком направлении выполнять дальнейший поиск при неудачном сравнении слов в текущем узле. Для решения этой задачи нами привлекается механизм хэш-функций, используемый в системах ассоциативного поиска.

Выбор направления в B-дереве. Теперь мы представим основную идею нашего поискового алгоритма. Эта идея основана на использовании хэш-кодов слов. В качестве функции хэширования применяем среднее арифметическое ASCII-кодов букв, образующих слово. При этом используем скелетоны слов, т. е. опускаем гласные буквы. В качестве примера приведем хэш-коды слов, найденные по представленному выше способу (см. табл. 1).

Таблица 1

Хэш-коды слов

Слово	Скелетон	Хэш-код
apple	pl	110.0
plum	plm	109.667
strawberry	strwbr	112.667
pear	pr	113.0
banana	bnn	106.0
potato	ptt	114.667

Предполагается, что использование среднего арифметического значения ASCII-кода слова более устойчиво к изменениям и ошибкам при написании слова. Теперь мы строим B-дерево, которое в качестве индекса использует хэш-код скелетона слова. Пусть, например, выполняется поиск слова по шаблону *ptas* (фрагмент слова *potatoes*). Поиск начинается с корневой вершины B-дерева, где записано слово *strawberry*.

Сначала выполняется идентификация слова с помощью техники на основе формул (1, 2) предыдущего раздела. Если интегральная оценка α не меньше 0.57, то слово найдено. В противном случае вычисляется средний арифметический код скелетона слова *ptas* (в нашем примере для слова *pts* он равен 114.33). Выполняется сравнение со средним ASCII-кодом слова *strwbr* (*strawberry*), равным 112.667. Из этого сравнения делается вывод о выборе ветки с корневой вершиной *pears*. Далее действия повторяются. В итоге выбирается слово *potatoes.*

Использование средних значений ASCII-кодов имеет серьезный недостаток: при наличии ошибок в слове разброс средних значений ASCII-кодов игнорировать нельзя. Иначе говоря, следует перейти от средних значений ASCII-кодов к интервальным значениям $[\alpha, \beta]$. Значения α, β определяются экспериментально.

Алгоритм поиска слов с ошибками. Теперь мы представим основную идею нашего поискового алгоритма, который можно резюмировать следующим образом для поискового дерева, вершины которого ассоциированы с интервальными значениями хэш-функций.

Шаг 1. Ввести искомое слово *s*. Найти его скелетон *sc*. Найти соответствующие *s* и *sc* списки биграмм *LS* и *LSC*. В качестве текущей вершины B-дерева взять корневую.

Шаг 2. Пусть текущей вершине B-дерева соответствует слово *w* и скелетон *wc*. Выполнить идентификацию по формулам (1), (2).

Шаг 3. Если интегральная оценка α не ниже 0.57, то завершить процесс поиска с выдачей в качестве ответа слова *w*. В противном случае – следующий шаг.

Шаг 4. Вычислить средний ASCII-код скелетона для выбора перехода по B-дереву из текущей вершины. Определить следующую текущую вершину. Новая вершина определяется таким образом. Пусть среднее значение ASCII-кода искомого слова составляет *z*. Пусть с текущей вершиной дерева ассоциирован подинтервал $[\alpha, \beta]$. Если $z < \alpha$, то очередная текущая вершина выбирается как левая дочерняя вершина. Если $z > \beta$, то очередная текущая вершина выбирается как правая дочерняя вершина. Если ни одно из указанных соотношений не выполняется, то поиск завершается ответом «nil».

Шаг 5. Текущая вершина nil? Если ДА, то конец – алгоритм не дал результата, т. к. поисковое слово не идентифицировано (назовем этот исход «случай nil».) В противном случае – переход к шагу 2.

Обработка вопроса на B-дереве. Рассмотрим некорректно представленный вопрос: «Чем параметризуется АДС?»

Данная фраза разбивается на лексемы: Чем, параметризуется, АДС. Вопросительные слова не используются в поиске по дереву. Поэтому поиск ведется по двум лексемам: «параметризуется» и «АДС». Особенностью поиска является следующее:

– поиск ведется по нескольким словам одновременно;

– слова могут быть искажены.

Реализация поиска по нескольким словам при техническом исполнении требует лишь породить несколько потоков (THREAD), одновременно сканирующих дерево, начиная с корня. В нашем случае будут организованы два потока – один поток для слова «АДС», второй – для слова «параметризуется». Цель каждого потока – найти узел, соответствующий данному слову. Для слова *параметризуется* смысловой каркас задается как *прмтрзтс*. Аналогично, для *АДС* это будет *ДС*.

В нашем случае рассмотрим узел ЭДС. Слово ЭДС характеризуется следующим списком биграмм [ЭД, ДС]. Слово АДС характеризуется списком биграмм вида [АД, ДС]. Если не рассматривать гласные, то списки одинаковы – [ДС]. Вычислим раздельно, какая часть списка биграмм совпадает как в случае слов с гласными, так и в случае слов с выброшенными гласными: $\alpha_1 = 0.5$, $\alpha_2 = 1$. Принимаем $\lambda_1 = 0.4$, $\lambda_2 = 0.6$. Отсюда $\alpha = 0.4 \cdot 0.5 + 0.6 \cdot 1 = 0.8$.

Узел распознается, если значение интегральной метрики не ниже 0.57. В нашем случае узел *ЭДС* распознан. Никакой узел не сопоставлен слову «*параметризуется*». Предположим, что в узле ЭДС имеется два предложения:

clause("*ЭДС зависит от температуры по линейному закону* E = a + bT"),

clause("*ЭДС характеризуется температурным коэффициентом*").

Далее надо проверить, ассоциируется ли слово *параметризуется* с каким-либо из слов в этих предложениях. Опять используем метрику (1). Так, для слова *характеризуется* имеем:

[ха, ар, ра, ак, кт, те, ер, ри, из, зу, уе, ет, тс, ся].

Слово *хрктрзтс* имеет следующий список биграмм – [хр, рк, кт, тр, рз, зт, тс]. Аналогично, *параметризуется* – [па, ар, ра, ам, ме, ет, тр, ри, из, зу, уе, ет, тс, ся]; *прмтрзтс* – [пр, рм, мт, тр, рз, зт, тс]. Находим: $\alpha_1 = 0.62$, $\alpha_2 = 0.57$.

Интегральный коэффициент в этом случае такой:

$$\alpha = 0.4 \cdot 0.62 + 0.6 \cdot 0.57 = 0.59.$$

Коэффициент больше 0.57, следовательно, можно принять слово, а вместе с ним и предложение:

clause("*ЭДС характеризуется температурным коэффициентом*").

В итоге распознавания и коррекции получено следующее соответствие слов (см. табл. 2)

Таблица 2

Правка слов

До правки	После правки
ЧЕМ	ЧЕМ
АДС	ЭДС
ПАРАМЕТРИЗУЕТСЯ	ХАРАКТЕРИЗУЕТСЯ

Ответом служит найденное предложение:

«*ЭДС характеризуется температурным коэффициентом*».

Локализация ответа. Структура предложения определяется грамматическими правилами следующего вида.

ПРЕДЛОЖЕНИЕ: – <БЛОК СУЩЕСТВИТЕЛЬНОГО> <БЛОК ГЛАГОЛА>
<БЛОК СУЩЕСТВИТЕЛЬНОГО>
БЛОК СУЩЕСТВИТЕЛЬНОГО: – <ПРИЛАГАТЕЛЬНОЕ> <СУЩЕСТВИТЕЛЬНОЕ>
БЛОК СУЩЕСТВИТЕЛЬНОГО: – <СУЩЕСТВИТЕЛЬНОЕ> <СУЩЕСТВИТЕЛЬНОЕ>
БЛОК СУЩЕСТВИТЕЛЬНОГО: – <СУЩЕСТВИТЕЛЬНОЕ>
БЛОК ГЛАГОЛА: – <НАРЕЧИЕ> <ГЛАГОЛ>
БЛОК ГЛАГОЛА: – <ГЛАГОЛ>
СУЩЕСТВИТЕЛЬНОЕ: – <любое слово Z – аргумент функтора noun(Z)>
ПРИЛАГАТЕЛЬНОЕ: – <любое слово Z – аргумент функтора adj(Z)>
ГЛАГОЛ: – <любое слово Z – аргумент функтора verb(Z)>
НАРЕЧИЕ: – <любое слово Z – аргумент функтора adverb_mode(Z) или adverb_time(Z)>

Для локализации ответа на вопрос «Чем характеризуется ЭДС?» нужно определить грамматическую структуру вопроса, т. е. распознать, какие блоки представляют слова «ЭДС» и «характеризуется». В настоящей работе этот вопрос опускаем ввиду ограниченности рамками статьи. Недостающий блок будет соответствовать ответу на вопрос, именно: «*температурным коэффициентом*».

Заключение. Для решения поставленной задачи коррекции распознавания речи разработана программа на языке C++, реализующая представленный в статье алгоритм. Основной вопрос стоял о степени релевантности ответов, даваемых алгоритмом. Слова, состоящие из семи и более букв, распознавались правильно в 100% примеров даже при двойных ошибках (пропуск буквы, замена буквы, нарушение порядка двух смежных букв). Процент распознавания более коротких слов с одиночными и двойными ошибками составил 77%. Полученные результаты позволяют рекомендовать алгоритм к практическому применению.

Литература

1. Формальный метод нечеткого поиска персональной информации / А. В. Бондаренко [и др.] // Препринты ИПМ им. М. В. Келдыша [Электронный ресурс]. – 2009. – № 64. – 25 с. – Режим доступа: http://library.keldysh.ru/preprint.asp?id=2009-64. – Дата доступа: 10.03.2013.

2. Navarro, G. A guided tour to approximate string matching // ACM Computing Surveys. – 2001. – Vol. 33, № 1. – P. 31–88.

Поступила 04.03.2013