ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Сборник трудов
1967 г. Института математики СО АН СССР Выпуск 28

ОБ ИСПОЛЬЗОВАНИИ ЯЗЫКОВОЙ ИЗБЫТОЧНОСТИ ДЛЯ ПОВЫШЕНИЯ НАДЕЖНОСТИ АВТОМАТИЧЕСКОГО РАСПОЗНАВАНИЯ РЕЧЕВЫХ СИГНАЛОВ

Г.Я. Волошин

I. Постановка задачи

Задача построения алгоритмов, позволяющих использовать языковую избыточность для повышения надёжности автоматического распознавания речевых сигналов, достаточно важна в практическом отношении. Уже неоднократно предпринимались попытки использовать лингвистические сведения в распознающих автоматах. Однако применявшиеся методы опирались на отрывочные данные о языке, имели весьма ограниченные возможности и были приспособлены только к конкретным распознающим автоматам.

Можно сослаться, например, на известные устройства Олсона и Белара [I], Сакаи и Дошита [2], в которых осуществляется сравнение распознанных элементов с некоторыми типовыми разрешёнными их комбинациями. Вполне понятно, что подобные методы можно использовать лишь в отдельных, частных случаях, например, при ограниченном словаре. К тому же среди разрешённых комбинаций не все являются равноправными, некоторые используются часто, другие очень редко. Этот факт никак не учитывается в упомянутых работах. Таким образом, значительный интерес представляет и теоретическая сторона этого вопроса.

Действительно, весьма целесообразно искать наиболее формализованное решение, не привязываясь к конкретному виду

распознаваемых элементов. Это позволит применять его в любых иерархических распознающих автоматах, если распознаваемые элементы нижней ступени образуют последовательности (распознаваемые элементы следующей ступени) с избыточностью. Примеров таких задач можно привести много: распознавание речи, распознавание рукописных и печатных осмысленных текстов, целый ряд задач обработки фотоснимков, данных геологоразведки и т.д.

Формальная методика использования избыточности полезна и в том отношении, что её можно было бы применять на различных иерархических уровнях процесса автоматического распознавания.

Иерархичность процесса восприятия речи человеком можно в настоящее время считать установленным фактом. Содержимое каждой из ступеней этой иерархической структуры еще недостаточно известно, но это нисколько не мешает поиску формального решения поставленной здесь задачи.

Проиллюстрируем возможность применения единой методики использования языковой избыточности при распознавании речи.

Исходя из современного уровня знаний о процессе восприятия речи человеком, можно считать, что на участке речевого сигнала, соответствующего фонеме, не содержится вся информация, необходимая для достаточно надёжного её распознавания [x) [3]. Существенная часть информации об анализируемой фонеме приходится на соседние участки речевого сигнала. Если это явление назвать взаимным влиянием фонем, то по данным Л.А. Чистович, Л.Р. Зиндера, Л.В. Бондарко и ряда других авторов участок распространения взаимного влияния фонем соответствует приблизительно открытому слогу. При восприятии речевых сигналов на коротких (порядка 20 мсек) участках производится некоторая его перекодировка, то есть даётся приближённое описание сигнала с помощью определённой системы признаков, причём считается, что такое описание не позволяет однозначно (или с высокой степенью надёжности) относить данный звук к какой-либо фонеме. Окончательное решение о фонеме принимается после обзора достаточно длительного интервала речевого сигнала, например, открытого слога (или ещё большего участка).

Эту модель можно представить в следующем виде. На коротких участках сигнала осуществляется предварительная перекодировка, причём полученное описание не позволяет нам точно опознать произносимую фонему, но зато мы можем вынести решение о

x) Разумеется, это утверждение не является бесспорным.

группе фонем, к которой должен принадлежать анализируемый звук. Все фонемы, входящие в эту группу, мы можем выписать в виде столбца в порядке убывания апостериорной вероятности (суммарная по столбцу вероятность должна быть равна или близка к единице). По окончании звучания достаточно крупного речевого элемента мы получим матрицу промежуточных решений:

$$\begin{matrix} a_1^1 & a_1^2 & \dots & a_1^N \\ a_2^1 & a_2^2 & & a_2^N \\ \vdots & \vdots & & \vdots \\ a_m^1 & a_n^2 & & a_c^N \end{matrix} , \qquad (I)$$

из которой нужно выбрать наиболее вероятную последовательность (наборы элементов в столбцах в общем случае различны). Постановка задачи не изменится от того, что мы будем рассматривать в качестве элементов матрицы (I): отрезки по 20 мсек, фонемы или слоги, а в качестве более крупного речевого элемента - открытый слог, словоформу или фразу соответственно. То же самое может относиться и к нахождению наиболее вероятной синтагмы или фразы по таким элементам, как словоформы. Следует отметить, что подобные матрицы можно строить по иерархической структуре. Действительно, если у нас имеется матрица (I), составленная, например, из ненадёжно распознанных участков речевых сигналов фиксированной длительности (положим, по 20 мсек), а крупным элементом является открытый слог, то результаты обработки такой матрицы можно использовать как исходные данные для построения матрицы, например, "слог-словоформа", затем "словоформа-синтагма (или фраза)" и т.п. Таким образом, решение задачи о поиске наиболее вероятной последовательности из исходной матрицы типа (I) было бы достаточно общим в плане практического применения.

Вполне понятно, что рассмотрение матрицы (I) имеет смысл только в том случае, если мы будем использовать какие-либо сведения о взаимосвязи столбцов друг с другом. При этом взаимосвязи не должны быть все допустимыми и равноправными, ибо в противном случае достаточно из каждого столбца выбрать элемент, имеющий максимальную апостериорную вероятность. К сожалению, достаточно полными строгими, "аналитическими", если можно так выразиться, сведениями о структуре русского языка мы в настоящее время не располагаем, поэтому, пожалуй, основные сведения подобного рода, которые можно использовать сейчас на

практике, – это данные статистической обработки русской речи: частоты встречаемости фонем, открытых слогов, словоформ, сочетаний двух фонем (диад), сочетаний трех фонем (триад), некоторые временные характеристики и т.д. Кроме того, можно использовать некоторые детерминированные правила типа:
– необходимость и достаточность одной ударной гласной в слове;
– запрещение некоторых сочетаний звуков и т.п.

Следует отметить, что учёт частоты встречаемости сочетаний более чем трёх, четырёх элементов затруднён, поскольку при этом (как будет показано ниже) резко возрастает сложность задачи, да и отсутствуют соответствующие статистические данные о русской речи, полученные на материале, обеспечивающем достаточно высокую достоверность результатов.

2. Математическая модель

Рассмотрим сначала случай учёта взаимосвязи элементов только двух соседних столбцов. В начальный момент времени система находится в нулевом состоянии (на вход автомата не подаются реализации, подлежащие распознаванию). В первый момент времени она может перейти в одно из состояний $a_1, \dots a_n$. Выпишем в строку вероятности $p^*_{0a_1}, \dots, p^*_{0a_n}$ (так называемые начальные вероятности), соответствующие частоте появления элементов $a_1, \dots a_n$ на первом месте в последовательностях (в случае речи ими являются лингвистические вероятности фонем). Далее умножим эти вероятности на соответствующие апостериорные вероятности, выданные распознающим автоматом. После этого произведем нормировку полученных вероятностей таким образом, чтобы выполнялось условие

$$\sum_{i=1}^{n} p_{0a_i} = 1$$

Теперь вероятность правильного опознавания в первый момент времени a_i-го элемента равна p_{0a_i}. Перейдем ко второму моменту времени. Построим матрицу переходных вероятностей $p^*_{a_i a_j}$ появления элементов a_j вслед за a_i в последовательности на первом и втором месте соответственно: x)

	a_1	a_2	a_3	...	a_n
a_1	p^*_{11}	p^*_{12}	p^*_{13}		p^*_{1n}
a_2	p^*_{21}	p^*_{22}	p^*_{23}		p^*_{2n}
a_3	p^*_{31}	p^*_{32}	p^*_{33}		p^*_{3n}
⋮					
a_n	p^*_{n1}	p^*_{n2}	p^*_{n3}		p^*_{nn}

Умножим все элементы каждого столбца на апостериорные вероятности, выданные распознающим автоматом в первый момент времени для каждого элемента, стоящего в вершине того или иного столбца. Например, автомат принял решение о том, что пришедшая на его вход в первый момент времени реализация является либо образом a_1 с вероятностью $p_1 = 0{,}7$, либо a_2 – с вероятностью $p_2 = 0{,}3$. Тогда мы все элементы первого столбца умножаем на 0,7, второго – на 0,3, а все остальные – на 0.

После этого осуществляется нормировка матрицы таким образом, чтобы сумма вероятностей в каждой строке была равна I. Элементами полученной матрицы будут $p^{(2)}_{a_i a_j}$, где верхний индекс соответствует моменту времени.

Совершенно аналогичным способом строим матрицы переходных вероятностей $p^{(k)}_{a_i a_j}$ для всех последующих моментов времени.

---

x) Если используемые лингвистические вероятности получены с привязкой к номеру столбца исходной матрицы (например, вероятность появления сочетания фонем $a_i\, a_j$ на первом месте в слове и т.п.), то это только усилит предельные возможности метода. Необходимые статистические данные нетрудно получить, если распознается ограниченный набор последовательностей (слов). При неограниченном же словаре в настоящее время можно использовать лишь не привязанные к номерам столбцов матрицы (I) вероятности, поскольку более "сильной" лингвистической информацией мы пока что не располагаем. Вполне понятно, что чем более подробными языковыми сведениями мы воспользуемся, тем большего успеха добьёмся в устранении ошибок исходного распознающего автомата. В дальнейшем изложении мы не будем особо оговаривать качество лингвистических данных, так как это не влияет на формальные выкладки.

Поскольку в конце концов мы должны из каждого столбца исходной матрицы (I) элементов (но не из матрицы переходных вероятностей) выбрать только по одному элементу, то становится очевидным, что мы имеем дело с простой неоднородной цепью Маркова [4]. Действительно, у нас имеется конечная система единственно возможных и несовместимых состояний $a_1, \dots a_n$, которые сменяют друг друга через конечные промежутки времени, причем состояние a_k стохастически зависит лишь от состояния a_{k-1}. Матрица переходных вероятностей нормирована по строкам и зависит от момента времени. Таким образом, все признаки простой неоднородной цепи Маркова имеют место.

Теперь нам нетрудно вычислить вероятность любого элемента в любой момент времени, причём эта вероятность будет учитывать взаимосвязи соседних элементов.

Так, полная вероятность появления элемента a_p в τ-й момент времени будет:

$$P_{a_p}^{(\tau)} = \sum_{i,j,k\dots l} P_{oa_i}^{(1)} P_{a_i a_j}^{(2)} \cdot P_{a_j a_k}^{(3)} \dots P_{a_l a_p}^{(\tau)} . \qquad (2)$$

Можно использовать и другие методы вычисления $P_{a_p}^{(\tau)}$, известные из теории цепей Маркова, однако в этом нет необходимости, поскольку формула полной вероятности (2) достаточно наглядна и легко реализуема на ЭВМ (по крайней мере, в том случае, когда последовательности достаточно короткие и число возможных состояний невелико, что и имеет место в рассматриваемой речевой задаче). Вычисление $P_{a_p}^{(\tau)}$ можно реализовать последовательно для $\tau = 1,2,3,\dots$

$$\begin{aligned} P_{a_p}^{(1)} &= P_{oa_p}^{(1)}, \\ P_{a_l}^{(2)} &= \sum_i P_{oa_i}^{(1)} \cdot P_{a_i a_l}^{(2)}, \qquad (3) \\ P_{a_m}^{(3)} &= \sum_l P_{a_l}^{(2)} \cdot P_{a_l a_m}^{(3)}, \\ &\dots\dots\dots \end{aligned}$$

После этого остаётся выбрать в столбцах матрицы (I) по одному элементу с максимальными вероятностями, которые получены по формулам (3).

Перейдем теперь к вопросу об учете взаимосвязей трех и более элементов. В этом случае мы получаем сложную неоднородную цепь Маркова, поскольку состояние системы в k-й момент времени стохастически зависит от состояния системы в несколь-ких предшествующих моментах времени.

Известно (см.[4]), что сложную цепь Маркова можно привести к простой. При этом матрица переходных состояний будет значительно больше, чем для простой цепи. Так, если для односвязной цепи (в ней учитываются связи двух элементов) матрица переходных вероятностей имеет порядок n, то для двухсвязной цепи (в ней учитываются связи трех элементов) матрица имеет порядок n^2, для трехсвязной цепи - n^3 и т.д. Для ν-связной цепи элементами матриц будут вероятности $P_{a_{i_1} a_{j_2} \dots a_{l_{\nu+1}}}$ сочетаний $\nu+1$ элементов $a_{i_1} a_{j_2} \dots a_{l_{\nu+1}}$.

Как и в случае односвязной цепи, основой для построения матриц переходных вероятностей является матрица, состоящая из лингвистических вероятностей сочетаний элементов $P^*_{a_{i_1} a_{j_2} \dots a_{l_{\nu+1}}}$.

В каждый новый (k-й) момент времени распознающий автомат выдаёт апостериорные вероятности элементов $P_{a_1}^{1(k)}, P_{a_2}^{1(k)}, \dots P_{a_n}^{1(k)}$. Необходимо все элементы столбцов, в которых записаны вероятности $P^*_{a_{i_1} a_{j_2} \dots a_{l_{\nu+1}}}$ умножить на $P_{a_l}^{1(k)}$ аналогично тому, как мы это делали для односвязной цепи. После этого остается произвести нормировку вероятностей по строкам.

Выпишем последовательный процесс вычисления вероятностей состояний для двухсвязной цепи:

$$\begin{aligned} P_{a_p}^{(1)} &= P_{oa_p}^{(1)}, \\ P_{a_l}^{(2)} &= \sum_i P_{oa_i}^{(1)} \cdot P_{a_i a_l}^{(2)}, \qquad (4) \\ P_{a_m}^{(3)} &= \sum_{p,l} P_{a_p}^{(1)} P_{a_p a_l a_m}^{(3)} \cdot P_{a_l}^{(2)}, \\ P_{a_q}^{(4)} &= \sum_{l,m} P_{a_l}^{(2)} P_{a_l a_m a_q}^{(4)} P_{a_m}^{(3)}. \end{aligned}$$

По окончании процесса счёта по формулам (4) остаётся выбрать для каждого момента времени наиболее вероятный элемент.

Как видно из (4), учёт взаимосвязи трёх элементов усложняет счет, поскольку приходится вычислять не простые суммы,

как в (3), а двойные. При использовании взаимосвязей четырех элементов потребуется вычисление тройных сумм и т.д.Таким образом, мы вынуждены ограничить длину используемых взаимосвязей, дабы не выйти за пределы возможностей современной вычислительной техники.

Достоинствами рассмотренного метода учёта лингвистических взаимосвязей элементов являются простота алгоритма вычисления вероятностей, удобство его реализации на ЭВМ и,как следствие, сравнительно малое время счёта. Конечно, эти преимущества касаются задач типа речевых, где число возможных состояний системы невелико (в каждый момент времени порядка десяти, как максимум), последовательности элементов достаточно коротки (не более I0 - I5) и учитываются взаимосвязи не более 3-х-4-х элементов (хотя бы потому, что сейчас ещё отсутствуют необходимые лингвистические данные для построения более сложных цепей). Вполне понятно, что основные затраты памяти при расчёте вероятностей по формулам типа (3) и (4) будут приходиться на исходные (лингвистические) матрицы переходных вероятностей, поэтому целесообразно искать пути минимизации необходимой лингвистической информации.

Интересно отметить, что, зная характер ошибок исходного распознающего автомата, мы можем ответить на вопрос, какими эти ошибки станут, если мы будем применять для их исправления лингвистические данные (имеются в виду вероятности сочетаний элементсв). Эту задачу решать заранее чрезвычайно полезно,так как, во-первых, мы узнаем, добьемся ли мы поставленной цели, применив к какому-либо конкретному распознающему автомату предложенный здесь метод исправления ошибок, и, во-вторых,если добьемся, то какой минимальной ценой (достаточно ли будет использовать сочетания только двух элементов или большего их числа).

Итак, пусть нам задана усредненная матрица ошибок исходного распознающего автомата, полученная при контрольных испытаниях

	a_1	a_2	a_3	...	a_n
a_1	P_{11}	P_{12}	P_{13}		P_{1n}
a_2	P_{21}	P_{22}	P_{23}		P_{2n}
a_3	P_{31}	P_{32}	P_{33}		P_{3n}
⋮					
a_n	P_{n1}	P_{n2}	P_{n3}		P_{nn}

Пусть, кроме того, известны лингвистические вероятности порядка r и ниже

$$P_{k_1 k_2 \dots k_r},\quad P_{k_1 k_2 \dots k_{r-1}},\quad P_{k_1 k_2, \dots k_{r-2}},\dots P_{0 k_1}.$$

Нам необходимо получить новую матрицу ошибок, которая будет иметь место после применения лингвистических вероятностей.

Пересчет матрицы будем производить по столбцам.

Для элементов, стоящих на первом месте в последовательности, новое значение элементов искомой матрицы получается по формуле

$$P^*_{k_1 e} = \frac{P_{k_1 e}\, P_{0 k_1}}{\sum\limits_{k_1} P_{k_1 e}\, P_{0 k_1}},$$

где индекс при k обозначает порядковый номер элемента a_k в последовательности,

$P_{k_m e}$ - числа, взятые из исходной матрицы ошибок,

$P_{0 k_m}$ - а далее $P_{k_n k_m}$ - лингвистические вероятности.

Для элементов, стоящих на втором месте в последовательности, получаем

$$P^*_{k_2 e} = \frac{P_{k_2 e} \sum\limits_{e'} \sum\limits_{k_1} P^*_{k_1 e'} \cdot P_{k_1 k_2}}{\sum\limits_{k_2} P_{k_2 e} \sum\limits_{e'} \sum\limits_{k_1} P^*_{k_1 e'} P_{k_1 k_2}}.$$

Аналогично для элементов, стоящих на q -м месте ($q \geq r$)

$$P^*_{k_q e} = \frac{P_{k_q e} \sum\limits_{e'} \dots \sum\limits_{e(q-1) k_{q-1}} \dots \sum\limits_{k_{q-r+1}} P^*_{k_{q-r+1} e'} P^*_{k_{q-r+2} e''} \dots P^*_{k_{q-1} e(q-1)} P_{k_{q-r+1} k_{q-r+2} \dots k_q}}{\sum\limits_{k_q} P_{k_q e} \sum\limits_{e'} \dots \sum\limits_{e(q-1) k_{q-1}} \dots \sum\limits_{k_{q-r+1}} P^*_{k_{q-r+1} e'} P^*_{k_{q-r+2} e''} \dots P_{k_{q-1} e(q-1)} P_{k_{q-r+1} k_{q-r+2} \dots k_q}}.$$

Для речевых сигналов r невелико (порядка нескольких единиц), поэтому вычисление $P_{k_q e}$ не вызывает серьезных затруднений.

Отметим, что при распознавании непрерывного потока речи вычисление $P^*_{k_q e}$ можно осуществлять лишь для случая $q \geq r$. Действительно, человек слитно произносит при нормальном темпе речи приблизительно 20-30 фонем. И из них только первые 2-4 будут иметь вероятности, отличные от $P^*_{k_q e}$, вычисленные для случая $q \geq r$. Для практических целей этими отличиями в ряде случаев можно пренебречь. Необходимо лишь всегда иметь в виду, что ошибки в распознавании начальных ($q < r$) элементов последовательностей труднее поддаются исправлению,

так как мы не можем по отношению к ним в полной мере использовать имеющуюся в нашем распоряжении лингвистическую информацию.

Обсудим некоторые особенности вышеизложенного метода. Прежде всего следует отметить, что по формулам типа (3) и (4) мы вычисляем полную вероятность состояния системы в κ -й момент времени, то есть считаем, с какой вероятностью в момент κ система перейдет в состояние a_p независимо от её состояний в промежуточные моменты времени. Это выражается в осуществлении суммирования в (3) и (4) по всем возможным промежуточным состояниям. Тем самым мы придерживаемся "осторожной" стратегии в выборе наиболее вероятной последовательности, минимизируем риск, учитываем при выборе решения на κ -м шаге возможность того, что на предыдущих шагах мы могли принять ошибочные решения. Возможна и другая, более "рискованная" стратегия: вычислять конкретные вероятности всех возможных последовательностей и затем выбирать последовательность с максимальной вероятностью.

Для примера рассмотрим односвязную систему с тремя последовательными элементами и четырьмя возможными состояниями a_1, a_2, a_3 и a_4. Начальные вероятности имеют следующие значения: $P_{oa_1} = 0,8$; $P_{oa_2} = 0,2$. $P_{oa_3} = 0$; $P_{oa_4} = 0$.

Зададим матрицу переходных вероятностей

	a_1	a_2	a_3	a_4
a_1	0	0	0,6	0,4
a_2	0	0	0,2	0,8
a_3	0,45	0,55	0	0
a_4	0,6	0,4	0	0

(5)

Если воспользоваться формулами полной вероятности, то наиболее вероятной окажется последовательность $a_1 a_3 a_1$. Если же вычислить конкретные вероятности всех последовательностей, то наиболее вероятное сочетание будет иметь вид: $a_1 a_3 a_2$.

В дальнейшем два рассмотренных метода будем называть соответственно методом полной вероятности и методом наиболее вероятной последовательности.

Метод наиболее вероятной последовательности (НВП) "сильнее" метода полной вероятности (ПВ) в том смысле, что он имеет потенциальную возможность исправлять более грубые ошибки распознающего автомата за счет лингвистических вероятностей. Но зато он и "рискованнее" метода ПВ, поскольку ошибочное решение в каком-либо звене может привести к дополнительным (по сравнению с методом ПВ) ошибкам при принятии решений в последующих звеньях ("последействие" ошибочного решения).

По всей видимости, вопрос о целесообразности применения того или иного метода нужно решать, опираясь на конкретные характеристики исходного распознающего автомата.

Рассмотрим подробнее метод НВП. Вполне понятно, что он сложнее в реализации, чем метод ПВ, поскольку требуется расчет вероятностей всех возможных последовательностей в матрице (I). Вообще говоря, мы можем решить эту задачу методом полного перебора. Однако такой приём наталкивается на трудности, преодолеть которые не позволяют возможности вычислительной техники как настоящего, так и ближайшего будущего времени.

Действительно, если матрица (I) содержит, например, IO столбцов, а в каждом столбце по 8 элементов (в общем случае в разных столбцах может быть разное число элементов), то всего возможно 8 последовательностей, причём расчёт вероятности каждой из них требует выполнения значительного количества вычислительных операций. Очевидно, что такой подход явно нецелесообразен. Ниже излагается один из методов решения этой задачи, позволяющий сравнительно легко находить наиболее вероятные последовательности элементов, используя возможности современной вычислительной техники.

Сначала рассмотрим случай использования взаимосвязей только двух соседних элементов. Как отмечалось выше, здесь мы имеем дело с простой неоднородной цепью Маркова. После построения матриц переходных вероятностей для всех моментов времени мы можем вычислить вероятность любой последовательности

$$a_i^1 a_j^2 \dots a_q^{N-1} a_k^N$$

по формуле

$$P_{a_i^1 a_j^2 \dots a_q^{N-1} a_k} = P^{(1)}_{oa_i} P^{(2)}_{a_i a_j} \dots P^{(N)}_{a_q^{N-1} a_k^N}. \qquad (6)$$

Теперь необходимо организовать направленный перебор возможных последовательностей, для чего предлагается исполь-

зовать идеи и методы сетевого планирования. [x]

Рамки журнальной статьи не позволяют подробно остановиться на данном вопросе, поэтому мы ограничимся лишь некоторыми определениями и построениями, приводящими исходную матрицу (I) к сетевому графику (систематическое изложение математических вопросов, связанных с обработкой сетевых графиков, можно найти, например, в [5]).

Граф, в котором две вершины могут быть соединены дугами, ориентированными только в одну сторону, называется антисимметрическим. Антисимметрический граф, в котором существует лишь одна вершина, не имеющая входящих дуг, и лишь одна вершина, не имеющая выходящих дуг, и каждой дуге которого приписано некоторое числовое значение, мы будем называть сетью. Последовательность дуг, в которой конец каждой предыдущей совпадает с началом последующей, называется путём. Поскольку в сетевом планировании вершины графа интерпретируют события, а дуги – работы (точнее, длительности работ во временных сетях), числа, приписываемые дугам, должны быть неотрицательными. Рассмотрим упрощенную матрицу типа (I), состоящую из четырех столбцов с тремя элементами в каждом из них (см. рис. I , область внутри штриховой линии). Введем начальное a_0 и конечное a_k события. Соединим все события ориентированными дугами, как показано на рис. I. Дугам припишем следующие числовые значения:

$$a_0 \text{ и } a_i^1 - (-\ell g P_{0\,a_i^1}^{(1)});$$

$$a_i^1 \text{ и } a_j^2 - (-\ell g P_{a_i^1 a_j^2}^{(2)});$$

$$a_j^2 \text{ и } a_t^3 - (-\ell g P_{a_j^2 a_t^3}^{(3)});$$

$$a_t^3 \text{ и } a_s^4 - (-\ell g P_{a_t^3 a_s^4}^{(4)});$$

$$a_s^4 \text{ и } a_k - 0.$$

В такой системе обозначений наша матрица становится сетью, и в этой сети путь минимальной длины [xx] проходит через

---

x) Возможно использование и других методов направленного перебора.

xx) В сетевом планировании обычно ищут путь максимальной длины или так называемый критический путь. В отличие от него путь минимальной длины мы будем называть минимальным.

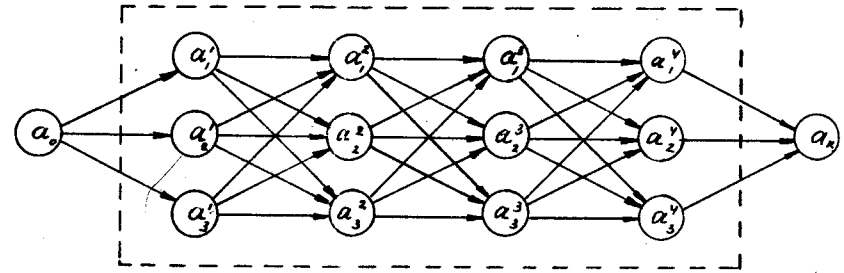

Рис. I. Простейший сетевой график для поиска наиболее вероятной последовательности.

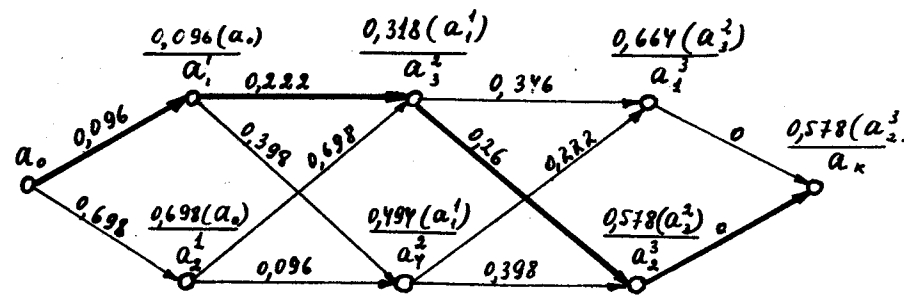

Рис. 2. Пример анализа сети по алгорифму Форда.

вершины (события), соответствующие наиболее вероятной последовательности элементов исходной матрицы, а длина T_{min} минимального пути связана с вероятностью выделенной последовательности (6) следующим соотношением:

$$T_{min} = -lgP^{(1)}_{oa_i'} - lgP^{(2)}_{a_i' a_j^2} - lgP^{(3)}_{a_j^2 a_t^3} - lgP^{(4)}_{a_t^3 a_5^4} = -lgP_{a_i' a_j^2 a_t^3 a_5^4},$$

откуда

$$P_{a_i' a_j^2 a_t^3 a_5^4} = 10^{-T_{min}}. \qquad (7)$$

Для поиска минимального пути будем использовать алгорифм Форда (см.[5]). Предварительно только нужно ввести единую нумерацию сети, двигаясь в порядке возрастания номеров сверху вниз по первому столбцу (в нашем случае a_o), переходя затем на соседний справа столбец (a_1^1, a_2^1, a_3^1) и т.д.

Подробно описать алгорифм Форда здесь не представляется возможным, поэтому мы ограничимся лишь пояснением на конкретном примере. Снова обратимся к матрице (5) переходных вероятностей. Сеть, соответствующая этому случаю, представлена на рис. 2 [x] (те дуги, которые соответствуют нулевым вероятностям, в сети не представлены, поскольку длины их равны ∞ и в минимальный путь они заведомо не войдут).

Припишем событию a_o вес, равный нулю. Переходим к событию a_1'. В него входит единственная дуга. К её длине (0,096) прибавляем вес события, из которого она выходит. Полученное число впредь будет весом события a_1', его мы записываем над символом a_1' и помечаем номер события, из которого сюда пришла дуга (a_o). То же самое проделываем и с событием a_2'. Далее рассматриваем a_3^2. В это событие входят две дуги. К длине каждой из них прибавляем веса событий, из которых они вышли. Из двух полученных чисел выбираем минимальное (0,318), записываем его над символом a_3^2 и впредь будем считать весом события a_3^2. Здесь же помечаем номер события, из которого пришла выбранная дуга (a_1'). Аналогичные операции проделываем с событиями a_4^2, a_1^3, a_2^4 и a_k. Вес события a_k и является величиной T_{min}. В нашем случае T_{min} = 0,578.

---

x) На такой элементарной сети (всего 8 возможных разных путей) преимущества алгорифма Форда по сравнению с полным перебором незначительны, но они весьма быстро возрастают с усложнением сети.

Пересчитав эту величину в вероятность по формуле (7), получаем 0,264.

Теперь выпишем минимальный путь. Над символом a_k помечено событие a_2^3. Дугу, соединяющую a_2^3 с a_k, обводим жирной линией. Далее, над символом a_2^3 помечено событие a_3^2. Дугу, соединяющую a_3^2 с a_2^3, обводим жирной линией. Потом аналогичным способом переходим к событию a_1^1 и, наконец, к a_o. Теперь жирной линией выделен минимальный путь: $a_o a_1' a_3^2 a_2^3 a_k$. Поскольку события a_o и a_k пусты, приходим к окончательному решению: наиболее вероятной последовательностью является сочетание $a_1 a_3 a_2$, а конкретное значение его вероятности равно 0,264.

При небольших дополнительных затратах памяти можно осуществить поиск нескольких наиболее вероятных последовательностей. Это особенно необходимо в тех случаях, когда результаты обработки сети являются исходными данными для построения сети следующего иерархического уровня.

Теперь покажем, как можно, не меняя метода вычислений, учитывать взаимосвязь трех элементов. Обратимся к упрощенной исходной матрице размером 3x4:

$$\begin{matrix} a_1^1 & a_1^2 & a_1^3 & a_1^4 \\ a_2^1 & a_2^2 & a_2^3 & a_2^4 \\ a_3^1 & a_3^2 & a_3^3 & a_3^4 \end{matrix} \qquad (8)$$

Введением дополнительных событий, точнее, формальным расщеплением элементов исходной матрицы мы можем опять привести её к простой цепи Маркова. Этот прием в какой-то мере является графическим (сетевым) выражением общего принципа (см.[4]) приведения сложных цепей Маркова к простым, о чём уже говорилось выше. На рис. 3 представлена сеть, соответствующая этому случаю.

Пунктирными линиями обведены группы вершин (событий) сети, соответствующие элементам исходной матрицы (8). Правило формального расщепления элементов следующее: каждый элемент какого-либо столбца матрицы (8) (за исключением первого и последнего столбцов) расщепляется на столько вариантов, сколько элементов в предыдущем столбце исходной матрицы. В сети каждый вариант какого-либо элемента имеет входящие дуги только от вариантов одного и того же элемента предыдущего столбца (это утверждение не относится к последнему перед a_k столбцу).

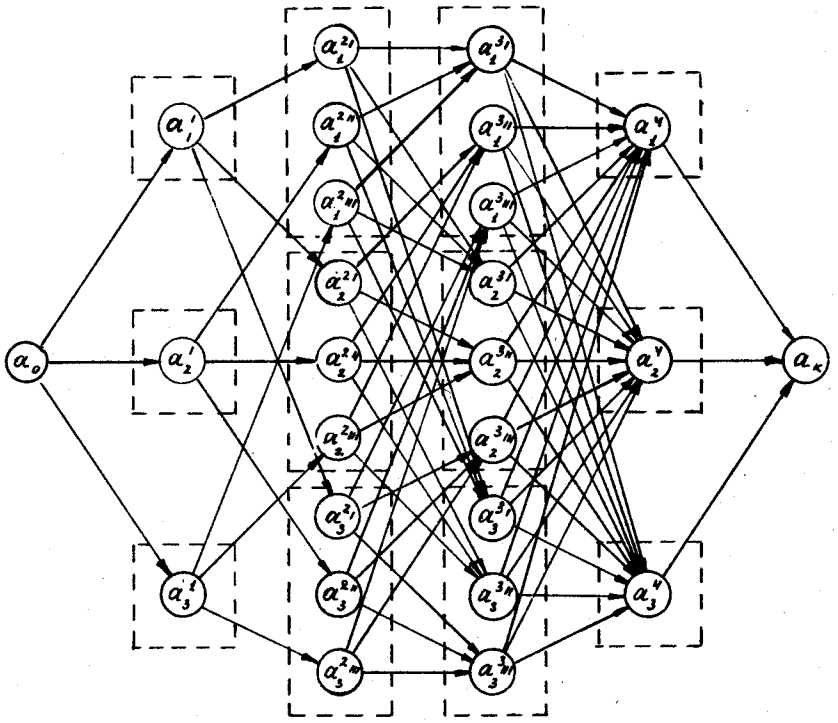

Рис. 3. Сетевой график, учитывающий взаимосвязь трех элементов .

При таком построении сети оказывается, что если имело место, скажем, событие a_1^{31}, то перед ним могли произойти события a_1^{2I}, a_1^{2II} или a_1^{2III} (они все соответствуют только одному элементу исходной матрицы), а перед каждым из них, в свою очередь, одно и только одно из событий a_1^1, a_2^1 или a_3^1 соответственно. Таким образом, мы можем учитывать взаимосвязь трех элементов исходной матрицы (8). Вместе с тем, поскольку дуги интерпретируют вероятностные связи между событиями, из рис. 2 мы видим, что любое событие непосредственно зависит лишь от событий предыдущего столбца сети (но не исходной матрицы), что является необходимым признаком простой цепи Маркова.

В качестве примера, иллюстрирующего возможность использования зависимости трех элементов матрицы (8), рассмотрим событие a_2^4 (рис. 3). Его вероятность может быть получена следующим образом:

$$P_{a_2^4} = P_{a_i^2} \cdot P_{a_i^2 a_j^3 a_2^4}.$$

Поскольку $i = 1,2,3$ и $j = 1,2,3$, то в событие a_2^4 должны входить 9 дуг, что мы и видим на рис. 3. Можно записать $P_{a_2^4}$ и в форме зависимости только двух соседних элементов

$$P_{a_2^4} = P_{a_0 a_i^1} P_{a_i^1 a_j^{2(c)}} P_{a_j^{2(c)} a_k^{3(c)}} P_{a_k^{3(c)} a_2^4}.$$

По числу переменных индексов в событие a_2^4 должна входить 81 дуга. Но в связи с тем, что из a_i^1 выходят только 3 дуги, а не 9, и из $a_j^{2(c)}$ тоже 3 вместо возможных 9 (остальные связи являются запрещёнными, то есть длина их равна ∞), общее число дуг, входящих в a_2^4, сокращается в 9 раз.

Учёт взаимосвязи четырёх элементов приводит к еще большему расширению сети, соответствующей исходной матрице (8). Для сравнения размеров сетевых графиков, получающихся при реализации предложенного метода, подсчитано количество событий и дуг, соответствующее учету взаимосвязи двух, трёх и четырех элементов. Для простоты предполагалось, что во всех столбцах матрицы число (k) элементов одинаково (в реальных задачах это необязательно). Число столбцов равно n. Полученные формулы, а также некоторые числовые примеры сведены в таблицу I.

Ограничения, накладываемые на размеры исходных матриц, следующие: при учете зависимости двух элементов должно иметь

место соотношение $n \geq 2$, трех элементов $n \geq 3$, четырех элементов $n \geq 4$.

Штриховкой в табл. I отмечены варианты сетей, не позволяющие производить расчет с использованием лишь оперативной памяти ЭВМ М-20. В ориентировочное время t обработки сети на ЭВМ М-20 не включены затраты на ввод и вывод данных, а также на формирование сети. Здесь также не учтены затраты памяти, необходимой для хранения информации о взаимосвязях элементов. Как уже отмечалось выше, эти затраты могут оказаться существенными, поэтому целесообразно искать методы минимизации необходимых лингвистических данных, позволяющих снизить процент ошибок исходного распознающего автомата до требуемого уровня.

3. Области применения

Предлагаемые методы устранения (уменьшения) ошибок распознавания мелких элементов за счёт использования языковой избыточности могут быть применены на базе уже существующих устройств (или методов) автоматического распознавания речи. Остановимся прежде всего на автоматах, работающих по классической схеме: членение непрерывного речевого потока на участки, соответствующие фонемам, затем распознавание фонем. В таких устройствах возможны ошибки двух видов: ошибки при членении и ошибки при распознавании.

Если отвлечься от ошибок членения и если автомат при распознавании анализируемого участка выдает в качестве решени набор фонем наиболее близких в пространстве используемых параметров, причём каждой фонеме из этого набора приписывается своя апостериорная вероятность, то методы исправления ошибок (НВП и ПВ) могут быть применены непосредственно в том виде, как они описаны во втором разделе настоящей работы. Однако большинство существующих автоматов выдает в качестве решения не набор, а одну наиболее вероятную фонему. И в этом случае предлагаемый метод может быть использован, для чего необходимо набрать статистику ошибок: надёжность распознавания каждой фонемы, список возможных сбоев со своими вероятностями. Так, например, если автомат выдает в качестве решения фонему "И", то из контрольных испытаний мы должны знать, что надежность этого решения, скажем, 60%. Кроме того, нам должно быть известно,

Таблица I.

Числовые графиков.

Числовые примеры	Количество событий			Количество дуг		
	для дуг	для троек	для четвер	для дуг	для троек	для четвер
	$nk+2$	$2(k+1)+(n-2)k^2$				
$k=10$ $n=10$	102	822	7122	920	8120	71120
t мин	30÷50					
$k=5$ $n=10$	52	212	912	235	1035	4535
t мин	20÷25	150÷200				
$k=3$ $n=10$	32	80	206	87	231	609
t мин	10÷15	100÷150	300÷400			

что данный автомат в 20% случаев называл фонемой "И" фонему "$б$", в 10% случаев - фонему "j" и в 10% случаев - фонему "$и^{е}$". Тогда мы знаем, что если автомат распознал фонему "$и$", то вероятность этого события 0,6, и мы можем построить столбец исходной матрицы со следующими апостериорными вероятностями:

$$\begin{aligned} и &- 0{,}6 \\ б &- 0{,}2 \\ j &- 0{,}1 \\ и^{е} &- 0{,}1 \end{aligned}$$

Суммарная вероятность должна быть равна I или близка к ней. Вполне понятно, что такой метод построения матриц хуже предыдущего, но и он может быть использован для исправления ошибок.

Теперь обратимся к ненадежности членения.

Сначала рассмотрим метод ПВ. В том случае, если автомат может выдавать лишние границы членения, необходимо обеспечить возможность вывода лишних фонем из выделяемой последовательности. Для этого достаточно ввести во все матрицы переходных вероятностей величины $P_{a_i z}$, где z - "пустой" элемент, причем $P_{a_i z}$ должно соответствовать вероятности появления лишнего сигнала членения после фонемы a_i. Если подряд могут появиться два лишних сигнала членения, то нужно знать вероятность такого события и занести её в матрицу переходных вероятностей.

Вполне понятно, что перед нормированием матрицы необходимо вероятности всех переходов к "непустым" элементам умножить на вероятность правильного членения (то есть на $1-q$, где q - вероятность появления лишнего сигнала членения).

В том случае, если исходный распознающий автомат может пропустить нужную границу, возникает необходимость восстановления фонем, не вошедших в исходную матрицу типа (I). Для этого достаточно ввести ещё один элемент $\bar{z}$ и включить в матрицу переходных вероятностей величину вида $P_{a_i \bar{z}}$, соответствующую вероятности пропуска автоматом одной фонемы, следовавшей за a_i. В данном случае нам должны быть известны из контрольных испытаний уже три вероятности: вероятность правильного членения, вероятность пропуска границы и вероятность появления лишней границы (очевидно, в сумме они должны составлять единицу).

Можно предложить следующий алгоритм восстановления пропущенных фонем. Если на $(k+1)$-м шаге наиболее вероятным (по

полной вероятности) окажется элемент $\bar{z}$, то необходимо вернуться на k-й шаг и в качестве $(k+1)'$-й матрицы переходных вероятностей воспользоваться нормированной матрицей лингвистических вероятностей в чистом виде, поскольку автомат не выдал нам апостериорных вероятностей и трудно опираться на что-либо более разумное, чем лингвистические данные.

После принятия решения на $(k+1)'$-м шаге переходим вновь на $(k+1)$-й шаг и обрабатываем соответствующую ему матрицу переходных вероятностей.

Из приведенного описания видно, что ненадежность членения существенно усложняет алгоритм исправления ошибок. Особенно это касается возможности пропуска фонем исходным распознающим автоматом. Если лишние элементы могут быть устранены простым добавлением [x] элемента z в исходные матрицы типа (I), то в случае пропуска фонем мы вынуждены возвращаться и обрабатывать дополнительные матрицы, причем матрицы эти довольно громоздкие (для русской речи что-нибудь порядка 50x50 в случае учёта взаимосвязи двух элементов). Наличие апостериорных вероятностей значительно сокращает исходную матрицу, ибо целому ряду фонем приписывается нулевая вероятность.

Поэтому устранение лишних фонем может быть достигнуто более простыми средствами, чем восстановление пропущенных. В этом плане целесообразно добиваться такого положения, чтобы исходный распознающий автомат не делал пропусков границ пусть даже ценой увеличения числа лишних сигналов членения.

Теперь обращаемся к методу НВП. Нам известно, что он сложнее, чем метод ПВ. В связи с этим усиление его возможностей более затруднительно. Особенно это касается восстановления пропущенных фонем. Действительно, нам неизвестны апостериорные вероятности, а поэтому мы вынуждены перебирать весь список фонем, что приводит к катастрофическому увеличению размеров сети и невозможности получения решения за приемлемое машинное время (по крайней мере, это относится к современным вычислительным средствам). Поэтому, хотя принципиальная возможность решения поставленной задачи и имеется, мы на ней останавливаться не будем. Гораздо проще обстоит дело в том случае, если исходный автомат не делает пропуска границ, а ставит лишние, и чем меньше лишних границ п о д р я д, тем проще

---

x) Конечно, самих матриц переходных вероятностей становится больше, ведь некоторые из них оказываются лишними.

получится сеть. Рассмотрим случай сети, учитывающей взаимосвязь двух фонем и предусматривающей возможность появления ложного сигнала членения, причём ложные (лишние) сигналы членения могут следовать не более одного раза п о д р я д. Соответствующий сетевой график представлен на рис. 4. В качестве исходных данных для построения сети использована матрица (8) размером 3x4.

Как видно из рис. 4, каждый столбец исходной матрицы дополняется, как и в методе ПВ, определённым числом "пустых" элементов z, что позволяет получить путь, обходящий любой столбец. Число "пустых" элементов в i-м столбце равно числу непустых элементов $(i-1)$-го столбца (это утверждение не относится к последнему перед a_k столбцу сети). Известно, что надёжность членения зависит от того, какие фонемы мы разделяем. Эту зависимость желательно учитывать при составлении сети.

Как видно из рис. 4, полученная сеть даже проще сети, учитывающей взаимосвязь трех элементов (рис. 3). Если учесть возможность появления двух ложных границ подряд, то сеть усложнится (увеличится приблизительно до размеров сети рисунка 3, число событий будет меньше, но число работ - больше). Можно составить сеть, учитывающую взаимосвязь трёх элементов и возможность появления лишних границ. Если эти лишние границы следуют не более одной подряд, то размеры полученной сети будут меньше, чем у сети, учитывающей взаимосвязь четырех элементов, но не учитывающей ненадежности членения (о конкретных размерах сетей можно получить представление из таблицы I).

Таким образом, метод НВП позволяет исправлять ошибки не только распознавания, но и членения, особенно если эти ошибки выражаются в виде появления лишних границ.

Рассмотрим теперь случай, когда автомат не производит членения на фонемы, а принимает решения на фиксированных временных интервалах малой длительности (например, через каждые 20 мсек). Этот вариант более интересен в свете современных представлений о восприятии речи человеком. Остановимся только на примере метода НВП. Здесь появляется необходимость в построении, по крайней мере, двух иерархических ступеней сетевых графиков. Первая ступень в качестве исходных данных использует показания распознающего автомата. Законы, связывающие мелкие элементы друг с другом, уже не исчерпываются лингвистиче-

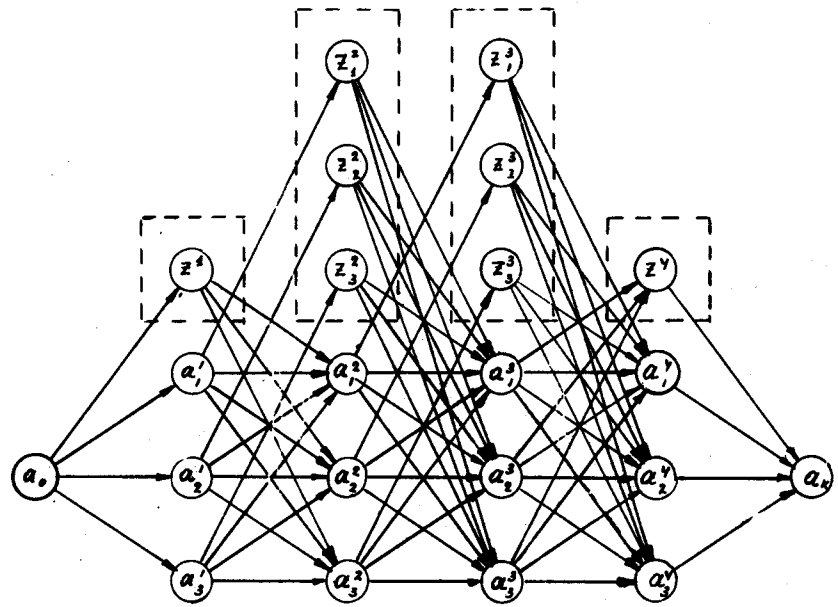

Рис. 4. Сетевой график, учитывающий ненадежность членения.

скими вероятностями. Необходимо использовать и временные соотношения речи. В частности, теперь допустимы такие сочетания фонем, как "ааа..." или "ммм...", причём априорные вероятности таких сочетаний определяются соотношением средней длительности данной фонемы и длительности участков, на которых принимаются решения. Подобная сеть "стабилизирует" показания распознающих автоматов на квазистационарных участках звуков, т.е. вместо ошибочной последовательности, например "ммум", будет выдана последовательность "мммм" (если в одном столбце с "у" была и фонема "м"), так как сочетание "мум" в данном случае является запрещенным, ибо невозможно в течение 10-20 мсек перейти с фонемы "м" на "у" и снова вернуться к "м". После работы первичной сети можно объединить все соседние одинаковые фонемы в одну[x)] и таким образом укрупнить исходную матрицу. Далее работает сеть, применяемая для автоматов, членящих речевой поток на фонемы. Пример функционирования такой иерархической сети будет приведен в следующим разделе. В данном случае практически не приходится опасаться пропуска фонем, зато появление лишних фонем весьма вероятно, а по предлагаемой методике, как было показано выше, легче справляться с лишними элементами, чем с недостающими (пропущенными).

Рассмотренные методы могут быть непосредственно использованы прежде всего в автоматических стенографах, фонемных вокодерах, читающих машинах для слепых, уменьшая количество ошибок в печатаемом тексте и делая его "читабельным" (это наглядно иллюстрируется в следующем разделе). Вместе с тем не исключено, что ряд ошибок не будет исправлен. Это не так опасно, если оконечным распознающим "автоматом" является человек (что и имеет место в стенографии, вокодерной технике связи, читающих машинах для слепых и т.п.). Иначе обстоит дело в том случае, если окончательное решение принимает автомат (принимает к исполнению устные команды). Тогда после выбора наиболее вероятной последовательности фонем может оказаться, что эта последовательность не совпадает ни с одной из команд, имеющейся в памяти. На этот случай можно предложить несколько возможных вариантов алгоритма дальнейшей обработки.

---

x) Правила объединения мелких элементов, конечно, несколько сложнее. Для гласных это простое объединение, для глухих взрывных - это объединение с предшествующим участком нуля звука (смычкой) и т.д.

Организация переспроса до тех пор, пока выделенная последовательность не совпадёт с какой-нибудь "известной" автомату командой (эту процедуру целесообразно применять при восприятии автоматом особо ответственных команд).

Введение для каждой команды не одной, а нескольких эталонных последовательностей.

Выбор в сети (или в исходной матрице) только тех последовательностей, которые имеются в словаре, подсчет вероятностей их и выбор максимума.

У всех этих алгоритмов есть один недостаток, они требуют полного совпадения выделенной последовательности с одной из эталонных. В связи с этим не исключена возможность частых переспросов, что может сделать эксплуатацию системы неудобной. Было бы целесообразнее для автоматов, распознающих конечный набор команд, выносить суждение о степени "близости" выделенной и эталонных последовательностей. Для этого необходимо ввести, точнее, найти соответствующее пространство и метрику, которые были бы инвариантными по отношению к конкретному содержанию словаря, то есть нужно, по существу, изучить структуру речевого кода, что является весьма важной и интересной задачей дальнейших исследований.

Легко видеть, что использование методов НВП и ПВ позволяет производить экстраполяцию, то есть предсказывать наиболее вероятные элементы следующего шага. Это можно использовать для оптимизации метода принятия решений. Действительно, если всё пространство разбито на области, соответствующие тому или иному распознаваемому элементу, и если нам нужно перебирать эти области при распознавании, чтобы узнать, в какую из них попала анализируемая реализация, то порядок обхода этих областей будет небезразличен. Желательно так организовать перебор, чтобы можно было попасть в искомую область на первых же шагах.

В таком случае порядок перебора можно определять по предсказаниям рассмотренных в настоящей работе алгоритмов (ПВ и НВП).

4. Экспериментальная проверка

Небольшой экспериментальной проверке был подвергнут лишь

метод НВП.[x] В качестве исходного материала был взят входной язык АЛГОЛ-60 и "озвучен". Всего получилось 123 различных словоформы. После перевода этих словоформ в фонетическую транскрипцию были набраны необходимые статистические данные о взаимосвязях фонем (с учётом их положения в слове). Все фонемы (60 штук) были разбиты на 17 групп (перекрывающихся) по 4-5 фонем в каждой.

Предполагалось, что автомат, распознающий фонемы, выделяет группу безошибочно, а внутри группы допускает сбои. Эти сбои моделировались с помощью датчика случайных чисел. Было использовано 5 градаций апостериорных вероятностей: 1) 0,5; 2) 0,3; 3) 0,1; 4) 0,07; 5) 0,03. Рассмотрим на примере, как вносятся ошибки в исходный текст. Допустим, что мы искажаем слово с' э́ м' (семь). В группу, соответствующую фонеме с' входят ещё фонемы С, Ц, Ш', Щ. С помощью датчика случайных чисел каждой из этих фонем приписывается одна из пяти апостериорных вероятностей. Получаем один столбец матрицы. Затем аналогично поступаем с группой фонем э', ъ, и, ^, содержащей э', и - м, н, м', н', содержащей м'. Получили матрицу

ц	э́	м
с	и ъ	м'
с'		н
Ш'	и	
щ	^	н'

Если из этой матрицы выбрать просто наиболее вероятные фонемы, то получим слово Ц э́ м (подчёркнуты ошибки, а над ними стоит цифра, указывающая порядковый номер нужной фонемы в данном столбце). Результаты обработки таких матриц приведены в табл. 2. После внесения ошибок в исходный текст они составляли 86% от всех фонем (220 из 254). Сеть, использующая взаимосвязи диад, уменьшила число ошибок до 15 (6%), а триады совершенно устранили ошибки. Следует отметить, что к триадам можно было бы и не прибегать, если использовать правило необходимости и достаточности одной ударной гласной в слове. При этом число ошибок (без использования триад) уменьшилось бы до 7 (2,7%).

---

[x] Вся лингвистическая часть эксперимента ("озвучение", транскрибирование и подсчёт необходимых лингвистических вероятностей) была выполнена Л.С. Юдиной.

Такие хорошие результаты объясняются тем, что исходный словарь довольно ограничен и статистические данные, полученные на нём, "сильны" (много запрещенных комбинаций). На ещё меньших словарях статистические данные превратятся просто в список разрешенных комбинаций фонем, и тогда описанные здесь алгоритмы совпадут с методикой Олсона и Белара, которая является их частным случаем. При увеличении словаря статистические данные будут более "размытыми" и подействуют менее радикально. Тогда, по всей вероятности, придётся в полную силу использовать триады, а, возможно, и тетрады.

Был промоделирован один пример построения двух иерархических ступеней сетевых графиков. В качестве исходного материала была использована осциллограмма слова во́сьм' (восемь). После разбиения всего интервала слова на участки по 20 мксек получилось в в в в$_о$ в$_в$ о о о о о о о$_с$, о$_с$, с' с' с' с' с' с$_в$ б б б б$_и$ м' м' м' м' ("в$_о$" означает, что на этом интервале преобладает "в", но появились и колебания, характерные для "о"). Затем, как описано выше, моделировались показания распознающего автомата (предполагалось, что он принимает решения на участках по 20 мсек). Получили текст с ошибками

2 3 2 4 3 2 1 3 1 1 2 5 3 2 2 2 5 2 3
в о в в у у у о у о о у у с с' ш' с' с б у б и б н н' м' м' м'

После обработки этой матрицы первичной сетью и объединения следующих подряд одинаковых фонем в одну получили укрупненную матрицу

в	о́	с'	б	н
в'	у			м'

Обработав эту матрицу сетью, использующей диады и триады, получили слово " в о́ с' б м' ". Таким образом, все ошибки удалось устранить.

Вполне понятно, что приведенные примеры экспериментальной проверки алгоритма носят иллюстративный характер, они не привязаны к каким-либо конкретным автоматам. Вместе с тем очевидно и то, что такая привязка не повлечёт за собой принципиальных изменений метода.

№ № п/п	Исходное слово		Искаженное слово	Исправленное слово с учетом диад	Исправленное слово с учетом триад
1	ʌд'ин	(один)	ъ г н м	ʌд'ин	ʌд'ин
2	два́		о в' ъ	два́	два́
3	тр'и́	(три)	п' р ы	тр'и́	тр'и́
4	п'а́т'	(пять)	к' а к	п'а́к	п'а́т'
5	шэ́с'т'	(шесть)	чъш'т'	шэ́с'т'	шэ́с'т'
6	с'э́м'	(семь)	ц э м	с'э́м'	с'э́м'
7	но́л'	(ноль)	н'ур'	но́л'	но́л'
8	пу́ск		тус'т	пу́ск	пу́ск
9	сто́п		скук	ско́п	сто́п
10	цы́кл	(цикл)	с'ык'л'	цы́кл	цы́кл
11	пл'у́с	(плюс)	кл'уц	пл'у́с	пл'у́с
12	дл'а́	(для)	грʌ	дл'а́	дл'а́
13	с'и́гмʌ	(сигма)	с и о н' а	с'и́гмʌ	с'и́гмʌ
14	м'а́х'к'иј	(мягкий)	н'эф'т'ий	м'а́х'киј	м'а́х'к'иј
15	чи^е ты́р'ъ	(четыре)	шј п'ыл'и	чи^е ты́р'и	ч'и^е ты́р'ъ
16	во́с'ъм'	(восемь)	в'ушј н	во́с'ъм'	во́с'ъм'
17	д'э́в'ът'	(девять)	о'а в'ик	дэ́ в'ът'	д'э́в'ът'
18	м'и́нус	(минус)	н' ы н у ш'	м'и́нус	м'и́нус
19	кʌн'э́ц	(конец)	т ʌ м'ис'	кʌн'э́ц	кʌн'э́ц
20	ско́пкʌ	(скобка)	ц т'у т к э	ско́пка	ско́пкʌ
21	и́с'т'инʌ	(истина)	и с п'инʌ	и́с'т'и нʌ	и́с'т'инʌ
22	в'э́ктър	(вектор)	в ʌ к'т э р	в'ʌ́ктър	в'э́ктър
23	мʌс'с'и́ф	(массив)	н' з с' с ы х	мэ́с'с'и́ф	мʌс'с'и́ф
24	јэ́с'л'и	(если)	ъ иц л и	јэ́с'л'и	јэ́с'л'и
25	м'э́ткʌ	(метка)	н' и п' к'ʌ	м'э́ткʌ	м'э́ткʌ
26	цэ́лыј	(целый)	с' и р' ы и	цэ́лыј	цэ́лыј
27	д'э́с'ът'	(десять)	бэ ц и п'	дэ́с'ът'	дэ́с'ът'
28	то́чкʌ	(точка)	п' у ш т э	то́чкʌ	то́чкʌ
29	стро́к		ц т р у к	стро́к	стро́к
30	ско́пк'и	(скобки)	с т у т т' и	сто́пк'и	ско́пк'и
31	стрʌк'и́	(строки)	с' п' л' э т' и	стрʌк'и́	стрʌк'и́
32	склʌл'а́р	(скаляр)	с т ал ър'	стʌл'а́р	склʌл'а́р
33	прʌб'э́л	(пробел)	т р э о ʌ л'	прʌб'э́л	прʌб'э́л
34	ма́тр'ицʌ	(матрица)	м ъ к р ы ц э	мʌтр'ицʌ	ма́тр'ицʌ
35	и́пс'илън	(ипсилон)	й к с' и л ʌ м	и́пс'илън	и́пс'илън
36	бʌл'ша́јʌ	(большая)	о'ъ л ш' ъ и^е и^е	бʌл'ша́јʌ	бʌл'ша́јʌ
37	ру́скъјʌ	(русская)	л' ус' к'ʌ и и	ру́скъјʌ	ру́скъјʌ
38	зъп'и^е то́ј	(запятой)	з а п ы к у ъ	зъп'и^е то́ј	зъп'и^е то́ј
39	кʌвы́чк'и	(кавычки)	к а в и ш т ы	кʌвы́чк'и	кʌвы́чк'и
40	ко́с'инус	(косинус)	т у с ы м' у с	то́с'инус	ко́с'инус
41	ръз'д'и^е л'и́т'	(разделить)	л з жд или к	лъз'д'и^е ли́т'	ръз'д'и^е ли́т'
42	знʌчэ́н'иjъ	(значение)	з мз чэми ъи	знʌчэ́н'иjъ	знʌчэ́н'иjъ
43	пръцыду́рʌ	(процедура)	п лэ с' ыо ур' а	пръцыду́рʌ	пръцыду́рʌ
44	фу́нкцыјʌ	(функция)	ф' ум' тс' иий^е	фу́нкцыјʌ	фу́нкцыјʌ
45	ʌп'и^е ра́цыјʌ	(операция)	з кш лэ с' иъј	ʌп'и^е ра́цыјʌ	ʌп'ира́цыјʌ
46	зъп'и^е та́јʌ	(запятая)	жап ыкъъи^е	зъп'и^е та́јʌ	зъп'и^е та́јʌ
47	мʌс'с'и́въф	(массиво)	м' ас'цыв' аф'	м'э́с'с'и́въф	мʌс'с'и́въф
48	лъгʌр'и́фм	(логарифм)	рлдър' ыфн	лъгʌр'и́фм	лъгʌр'и́фм
49	кʌта́нг'ънс	(котангенс)	к' ъкэмдинс	кʌта́нг'ънс	кʌта́нг'ънс
50	ʌркс'и́нус	(арксинус)	эрк' цин' ус	э́ркс'инус	ʌркс'и́нус

Литература

I. М.А. Сапожков. Речевой сигнал в кибернетике и связи. Связьиздат., М, 1963.

2. T.Sakai,S.Doshita. The Automatic Speech Recognition System for Conversational Sound. IEEE Trans.On Electronic Computers,v.EC-12,N5,1963,pp.835-846.

3. Распознавание слуховых образов. Новосибирск, Изд-во Наука, Сиб.отд., 1966.

4. В.И. Романовский. Дискретные цепи Маркова. Гостехиздат, М-Л, 1949.

5. С.И. Зуховицкий, И.А. Радчик. Математические методы сетевого планирования. Изд.Наука, М. 1965.

Поступила в редакцию
21.VII.1967г.